
PyEngine

Version 1.0.0

août 24, 2019

1	Introduction	3
2	Téléchargement et Installation	5
3	FAQ	7
4	Changelog	9
5	Théorie	15
6	Hello World	19
7	Enumérations	23
8	Exceptions	27
9	World	29
10	Window	31
11	Entities	35
12	Systems	39
13	Components	45
14	Widgets	53
15	Network	61
16	Utils	65

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 1.5.0 - 28/07/19

- Entity-Tilemap : Move to « Entities » folder
- ControlComponent : Add MOUSEFOLLOW ControlType
- PhysicsComponent : Add MOUSEFOLLOW and MOUSECLICK CollisionCauses
- Client : Send type, author and message. Packet are used only in PyEngine
- Button : Add enabled property
- Entry : Add Ctrl-C and Ctrl-V
- Entry : Remove limit of caracters
- Entry : Repeats the characters if the key is kept pressed
- Console : Create console widget
- MultilineLabel : Create Label widget for multiline
- Label : Replace n to void. Label doesn't support break lines
- AnimatedImage : Create Image widget which support multiple images
- Selector : Create selector widget
- Others : Rename Maths to Others
- Others : Add wrap_text and get_images_from_gif functions
- Unit Tests : Fix Vec2 and Color tests
- Unit Tests : Add Console and MultilineLabel tests
- Optimization
- Bug Fix : ControlComponent make weird movements with MOUSECLICK ControlType
- Crash Fix : Crash when use show or hide
- Crash Fix : Crash when Client doesn't have callback
- Crash Fix : Crash when UISystem must show widgets without image or rect

4.2 V 1.4.2 - 17/07/19

- PhysicsComponent : Add callback parameters of constructor
- Tilemap : place tile at bottom
- Optimization
- Bug Fix : OUTFLOWWINDOW doesn't work with CameraSystem
- Bug Fix : Remove offset when change position
- Bug Fix : Hide and Show doesn't spread to child in Widgets
- Bug Fix : Focus stays when widget is hide

Cette MAJ ne casse pas la compatibilité avec la précédente.

4.3 V 1.4.1 - 14/07/19

- Color : Add parameter to apply darker or lighter x times
- Optimization
- Bug Fix : Fix Diagonal movement from MoveComponent
- Bug Fix : EntitySystem give wrong id to tiles of Tilemap
- Bug Fix : Sprite is rescale at every change of sprite in SpriteComponent
- Bug Fix : If tileset isn't in the same director than tilemap, sprites isn't found
- Bug Fix : ControlComponent save the keypressed when change world

Cette MAJ ne casse pas la compatibilité avec la précédente.

4.4 V 1.4.0 : All Update - 13/07/19

- AnimComponent : Add play attribute
- AnimComponent : Use clamp function to set timer
- PositionComponent : Offset is now a property
- PositionComponent : position return the position without offset
- SpriteComponent : Scale don't modify width and height
- LifeComponent : Add callback which is trigger when entity has 0 pv
- Network : If packet have « TOALL » as type, author will receive the packet
- Tilemap : Create basic tilemap (using Tiled)
- Button-Image : size is now a Vec2
- Button : Only Left Button of Mouse trigger Button
- Button : Remove btn argument on command of Button
- Checkbox : Create checkbox widget
- ProgressBar : Create progressbar widget
- Entry : Add accents letters in accepted and remove some weird symbol
- Color : Use clamp function
- Maths : Clamp function can be use without max or/and min value
- Font : Add rendered_size function
- Vec2 : Add divide operator
- Vec2 : Add iterator
- Vec2 : Change representation in string to « Vec2(X, Y) »
- Unit Tests : Add Tilemap, Prefabs and Network tests
- Unit Tests : Update Components and Widgets tests
- Setup : Add numpy as dependances
- README : Add version of pygame
- README : Remove useless section

- Optimization
- Bug Fix : Button must be focused to be hover
- Bug Fix : Rescale SpriteComponent can make weird result
- Bug Fix : Shift-capitals must be typed twice in Entry to be writed
- Bug Fix : Text can be more longer than the Entry
- Bug Fix : Press an other key than the actual break the movement with ControlComponent
- Crash Fix : Crash sometimes to create hover button with sprite
- Crash Fix : Crash when import Vec2 at first

