
PyEngine

Version 1.0.0

juil. 07, 2019

1	Introduction	3
2	Téléchargement et Installation	5
3	FAQ	7
4	Changelog	9
5	Théorie	13
6	Hello World	17
7	Enumérations	21
8	Exceptions	25
9	Window	27
10	World	29
11	Entity	31
12	MusicSystem	35
13	UISystem	37
14	EntitySystem	39
15	SoundSystem	41
16	EntitySystem	43
17	PositionComponent	45
18	SpriteComponent	47
19	TextComponent	49
20	PhysicsComponent	51

21	MoveComponent	53
22	ControlComponent	55
23	LifeComponent	57
24	Widget	59
25	Label	61
26	Image	63
27	Button	65
28	Entry	67
29	Color	69
30	Font	71
31	Vec2	73

Bienvenue sur la documentation de la bibliothèque PyEngine.

PyEngine est constamment en développement, la documentation est donc susceptible d'être changée. N'hésitez pas à y revenir dès que vous avez un problème.

Note : Il est important de rappeler que PyEngine est un projet OpenSource et développé par des personnes non professionnelles. Vous pouvez, vous aussi, y participer via le github.

Sommaire :

CHAPITRE 1

Introduction

PyEngine a été créé par LavaPower.

PyEngine se base sur PyGame pour fonctionner. Il a été fait pour être utilisé sur des jeux 2D de tout type : Platformer, Pong, Casse bricks. . .

Vous pouvez retrouver des tutoriels, des exemples et la documentation des différentes classes.

Note : PyEngine est encore très jeune et encore très limité.

Note : Cette documentation liste les méthodes et les attributs des classes qui peuvent être utilisées sans problème et sans risque.

Téléchargement et Installation

2.1 Dernière release (Méthode simple)

- Avoir Python et Pip installés
- Faire dans une console : `pip install PyEngine-2D`
- PyEngine est téléchargé et installé

2.2 Version en développement (Méthode moins simple)

- Avoir Python et Pip installés
- Télécharger et décompresser les fichiers du github (<http://github.com/pyengine-2D/PyEngine>)
- **Faire dans une console à l'endroit où sont les fichiers :** `python setup.py install.`
- PyEngine est téléchargé et installé

3.1 Qu'est-ce que PyEngine ?

PyEngine est une bibliothèque python permettant de créer des jeux vidéos 2D plus facilement. C'est une sorte de moteur de jeu très simplifié sans interface.

3.2 Pourquoi créer PyEngine ?

Pour créer un jeu vidéo en python, il existe déjà le très bon PyGame.

Mais en créant mon jeu, je devais créer des systèmes (comme le système d'entité) qui sont pourtant utiles pour tous. J'ai donc fait le choix de créer PyEngine (qui utilise PyGame lui même) (Et puis ça permet un bon entrainement en Python).

3.3 Quelles sont les dépendances de PyEngine ?

Si ce n'est Python, PyEngine utilise PyGame.

3.4 Quelles sont les plateformes où PyEngine est utilisable ?

Si vous pouvez utiliser PyGame et Python, vous pouvez utiliser PyEngine.

3.5 Je souhaite participer au développement de PyEngine, comment faire ?

Envoyez moi un message par Discord (LavaPower#2480) pour voir ce que vous pouvez faire.

Voici le changelog des versions de PyEngine.

4.1 V 1.2.0 : Property Update - 09/06/19

- All : Use property decorator
- All : Add annotation on function will be used by users
- Window : Modify management of Worlds
- Window : Created in middle of the screen
- Window : Can modify size
- GameState : Rename to World
- World : Remove has system function
- Entity-Exception : Replace WrongObjectError to TypeError
- Entity : Can remove component
- CameraSystem : Create basic camera
- MoveComponent : Remove speed
- TextComponent : Add background color
- TextComponent : Add scale
- TextComponent : Add rendered_size
- Label : Add background color
- Button : Add white filter when it is hovered
- Button : Can change image
- Vec2 : Create vector 2
- Color : Can be add, substract and compared
- Font : Can be compared
- Unit Tests : Create
- Bug Fix : Entity Text is not updated
- Bug Fix : Entity Text is not count in get_entity
- Bug Fix : MusicSystem return wrong volume
- Bug Fix : Window return wrong title
- Crash Fix : Crash when use Entry

- Crash Fix : Crash when use length setter of Vec2
- Crash Fix : Crash when use TextComponent
- Crash Fix : Crash when we use size of SpriteComponent
- Crash Fix : Crash when we use LifeComponent

4.2 V 1.1.2 : Patch Update 2 - 01/06/19

- UISystem : Add a show_debug function
- Optimization
- Bug Fix : EntitySystem give wrong id to Entities
- Bug Fix : EntitySystem is render after UISystem
- Bug Fix : Window is always in debug mode

Cette MAJ ne casse pas la compatibilité avec la précédente.

4.3 V 1.1.1 : Patch Update - 30/05/19

- Create and add PyEngine Logo
- Window : Add icon parameter
- Window : Use Color class
- TextComponent : Add text management
- Bug Fix : OutOfWindow don't take sprite size
- Critical Bug Fix : CollisionCallbacks is not defined in ControlComponent

Cette MAJ ne casse pas la compatibilité avec la précédente.

4.4 V 1.1.0 : General Update - 25/05/19

- LifeComponent : Remove creation of sprite
- LifeComponent : Add get_life and get_maxlife functions
- Entity : Add get_system function
- World : Remove world
- Enums : Move Enums in classes
- EntitySystem : Add function to remove entity
- UISystem : Add function to remove widget
- SoundSystem : Create
- Widgets : You can hide and show widgets
- Entry : You can use your own background
- Color-Colors : Create color class and colors enums
- Font : Create font class
- Optimisation of lib
- Bug Fix : Rotation of SpriteComponent don't work

4.5 V 1.0.2 : Fix Update 2 - 11/05/19

- Entity : Can get custom component
- Setup : Fix crash when pygame is not installed
- Setup : Don't get PyGame2

Cette MAJ ne casse pas la compatibilité avec la précédente.

