Forum: Moteur de jeu GameBlender et alternatives

Topic: Bonne conception des classes (vertes, pour les enfants :-D)

Subject: Re: Python - l'instruction return

PostÃ© par: DaWaaaaghBabal

Contribution le : 24/3/2018 14:39:50

Citation :
Pour pouvoir appuyer sur un bouton et obtenir un effet, je dois pouvoir envoyer un signal grâce à ma souris. Mais si je suis ta réflexion, le script ne fonctionnera qu'une fois, par conséquent, je peux cliquer autant que je veux, je n'aurai pas mon effet si je ne le place pas dans "init".
J'entrevois que tu n'as pas compris ce qu'est la méthode __init__. Elle est appelée **quand tu crées un objet**. C'est un initialiseur (je disais constructeur, et dans la plupart des langages c'est la même chose, mais en Python les deux sont séparés), qui sert à (duh) initialiser l'objet qu'on vient de créer. C'est tout.

Quand tu cliques, tu lances un script de gestion du clic. Il fait sa tambouille. Cette tambouille peut impliquer la création d'objets. Si ton script crée deux objets, la méthode __init__ sera appelée une fois sur chacun. S'il n'en crée aucun, __init__ ne sera jamais appelée. Je ne vois pas ce que tu veux dire avec ton histoire de script qui ne s'exécute qu'une fois sauf si on le met dans une méthode qui n'a de raison d'être que si elle s'exécute une seule fois...

Citation :
Maintenant, supposons que l'on rajoute l'affichage du texte: le "post_draw" n'a besoin que d'être lancé qu'une fois, ok. Mais pour le reste ? Si mon script ne se lance qu'une fois globalement, mes autre textes s’afficheront ? Non.
J'ai décidément du mal à suivre ta logique. Tu as regardé le code modifié que je t'ai proposé ?

Citation :
Tout ça pour dire que, dans le BGE, on est dans le cas ou le script doit être appelé en boucle. Ce n'est pas un simple programme "pur python" que l'on lancerai comme un batch Windows.
... Si ?
Un script n'a pas à être exécuté en boucle. Ce n'est pas la seule option. À ce stade, lis la doc.
Et même si c'était la seule option... Ça n'a toujours aucun rapport avec la question.
Hook et toi ne vous comprenez peut-être pas, mais c'est lui qui a raison.

Citation :
Enfin, pourquoi devrai-je rappeler la fonction "write" alors que je peux simplement modifier un texte ? En plus, si je fais ça, ça va superposer l'ancien texte avec le nouveau, ce qui vas donner un effet gras sur le texte. C'est pas ça que je veux, moi.
Mais on ne parle pas de rappeler la fonction write(). Nulle part. Ne commence pas à répondre à des choses qu'on ne t'a pas proposées, sinon on n'est pas sortis de l'auberge.

Quant à la modification du texte... Je vais me répéter, et je déteste ça pour information : tu as

regardé mon code modifié ?
La méthode write écrit la liste de textes telle qu'elle est actuellement. SI tu modifies son contenu, ça modifie l'affichage.

Citation :
Beeenn, d'un côté, si tu n'as pas de cas concrets et que tu te contentes de faire des ateliers, je vois pas trop comment la personne extérieur peut apporter un jugement correct.

Non ? Le programmeur est censé savoir programmer comme un pro d'entrée de jeu ??? Ça, c'est "l'Allemand blond aux yeux bleu", si je puis me permettre. Et c'est la raison pour laquelle il y a du chômage au 21e siècle, en plus.
Vas-y, fais-toi passer pour une victime. Et sors un point Godwin. Ça va avoir un effet positif sur ma bonne volonté et me donner envie de t'aider, je t'assure.
Je t'invite à remarquer que ni moi ni Hook ne sommes payés pour te répondre ou pour passer ton code en revue. C'est du temps qu'on te consacre par pure bonté d'âme. Et il y a des limites au temps et à l'effort que je suis prêt à consacrer à un anonyme sur Internet. Surtout quand l'anonyme en question me donne l'impression de ne lire mes réponses qu'en diagonale.