4.5 V 1.3.0 : Utils Update - 07/07/19

- Window : Add is_running and update_rate property
- Window : In debug, show fps in console (must be around 60)
- Window-World : Rename and move WorldCallbacks to WindowCallbacks
- WindowCallbacks : Add CHANGEWORLD and STOPWINDOW
- Entity : Can't have the same type of component two times
- EntitySystem-UISystem : Change debug color to blue for ui and red for entity
- EntitySystem : Add get_all_entities function
- PhysicsComponent : Create CollisionInfos
- MoveComponent : Unlock diagonal movement
- LifeComponent : Use clamp function to set life
- AnimComponent : Create basic animator system
- Entry : Add width and sprite property
- Entry : Add color and front parameters
- Entry : Define accepted letters
- Network : Create basic network system
- Vec2 : Add normalized function
- Color : Add to_hex and from_hex function
- Colors : Add new colors
- loggers : Create logging System
- Lang : Create translate system
- Config : Create Config system
- Maths : Create usefull functions (clamp)
- Unit Tests : Add AnimComponent, loggers, config and lang tests
- Unit Tests : Update Window, Color, Entry and World tests
- Optimization and fix some little errors
- Bug Fix : Entry is update without focus
- Crash Fix : Crash when show id of Entity Texts
- Crash Fix : Crash when use entity_follow of CameraSystem

4.6 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

4.7 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 combatibilité avec la précédente.

4.8 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 combatibilité avec la précédente.

4.9 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

4.10 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.

4.11 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.

4.12 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)

4.13 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)

4.14 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.

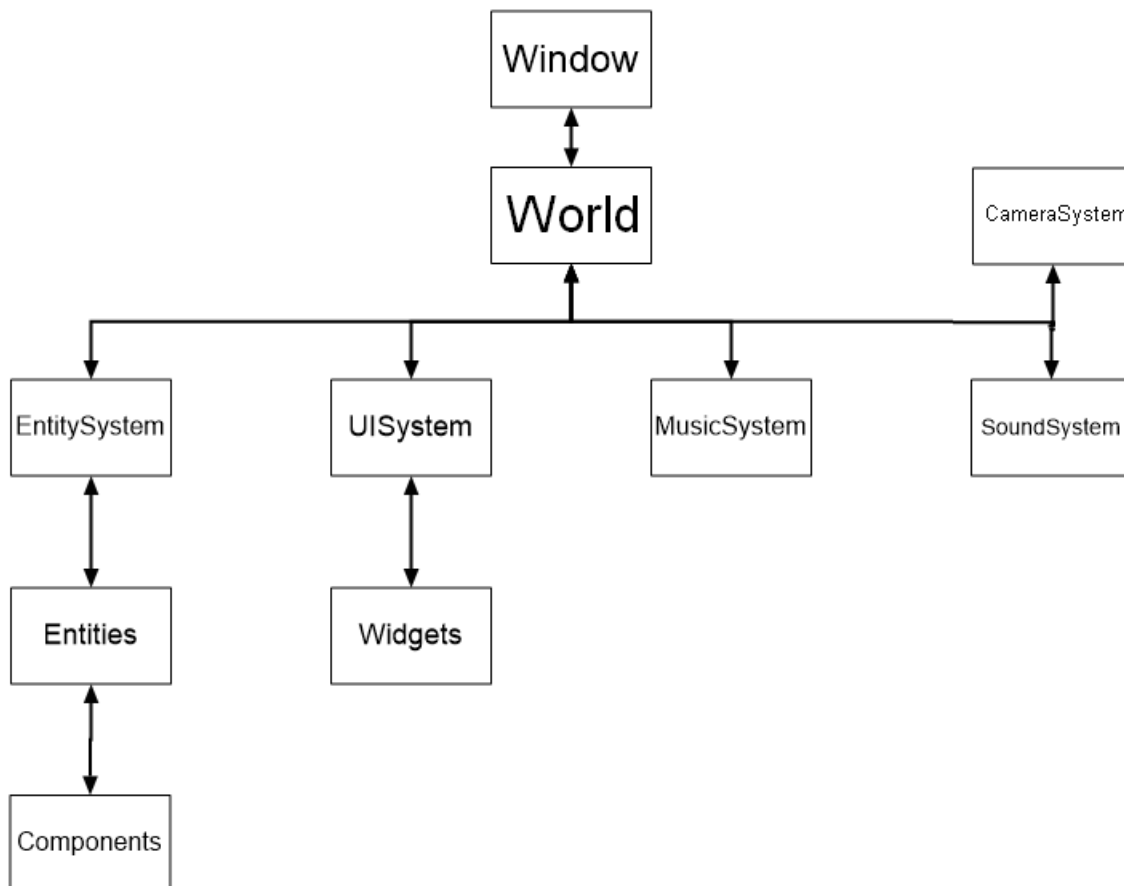
5.4 Les éléments finaux

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.