4.6 V 1.0.1 : Fix Update - 10/05/19

- Enums : Add Controls in `__all__`
- Entity : Can add custom component

Cette MAJ ne casse pas la compatibilité avec la précédente.

4.7 V 1.0.0 : First Update - 09/05/19

- Components : Create LifeBarComponent, MoveComponent
- Components : Rework on system (Work with constructor)
- World-Enums : Create WorldCallbacks (OUTOFWINDOW)
- Components/SpriteComponent : Add `set_size` function
- Components/PhysicsComponent-Enums : Add CollisionCauses in CollisionCallback
- Components/PhysicsComponent : Add gravity management
- Components/ControlComponent : Add speed management
- Components/ControlComponent : Add controls management
- Components/ControlComponent-Enums : Add LEFTRIGHT and UPDOWN ControlType
- Components/ControlComponent-Enums : Add Controls Enums
- GameState-Window-World : Create GameState System
- Systems/UISystem : Create Widgets System
- Widgets : Create Label, Image, Button, Entry widget
- Window : Add title and background color management
- Exceptions : Rework on system (rename and remove useless exceptions)

4.8 V 0.2.0-DEV : Little Update - 25/04/19

- Components/PhysicsComponent : Collision Callback return object
- Systems/EntitySystem : Remove condition to add entity
- Window : Add a function to end game
- Setup.py : Add dependances (PyGame)

4.9 V 0.1.0-DEV : Initial Update - 19/04/19

- First Version

Avant de s'attaquer aux tutoriels pratiques, il est important de comprendre l'architecture général de PyEngine.

5.1 Architecture en arbre

Il est possible de voir l'architecture de PyEngine comme un arbre.

Ceci induit plusieurs choses :

- Chaque élément peut accéder à tous les autres éléments.
- Il y a un élément initial
- Il y a des éléments transitifs
- Il y a des éléments finaux

Note : Accéder à n'importe quel élément à partir d'un autre est parfois caché ou idiot mais c'est toujours possible.

5.2 Window, début de Tout

PyEngine prend comme élément initial Window. Celui ci correspond en fait à la fenêtre qui s'ouvre quand vous le créez.

C'est à partir de lui que l'on va créer notre architecture.

5.3 Les éléments transitifs

Notre fenêtre a besoin de monde pour fonctionner. Ceux ci sont simplement les Worlds. Ce sont eux qui vont être directement rattaché à Window et qui correspondent aux mondes que vous voyez.

Les éléments qui les suivent sont les Systems. Actuellement PyEngine possède 5 systèmes :

- EntitySystem : Gestionnaire des entités

- UISystem : Gestionnaire des widgets
- MusicSystem : Gestionnaire de la musique de fond
- SoundSystem : Gestionnaire des sons et bruitages
- CameraSystem : Gestionnaire de la caméra

MusicSystem, CameraSystem et SoundSystem n'ont pas d'enfant et font donc aussi office d'éléments finaux.

Mais EntitySystem peut avoir comme enfant des entités et UISystem peut avoir des widgets.

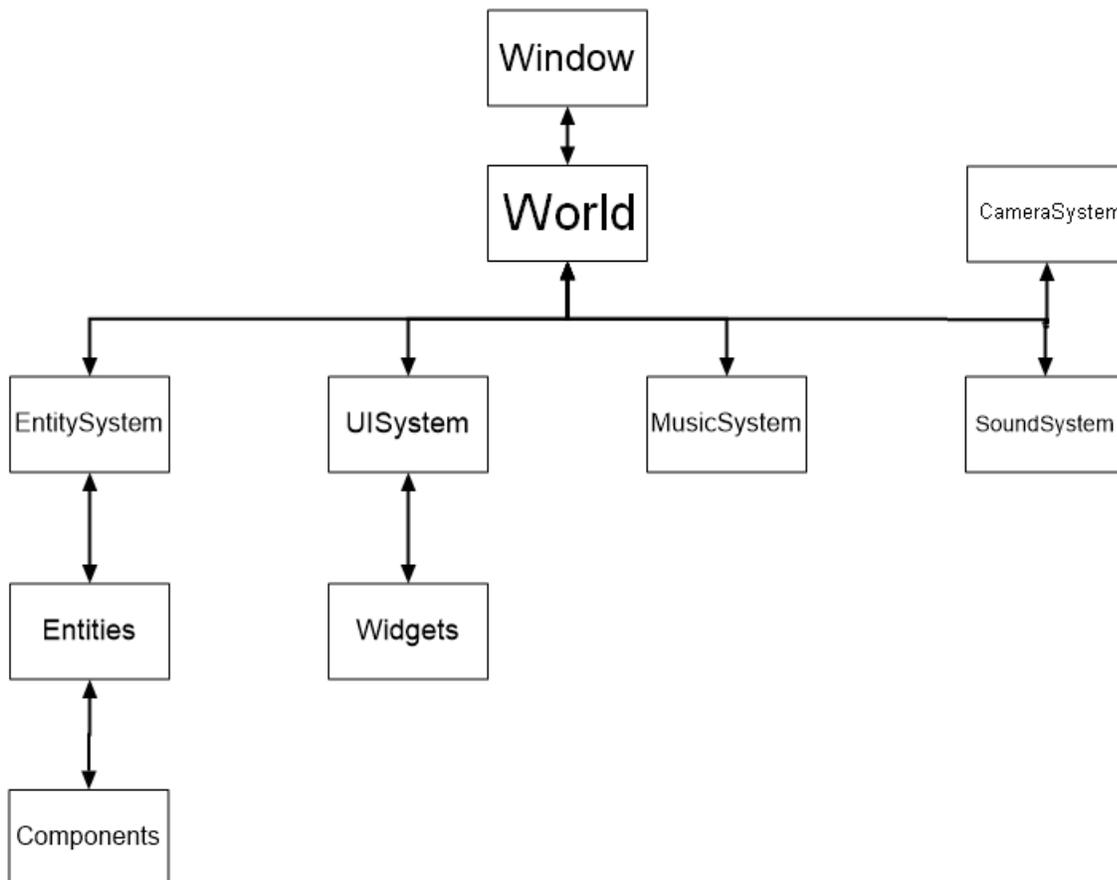
5.4 Les éléments finaux

Les entités ne sont pas forcément des éléments finaux. Elles ont dans la plupart des cas des composants les définissant. Ce sont ces composants qui font office d'éléments finaux.