Personne ici, à part ta parano ou ta stratégie de victimisation, ne prétend que tu es censé savoir programmer d'entrée de jeu. C'est pour ça que, comme tu as peut-être pu le constater, on consacre tous les deux un peu de temps à t'aider dans cet apprentissage. Alors ton guilt trip, tu le gardes pour toi bien sagement.

Citation :
Cette phrase me gêne. Si c'est un aimant à bug, comment faut-il rédiger le code pour que ce ne soit plus considéré comme tel ?
Cf le code que j'ai proposé.

Citation :
Si. Je l'ai expliqué plus haut: Dans mon projet, il y a 2 blends: le terrain et "barre chargement". Ils sont tous deux indépendants, par conséquent, ils n'ont aucun liens physique mais il faut lier l'un à l'autre pour l'avoir partiellement.

Là, tu es en train de me répéter pourquoi il y a un système pour faire passer des données dans tout le code. Ce n'est pas ce que j'ai demandé. J'ai demandé pourquoi chaque texte avait besoin d'y être, au lieu de les encapsuler quelque part. Et tu n'y réponds pas, parce qu'il n'y a pas de bonne raison. Encore une fois : cf le code que j'ai proposé.

Bon, attaquons ton cas concret maintenant.

Citation :
Voici un bout du début de mon code pour le terrain.
Remarque préliminaire : "mère", c'est aussi explicite et compréhensible que "toto".

Citation :
La classe mère contient des éléments qui sont censés se répéter un peu partout dans l'ensemble. J'ai estimé intéressant de mettre quelques fonctions là pour éviter de

réécrire à chaque fois ces même lignes.
Pour le vocabulaire, on appelle le fait de regrouper des éléments communs à plusieurs classes dans une classe-mère *factoriser*. Sur le principe, c'est bien.
J'ai un problème avec cette classe : elle a deux responsabilités. Premièrement, initialiser les variables qu'on utilise tout le temps genre obj et scene. Deuxièmement, lire des sons. Quel est le rapport ?
Et c'est facile à corriger, en la divisant en deux : une classe SceneElement qui reprend la méthode __init__, une classe AudioPlayer qui hérite de SceneElement et reprend les deux autres.

Citation :
Donc, "mere" ne s'exécute pas sans appels.
Euh, ça, ça ne veut rien dire. Au point que je ne peux même pas essayer de corriger cette phrase.

Pour passer à game : un poil de chipotage, game devrait hériter de mere, et utiliser super(self) au lieu d'une référence directe à mere. Ce sera plus souple, plus facile à modifier par la suite.
Plus grave : ton indentation, c'est n'importe quoi. Tu définis des méthodes les unes dans les autres sans raison, la hiérarchie apparente n'a pas de sens.
Voici une version corrigée de ce fichier. Ici encore, il convient d'en faire un module, genre game.py. Tu auras ensuite un script, placé à un endroit où il ne s'exécutera qu'une fois, qui donnera quelque chose comme :
import game from dictionary import dico current_game = Game() dico['current_game'] = current_game current_game.spawn_mob('cube loup', 'loup')
current_game.spawn_mob('cube renard', 'renard')
current_game.spawn_mob('cube sanglier', 'sanglier')
current_game.spawn_mob('cube corbeau', 'corbeau')
current_game.spawn_mob('cube boomslang', 'booms')
current_game.spawn_mob('cube biche', 'biche')
current_game.spawn_mob('cube thon', 'thon')
current_game.spawn_mob('Cube_cheval_charette', 'cheval char')
current_game.spawn_mob('Cube_raptor', 'raptor')

Et si ailleurs un autre script a besoin de spawn quelque chose, il va chercher le Game dans le dico, et appelle une des méthodes de spawn qu'il appelle.

EDIT j'oubliais un truc, concernant ton projet d'ensemble. Tu dis que tu as compris le principe de séparation des responsabilités, mais tu as des scripts de plusieurs milliers de lignes qui font tout... 4 000 lignes, c'est minimum 40 scripts. Pas 1.

Le volume de l'ensemble serait déjà un obstacle pour une revue de code d'ensemble ; clairement, je n'ai pas que ça à faire. Mais j'aurais pu prendre un module en particulier, à la rigueur, si c'était un concept applicable...