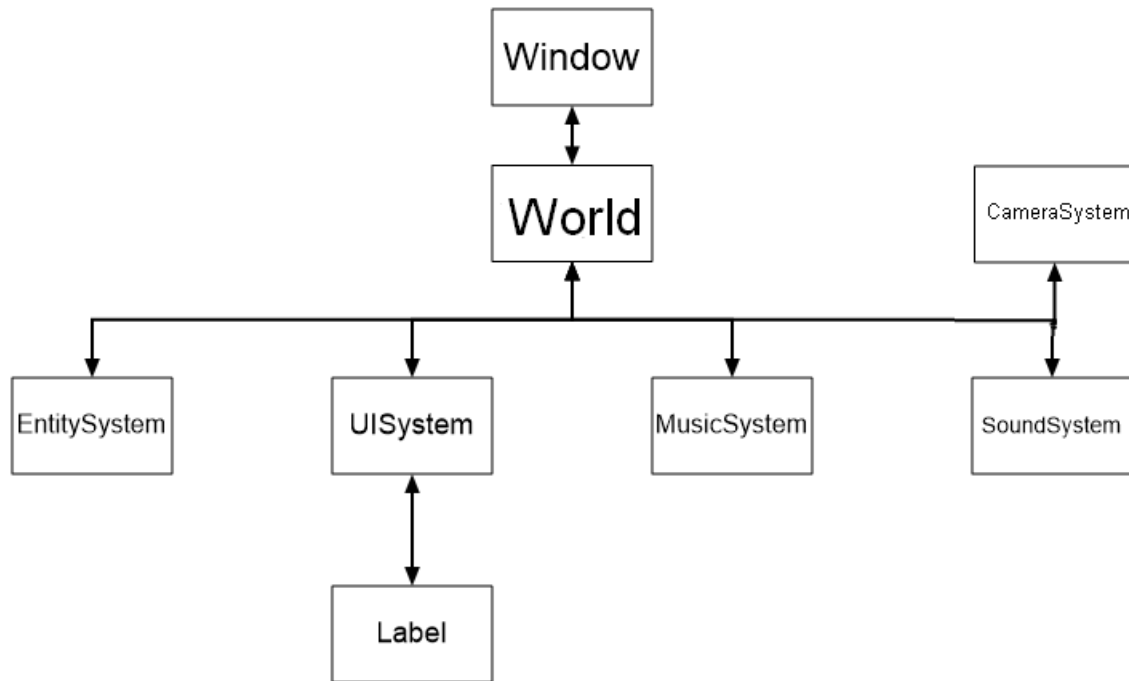
Du coup des widgets, ce sont bien les éléments finaux.