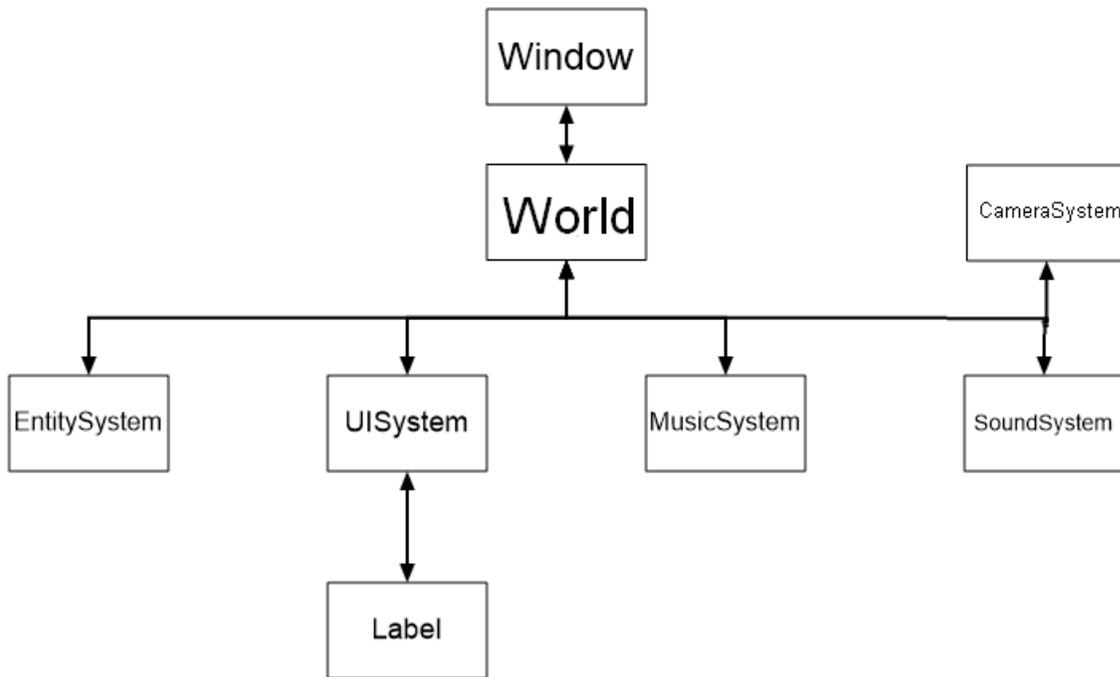
Du coup des widgets, ce sont bien les éléments finaux.

5.5 Représentation de l'architecture

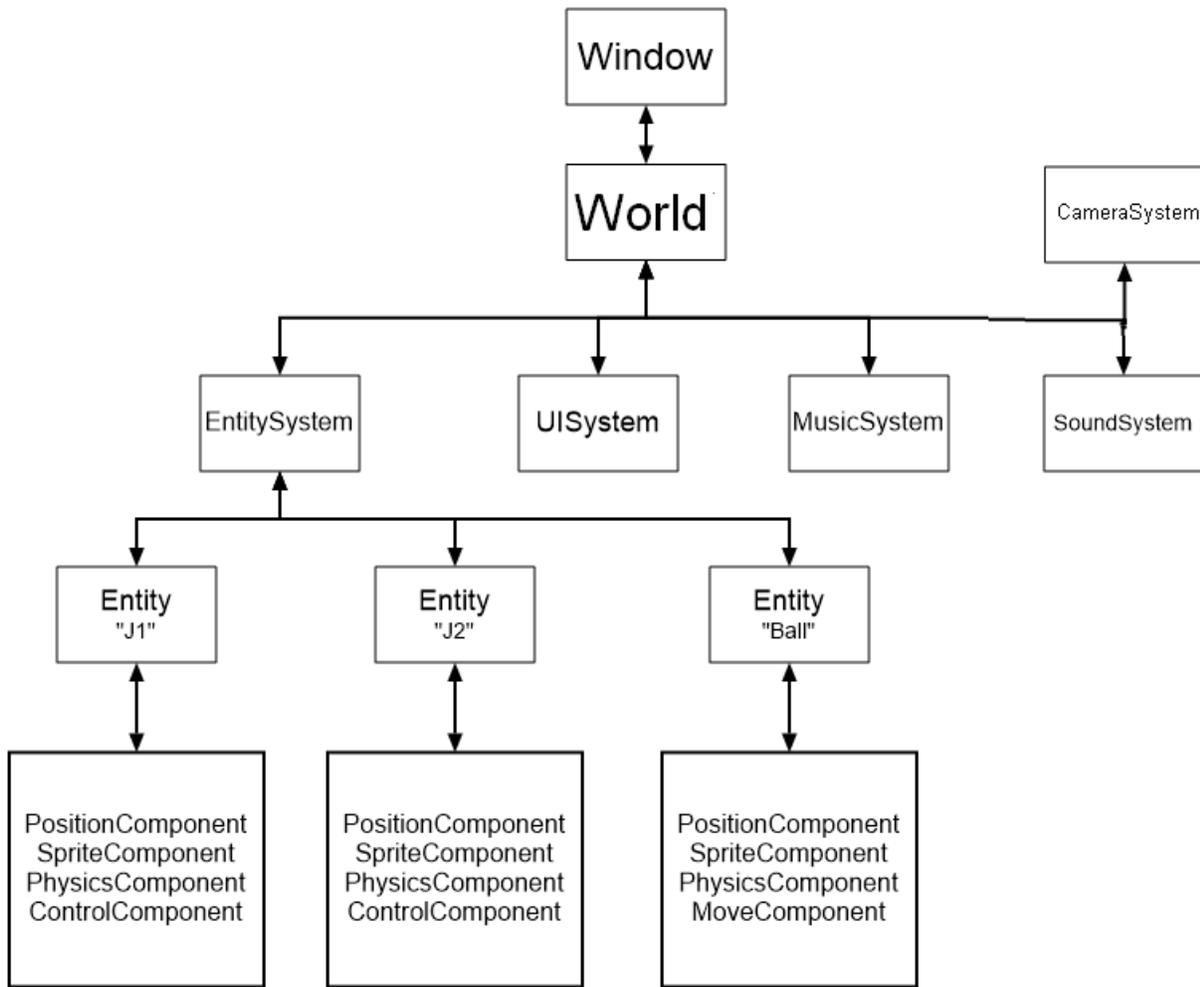
Architecture général :



Architecture du tutoriel HelloWorld :



Architecture de l'exemple Pong :



Hello World

Dans ce tutoriel, nous allons créer un programme très connu : le fameux Hello World.

Grâce à ce tutoriel, vous saurez créer une fenêtre graphique avec un état de jeu. De plus, vous saurez utiliser le widget Label venant du système d'UI.

6.1 Création de la fenêtre

La première étape est de créer la fenêtre graphique. Ici on va créer une fenêtre de 500 par 300 pixels avec un fond blanc.

Tout d'abord, il faut importer la classe de la fenêtre ainsi que de quoi utiliser les couleurs via :

```
from pyengine import Window # Window étant la classe de notre fenêtre.  
from pyengine.utils import Colors
```

Ensuite, il faut l'initialiser :

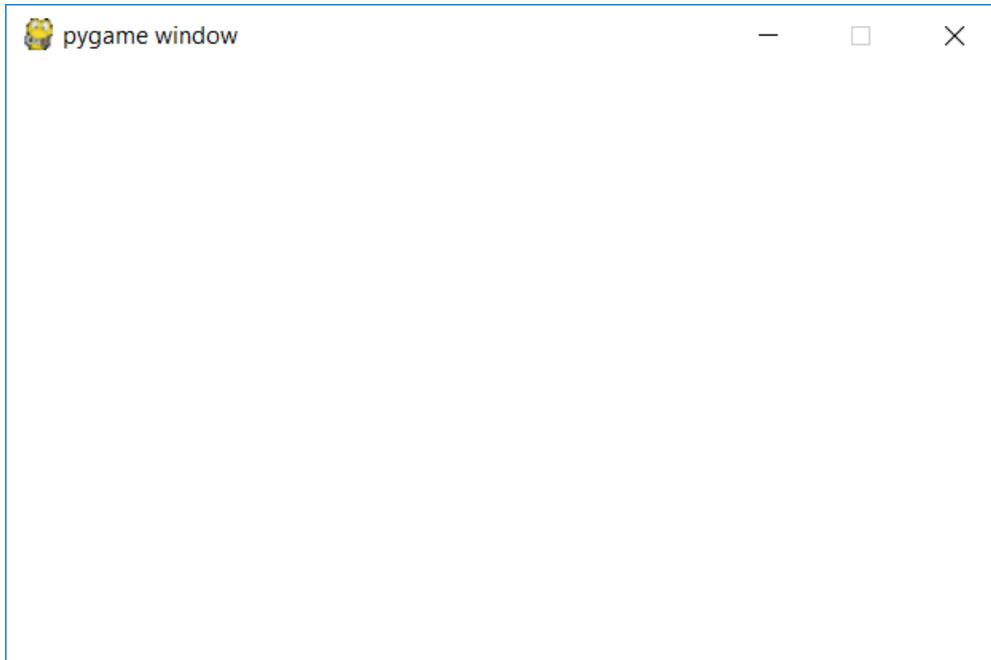
```
fenetre = Window(500, 300, Colors.WHITE.value)  
# 500 : Largeur  
# 300 : Longueur  
# Colors.WHITE.value : Couleur blanche
```

Si vous lancez ce code, vous verrez la fenêtre se lancer puis se fermer directement.

Pour régler ce problème, il faut lancer la boucle de la fenêtre. Pour cela, il suffit de faire :

```
fenetre.run()
```

Lancez le programme et vous devriez avoir ceci :



6.2 Création du texte

Maintenant, nous allons afficher notre texte.

Pour cela, nous allons utiliser le monde de notre fenêtre afin de récupérer le système qui gère l'ui.

```
from pyengine.Systems import UISystem

uisystem = fenetre.world.get_system(UISystem)
```

Ensuite, nous devons créer notre widget et l'ajouter à notre système :

```
from pyengine.Widgets import Label
from pyengine.Utills import Vec2

hello = Label(Vec2(0, 0), "Hello World !", Colors.BLACK.value)
# Vec2(0, 0) : Position x, y
# "Hello World !" : Texte
# Colors.BLACK.value : Couleur noire
uisystem.add_widget(hello)
```

Ce qui nous donne au final :

```
from pyengine import Window, GameState
from pyengine.Systems import UISystem
from pyengine.Widgets import Label
from pyengine.utills import Colors, Vec2

fenetre = Window(500, 300, Colors.WHITE.value)

uisystem = fenetre.world.get_system(UISystem)
```