5.5 Représentation de l'architecture

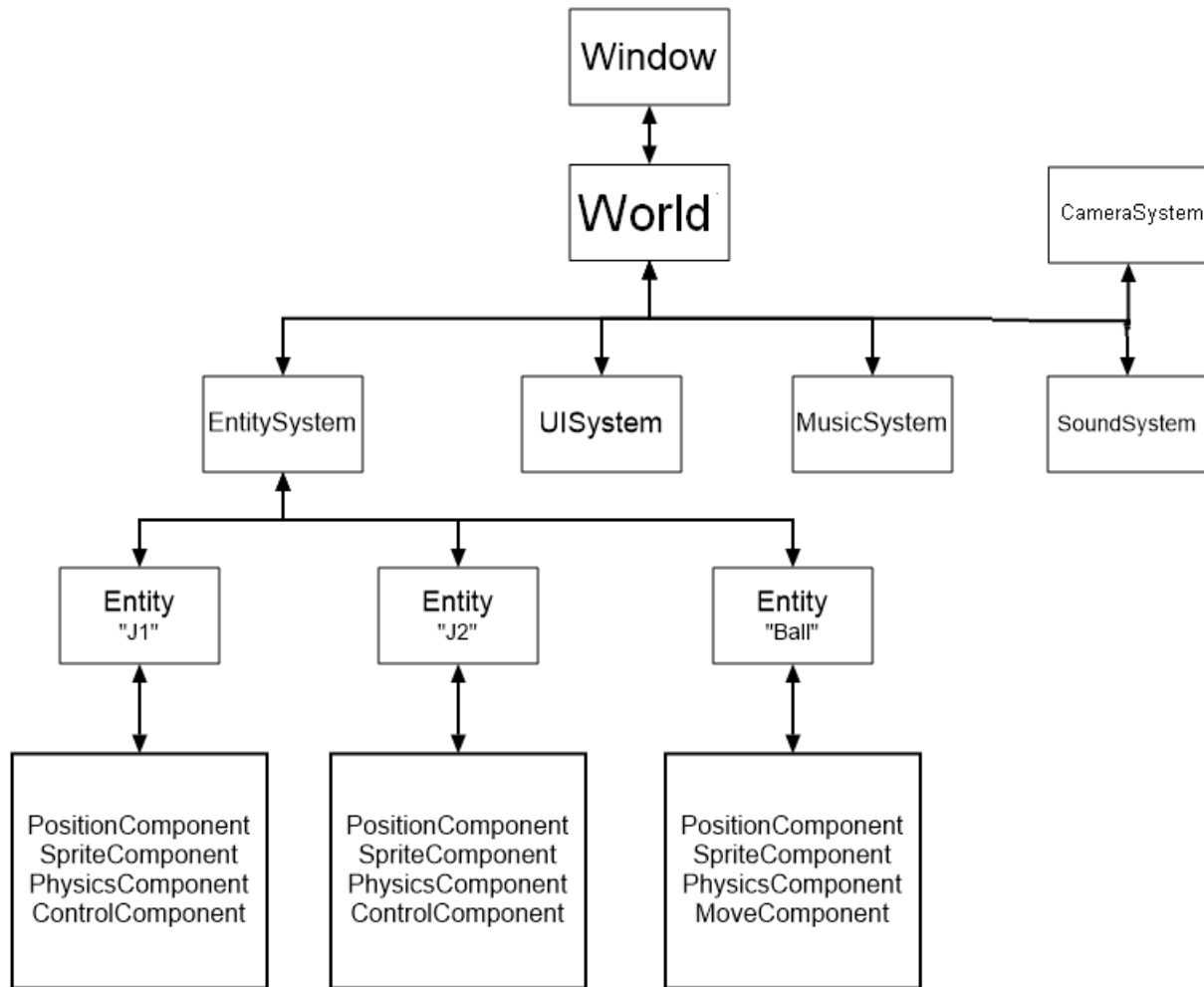
Architecture général :



Architecture du tutoriel HelloWorld :



Architecture de l'exemple Pong :



CHAPITRE 6

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 :

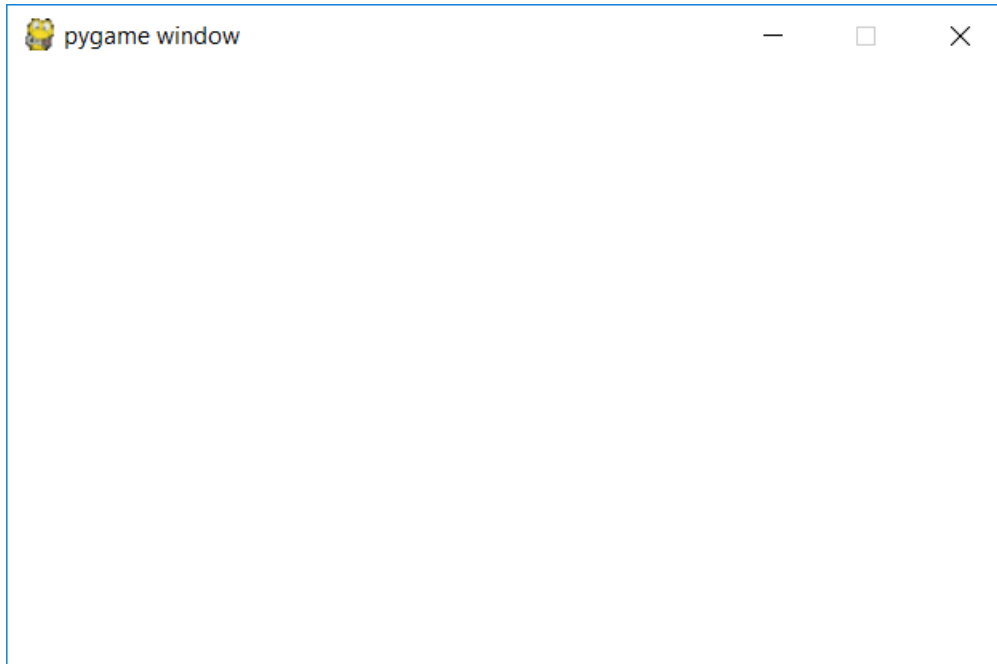
```
fenetre = Window(500, 300, Colors.WHITE.value)  
# 500 : Largeur  
# 300 : Longueur  
# Colors.WHITE.value : Couleur blanche
```