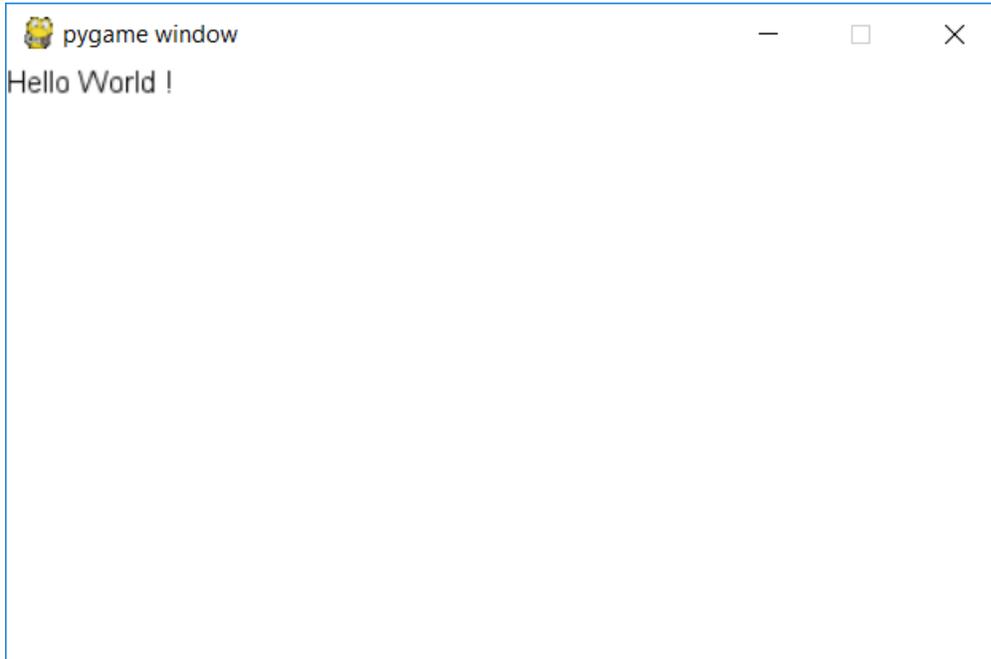
(suite sur la page suivante)

(suite de la page précédente)

```
hello = Label(Vec2(0, 0), "Hello World !", Colors.BLACK.value)
uisystem.add_widget(hello)

fenetre.run()
```

Avec comme résultat :



PyEngine incorpore beaucoup d'énumérations utilisées dans ses différentes classes.

Note : Pour les importer, il faudra donc importer soit le `__init__` soit la classe référente.

En voici la liste :

7.1 Colors

Description Correspond à des couleurs prédéfinies

Valeurs

- WHITE : Couleur blanche
- BLACK : Couleur noire
- BLUE : Couleur bleue
- RED : Couleur rouge
- GREEN : Couleur verte

Classe Color

Note : Pour utiliser ces couleurs, il faut utiliser la méthode `.value()` dessus. Exemple : `white = Colors.WHITE.value()`

7.2 ControlType

Description Correspond aux différents types de contrôles

Valeurs

- FOURDIRECTION : L'objet se déplace dans les quatres directions.
- CLASSICJUMP : L'objet se déplace latéralement et fait un saut simple

- DOUBLEJUMP : Comme le CLASSICJUMP mais avec un double saut
- CLICKFOLLOW : L'objet se déplace vers l'endroit du clic de la souris
- LEFTRIGHT : L'objet se déplace latéralement, sans saut
- UPDOWN : L'objet se déplace de bas en haut

Classe ControlComponent

7.3 Controls

Description Correspond aux contrôles utilisables dans PyEngine

Valeurs

- UPJUMP : Direction vers le haut (sert aussi au saut)
- LEFT : Direction vers la gauche
- RIGHT : Direction vers la droite
- DOWN : Direction vers le bas

Classe ControlComponent

7.4 MouseButton

Description Correspond aux différents boutons de la souris

Valeurs

- LEFTCLICK : Clic gauche
- MIDDLECLICK : Clic molette
- RIGHTCLICK : Clic droit

Classe ControlComponent

7.5 CollisionCauses

Description Correspond aux différentes causes d'une collision

Valeurs

- UNKNOWN : Cause inconnu
- GRAVITY : Causée par la gravité
- LEFTCONTROL : Causée par un déplacement à gauche du ControlComponent
- RIGHTCONTROL : Causée par un déplacement à droite du ControlComponent
- UPCONTROL : Causée par un déplacement en haut du ControlComponent
- DOWNCONTROL : Causée par un déplacement en bas du ControlComponent
- MOVECOMPONENT : Causée par un déplacement du MoveComponent

Classe PhysicComponent

Note : Si une collision a lieu à cause d'un saut, c'est la cause GRAVITY qui sera affichée.

7.6 WorldCallbacks

Description Correspond aux différents types de callback renvoyé par la state

Valeurs

— OUTFLOW : Activé quand un élément dépasse les bords de l'écran

Classe World

Note : Un callback est simplement une fonction lancée suivant des événements précis.

PyEngine incorpore beaucoup d'exceptions utilisées dans ses différentes classes. En voici la liste :

8.1 NoObjectError

Description L'objet n'existe pas.

8.2 CompatibilityError

Description L'objet est incompatible avec un autre objet.

Cette classe correspond à la fenêtre ouverte par votre jeu.

9.1 Constructeur

Description Crée l'objet Window

Paramètres

- width <integer> : Largeur de la fenêtre
- height <integer> : Hauteur de la fenêtre
- color <Color> (Color()) : Couleur de fond
- title <string> (« PyEngine ») : Titre de la fenêtre
- icon <string> (None) : Chemin vers l'icone de la fenêtre
- debug <boolean> (False) : Mode debug

Note : Si l'icone vaut « None » la fenêtre aura l'icone de PyGame

Voici ses attributs :

9.2 title

Description Titre de la fenêtre

Type string

9.3 color

Description Couleur de fond de la fenêtre

Type Color

9.4 size

Description Taille de la fenêtre

Type Tuple[int, int]

9.5 debug

Description Mode débog

Type boolean

9.6 world

Description Monde de la fenêtre

Type World

Voici ses méthodes :

9.7 stop

Description Arrête le jeu

Retourne Rien

Paramètre Rien

9.8 run

Description Lance le jeu

Retourne Rien

Paramètre Rien

Avertissement : Peut retourner les exceptions : NoObjectError
--

Cette classe correspond à un monde de votre jeu

10.1 Constructeur

Description Crée l'objet Monde

Paramètre window <Window> : Fenêtre liée au monde

Voici ses attributs :

10.2 window

Description Fenêtre liée au monde

Type Window

Voici ses méthodes :

10.3 set_callback

Description Définit un Callback

Retourne Rien

Paramètres

- callback <WorldCallbacks> : Callback à définir
- fonction <Function> : Fonction lancée au moment du callback

Les callbacks peuvent demander des paramètres. Il faut donc les fournir dans la fonction lancée.

OUTOFWINDOW

- <Entity> - Entité qui dépasse les bords
- <Vec2> - position de l'entité

Note : Un callback est simplement une fonction lancée suivant des évènements précis.

Avertissement : Peut retourner une exception : TypeError

10.4 get_system

Description Récupère un système du monde

Retourne <Systems|None> : Système du type <classe> s'il existe

Paramètre <Systems> : Classe du système à récupérer

Note : Systems fait référence aux systèmes (EntitySystem, UISystem, SoundSystem, CameraSystem et MusicSystem)

Cette classe correspond à une entité de votre jeu.

11.1 Constructeur

Description Crée l'objet Entity

Paramètre Rien

Voici ses attributs :

11.2 identity

Description Id de l'entité

Type int

Note : Le système définit l'id lui même. Elle ne doit pas être changée.

11.3 system

Description Système gérant l'entité

Type EntitySystem

Note : Le système se définit lui même. Il ne doit pas être changé.

Voici ses méthodes :

11.4 attach_entity

Description Attache une entité à l'entité

Retourne Rien

Paramètre entity <Entity> : Entité à attacher

11.5 add_component

Description Ajoute un composant à l'entité

Retourne <Components> : Composant ajouté

Paramètre component <Components> : Composant à ajouter

Note : Components fait référence à toutes les classes étant des composants.

Avertissement : Peut retourner une exception : TypeError

11.6 remove_component

Description Supprime un composant à l'entité

Retourne Rien

Paramètre component <Components> : Composant à ajouter

Note : Components fait référence à toutes les classes étant des composants.

Note : Supprimer un composant peut être dangereux. Faites le que si vous n'avez pas le choix.

11.7 has_component

Description Vérifie l'existence d'un composant dans l'entité

Retourne <bool> : Vrai si l'entité a le composant. Sinon Faux

Paramètre component <Components> : Composant à ajouter

Note : Components fait référence à toutes les classes étant des composants.

11.8 get_component

Description Récupère un composant de l'entité

Retourne <Components|None> : Composant dont le type est <component> s'il existe

Paramètre component <Components> : Composant à récupérer

Note : Components fait référence à toutes les classes étant des composants.

MusicSystem

Cette classe correspond au système de musique d'un monde

Avertissement : La gestion des .mp3 est un peu bugguée. Si votre mp3 ne fonctionne pas correctement, utilisez une autre extension (exemple : .wav)

Voici ses attributs :

12.1 loop

Description Vrai si la queue est en mode boucle

Type boolean

12.2 volume

Description Volume du système

Type int

Note : Le volume doit être entre 0 et 100 sinon une erreur ValueError est lancée

Voici ses méthodes :

12.3 next_song

Description Passe à la musique suivante

Retourne Rien
Paramètre Rien

12.4 clear_queue

Description Vide la queue
Retourne Rien
Paramètre Rien

12.5 play

Description Lance la musique
Retourne Rien
Paramètre Rien

Avertissement : Peut retourner une exception : NoObjectError

12.6 add

Description Ajoute une musique à la queue
Retourne Rien
Paramètre file <string> : Chemin vers le fichier de la musique

12.7 stop

Description Arrête la musique
Retourne Rien
Paramètre Rien

12.8 pause

Description Met en pause la musique
Retourne Rien
Paramètre Rien

12.9 unpauses

Description Relance la musique
Retourne Rien
Paramètre Rien

Cette classe correspond au système d'interface utilisateur du monde

Voici ses méthodes :

13.1 add_widget

Description Ajoute un widget au système

Retourne <Widgets> : Widget ajouté

Paramètre widget <Widgets> : Widget à ajouter

Note : Widgets fait référence à toutes les classes étant des widgets.

13.2 get_widget

Description Récupère un widget au système

Retourne <Widgets|None> : Widget s'il existe ou rien

Paramètre identity <int> : Id du widget à récupérer

13.3 has_widget

Description Savoir si un widget est enregistré

Retourne <bool> : Vrai si le Widget existe

Paramètre widget <Widgets> : Widget

Note : Widgets fait référence à toutes les classes étant des widgets.

13.4 remove_widget

Description Supprime l'enregistrement d'un widget

Retourne Rien

Paramètre widget <Widgets> : Widget à supprimer

Note : Widgets fait référence à toutes les classes étant des widgets.

Note : Peut retourner une ValueError

Cette classe correspond au système d'entité du monde

Voici ses méthodes :

14.1 get_entity

Description Récupère une entité

Retourne <Entity|None> : Entité si elle existe ou rien.

Paramètre id <int> : Id de l'entité à récupérer

14.2 add_entity

Description Ajoute une entité au système

Retourne <Entity> : Entité ajoutée

Paramètre entity <Entity> : Entité à ajouter

Avertissement : Peut retourner une exception : NoObjectError

14.3 has_entity

Description Savoir si une entité est enregistrée

Retourne <bool> : Vrai si l'entité existe

Paramètre entity <Entity> : Entité

14.4 remove_entity

Description Supprime l'enregistrement d'une entité

Retourne Rien

Paramètre entity <Entity> : Entité à supprimer

Note : Peut retourner une ValueError

Cette classe correspond au système de bruitage du monde

Voici ses attributs :

15.1 volume

Description Volume du système

Type int

Note : Le volume doit être entre 0 et 100 sinon une erreur ValueError est lancée

15.2 number_channel

Description Nombre de son jouable en simultané

Type int

Voici ses méthodes :

15.3 play

Description Joue un bruitage

Retourne Rien

Paramètre file <str> : Chemin vers le son

Cette classe correspond à la caméra du monde

Avertissement : Si vous utilisez la caméra, faites attention à la position de vos entités. PyGame n'incorpore pas de caméra donc pour chaque mouvement de caméra, PyEngine applique le mouvement inverse aux entités. Exemple : Si une entité est en 10, 10 et que l'on met la caméra en 10, 10, l'entité se trouvera en 0, 0.

Voici ses attributs :

16.1 position

Description Position de la caméra

Type Vec2

16.2 offset

Description Offset de la caméra

Type Vec2

16.3 zoom

Description Zoom de la caméra

Type int

16.4 entity_follow

Description Entité suivie par la caméra

Type Entity ou None

PositionComponent

Cette classe permet de définir une position à l'entité

17.1 Constructeur

Description Crée l'objet PositionComponent

Paramètres

- position <Vec2> : Position de l'entité
- offset <Vec2> (Vec2()) : Offset de l'entité

Note : L'offset n'est utile que dans le cas d'entité attachée à une autre entité.

Voici ses attributs :

17.2 position

Description Position de l'entité

Type Vec2

17.3 entity

Description Entité lié au composant

Type entity

Note : L'entité se définit elle même. Elle ne doit pas être modifiée

SpriteComponent

Cette classe permet de définir un sprite à l'entité

Avertissement : Non compatible avec le TextComponent

18.1 Constructeur

Description Crée l'objet SpriteComponent

Paramètres

- image <string> : Chemin vers le sprite
- scale <integer> (1) : « Zoom » sur le sprite
- rotation <integer> (0) : Rotation du sprite

Avertissement : Peut retourner une exception : CompatibilityError

Voici ses attributs :

18.2 scale

Description Scale du composant

Type int

18.3 size

Description Taille du sprite

Type Tuple[int, int] ou Vec2

Note : Size utilise un Vec2 pour se définir mais retourne un Tuple

18.4 rotation

Description Rotation du sprite

Type int

18.5 sprite

Description Chemin vers le sprite

Type string

18.6 entity

Description Entité lié au composant

Type entity

Note : L'entité se définit elle même. Elle ne doit pas être modifiée

Cette classe permet de définir un texte à l'entité

Avertissement : Non compatible avec le SpriteComponent

19.1 Constructeur

Description Crée l'objet TextComponent

Paramètres

- texte <string> : Texte à afficher
- color <Color> (Color()) : Couleur du texte
- font (Font()) : Police du texte
- background <Color|None> (None) : Couleur de fond
- scale <int> (1) : Scale du composant

Note : Si background est à None la couleur sera transparente.

Avertissement : Peut retourner une exception : CompatibilityError

Voici ses attributs :

19.2 scale

Description Scale du composant

Type int

19.3 background

Description Couleur de fond

Type None ou Color

19.4 color

Description Couleur du texte

Type Color

19.5 font

Description Police du texte

Type Font

19.6 text

Description Texte à afficher

Type string

19.7 rendered_size

Description Taille du texte rendu

Type Tuple[int, int]

19.8 entity

Description Entité lié au composant

Type entity

Note : L'entité se définit elle même. Elle ne doit pas être modifiée

Cette classe permet de définir une physique à l'entité

20.1 Constructeur

Description Crée l'objet PhysicsComponent

Paramètres

- affectbygravity <bool> (True) : Affecté par la gravité ou non
- gravity_force <int> (5) : Puissance de la gravité

Voici ses attributs :

20.2 gravity

Description Force de la gravité appliqué

Type int

20.3 callback

Description Fonction a appelé lors d'une collision

Type Function

Note : La fonction doit avec deux arguments : l'objet avec lequel il a eu collision et la cause

20.4 entity

Description Entité lié au composant

Type entity

Note : L'entité se définit elle même. Elle ne doit pas être modifiée

Cette classe permet de faire bouger automatiquement une entité

21.1 Constructeur

Description Crée l'objet MoveComponent

Paramètres

— direction <Vec2> : Direction du mouvement

Voici ses attributs :

21.2 direction

Description Vecteur direction du mouvement

Type Vec2

21.3 entity

Description Entité lié au composant

Type entity

Note : L'entité se définit elle même. Elle ne doit pas être modifiée

Cette classe permet de contrôler l'entité

22.1 Constructeur

Description Crée l'objet ControlComponent

Paramètres

- controltype <ControlType> : Type de contrôle
- speed <int> (5) : Vitesse du mouvement

Voici ses attributs :

22.2 speed

Description Vitesse du mouvement

Type int

22.3 entity

Description Entité lié au composant

Type entity

Note : L'entité se définit elle même. Elle ne doit pas être modifiée

Voici ses méthodes :

22.4 set_control

Description Définit un controle

Retourne Rien

Paramètres

- control <Controls> : Controle à définir
- key <const> : Touche correspondant au controle

22.5 get_control

Description Récupère un controle

Retourne <const> : Touche correspondant au controle

Paramètre control <Controls> : Controle à récupérer

Cette classe permet de définir une vie à l'entité.

23.1 Constructeur

Description Crée l'objet LifeComponent

Paramètre maxlife <int> (100) : Vie maximum

Voici ses attributs :

23.2 life

Description Vie actuelle

Type int

Note : life est compris entre 0 et maxlife. Si la valeur est inférieur à 0 alors life = 0. Si la valeur est supérieur à maxlife alors life = maxlife.

23.3 maxlife

Description Vie maximum

Type int

23.4 entity

Description Entité lié au composant

Type entity

Note : L'entité se définit elle même. Elle ne doit pas être modifiée

Cette classe sert de base à tous les autres widgets. Elle ne doit pas être utilisée en l'état.

Note : Le constructeur n'est pas listé vu qu'il ne doit pas être utilisé directement.

Voici ses attributs :

24.1 identity

Description Id du widget

Type int

Note : Le système définit l'id lui même. Elle ne doit pas être changée.

24.2 system

Description Système gérant le widget

Type UISystem

Note : Le système se définit lui même. Il ne doit pas être changé.

24.3 position

Description Position du widget

Type Vec2

Voici ses méthodes :

24.4 is_show

Description Permet de savoir si le widget est affiché

Retourne <boolean> : Vrai si le widget est affiché, sinon faux

Paramètre Rien

24.5 show

Description Affiche un widget

Retourne Rien

Paramètre Rien

24.6 hide

Description Cache un widget

Retourne Rien

Paramètre Rien

Cette classe permet d'afficher un texte.

Elle a les mêmes attributs et fonctions que Widget. Mais elle ajoute aussi les suivants.

25.1 Constructeur

Description Crée l'objet Label

Paramètres

- position <Vec2> : Position du widget
- texte <string> : Texte à afficher
- color <Color> (Color()) : Couleur du texte
- font (Font()) : Police du texte
- background <Color|None> (None) : Couleur de fond

Note : Si background est à None la couleur sera transparente.

Voici ses attributs :

25.2 background

Description Couleur de fond

Type None ou Color

25.3 color

Description Couleur du texte

Type Color

25.4 font

Description Police du texte

Type Font

25.5 text

Description Texte à afficher

Type string

Cette classe permet d'afficher une image.

Elle a les mêmes attributs et fonctions que Widget. Mais elle ajoute aussi les suivants.

26.1 Constructeur

Description Crée l'objet Image

Paramètres

- position <Vec2> : Position du widget
- sprite <string> : Chemin vers l'image
- size <None|Tuple[int, int]> (None) : Taille de l'image

Note : Si size vaut None, l'image ne sera pas redimensionnée.

Voici ses attributs :

26.2 size

Description Taille de l'image

Type Tuple[int, int]

26.3 sprite

Description Chemin vers l'image

Type string

Cette classe permet de d'afficher un bouton.

Elle a les mêmes attributs et fonctions que Widget. Mais elle ajoute aussi les suivants.

27.1 Constructeur

Description Crée l'objet Button

Paramètres

- position <Vec2> : Position du widget
- text <string> : Texte sur le bouton
- command <None|Function> (None) : Fonction lancé à l'appui du bouton
- size <None|Tuple[int, int]> ([100, 40]) : Taille du bouton
- image <None|string> (None) : Image du bouton

Note : Si l'image n'est pas spécifié, le bouton aura un rectangle gris comme fond.

Voici ses attributs :

27.2 sprite

Description Image du bouton

Type None ou string

27.3 size

Description Taille du bouton

Type Tuple[int, int]

27.4 command

Description Fonction lancé à l'appui du bouton

Type None ou Function

Cette classe permet de demander du texte à l'utilisateur.

Elle a les mêmes attributs et fonctions que Widget. Mais elle ajoute aussi les suivants.

28.1 Constructeur

Description Crée l'objet Entry

Paramètres

- position <Vec2> : Position du widget
- width <int> (200) : Largeur du widget
- image <None|string> (None) : Image du widget

Note : Si l'image vaut None, le widget utilisera deux rectangles colorés comme image

Voici ses attributs :

28.2 text

Description Text entré dans le widget

Type string

Cette classe permet de symboliser une couleur

29.1 Constructeur

Description Crée l'objet Color

Paramètres

- r <int> (255) : Rouge
- g <int> (255) : Vert
- b <int> (255) : Bleu

Voici ses méthodes :

29.2 get

Description Retourne la couleur en tuple

Retourne <Tuple[int, int, int]>

Paramètre Rien

29.3 set

Description Permet de définir une couleur

Retourne <Color>

Paramètre color <Color>

29.4 darker

Description Retourne un couleur plus foncé

Retourne <Color>

Paramètre Rien

29.5 lighter

Description Retourne une couleur plus clair

Retourne <Color>

Paramètre Rien

29.6 __add__

Description Correspond à l'addition

Retourne <Color>

Paramètre <Color>

29.7 __sub__

Description Correspond à la soustraction

Retourne <Color>

Paramètre <Color>

29.8 __repr__

Description Correspond à la représentation en string

Retourne <string>

Paramètre Rien

29.9 __eq__

Description Correspond à la comparaison “==”

Retourne <bool>

Paramètre <Color>

Cette classe permet de symboliser une police d'écriture

30.1 Constructeur

Description Crée l'objet Font

Paramètres

- name <string> (« Arial ») : Nom de la police
- size <int> (15) : Taille de la police
- bold <boolean> (False) : Vrai si la police est en gras
- italic <boolean> (False) : Vrai si la police est en italique

Voici ses attributs :

30.2 name

Description Nom de la police

Type string

30.3 size

Description Taille de la police

Type int

30.4 bold

Description Vrai si la police est en gras

Type boolean

30.5 italic

Description Vrai si la police est en italique

Type boolean

Voici ses méthodes :

30.6 `__repr__`

Description Correspond à la représentation en string

Retourne <string>

Paramètre Rien

30.7 `__eq__`

Description Correspond à la comparaison “==”

Retourne <bool>

Paramètre

Cette classe permet de symboliser un vecteur de dimension 2

31.1 Constructeur

Description Crée l'objet Vec2

Paramètres

- x <int> (0) : Coordonnée X
- y <int> (0) : Coordonnée Y

Voici ses attributs :

31.2 coords

Description Coordonnées du vecteur

Type Tuple[int, int]

31.3 length

Description Longueur du vecteur

Type int

Note : Length n'a pas de setter, vous ne pouvez pas la définir. Cependant, elle est modifiée par les coordonnées.

Voici ses méthodes :

31.4 `__add__`

Description Correspond à l'addition

Retourne `<Vec2>`

Paramètre `<Vec2int>`

31.5 `__sub__`

Description Correspond à la soustraction

Retourne `<Vec2>`

Paramètre `<Vec2int>`

31.6 `__mul__`

Description Correspond à la multiplication

Retourne `<Vec2>`

Paramètre `<Vec2int>`

31.7 `__repr__`

Description Correspond à la représentation en string

Retourne `<string>`

Paramètre Rien

31.8 `__eq__`

Description Correspond à la comparaison "=="

Retourne `<bool>`

Paramètre `<Vec2>`

31.9 `__neg__`

Description Correspond à la négation

Retourne `<Vec2>`

Paramètre Rien