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.Utils 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.utils import Colors, Vec2

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

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

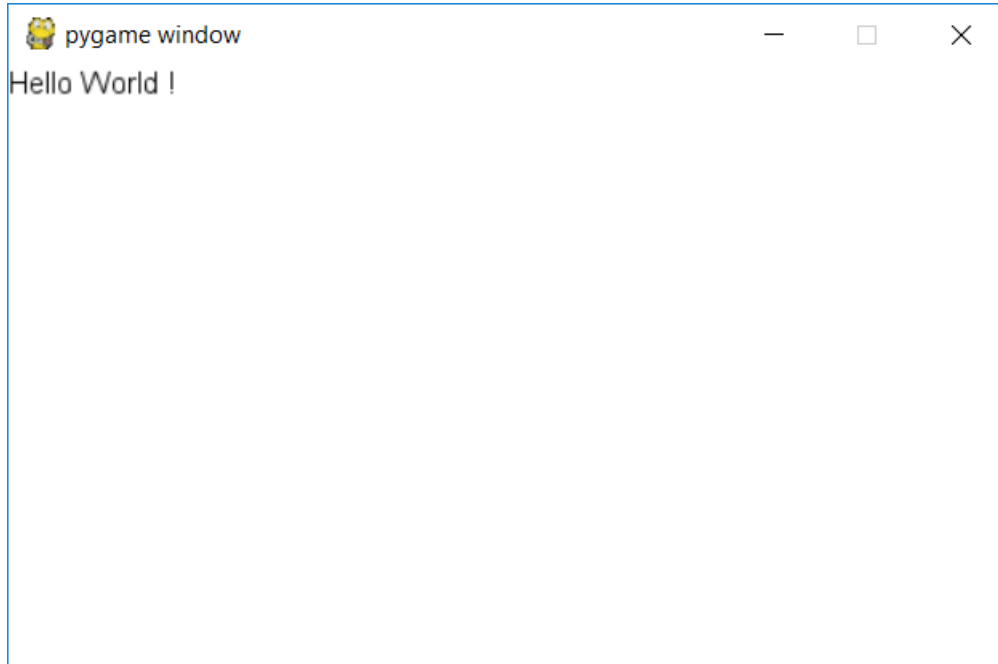
(suite sur la page suivante)

(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
- FUCHSIA : Couleur fuchsia
- GRAY : Couleur grise
- LIME : Couleur vert clair
- MAROON : Couleur marron
- NAVYBLUE : Couleur bleu clair
- OLIVE : Couleur olive
- PURPLE : Couleur violette
- SILVER : Couleur argent
- TEAL : Couleur teal
- YELLOW : Couleur jaune
- ORANGE : Couleur orange
- CYAN : Couleur cyan

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 à lieu à cause d'un saut, c'est la cause GRAVITY qui sera affichée.

7.6 WindowCallbacks

Description Correspond aux différents types de callback renvoyé par la state

Valeurs

- OUTOFWINDOW : Activé quand un élément dépasse les bords de l'écran
- CHANGeworld : Activé quand on change de monde
- STOPWINDOW : Activé à l'arrêt de la fenêtre

Classe Window

Note : Un callback est simplement une fonction lancée suivant des évènements 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 un monde de votre jeu

9.1 Constructeur

Description Crée l'objet Monde

Paramètre window <Window> : Fenêtre liée au monde

Voici ses attributs :

9.2 window

Description Fenêtre liée au monde

Type Window

Voici ses méthodes :

9.3 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 la fenêtre ouverte par votre jeu.

10.1 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'icon de la fenêtre
- debug <boolean> (False) : Mode debug

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

Voici ses attributs :

10.2 title

Description Titre de la fenêtre

Type string

10.3 color

Description Couleur de fond de la fenêtre

Type Color

10.4 update_rate

Description Nombre de frames par secondes théorique

Type int

10.5 size

Description Taille de la fenêtre

Type Tuple[int, int]

10.6 debug

Description Mode debug

Type boolean

10.7 world

Description Monde de la fenêtre

Type World

Voici ses méthodes :

10.8 stop

Description Arrête le jeu

Retourne Rien

Paramètre Rien

10.9 run

Description Lance le jeu

Retourne Rien

Paramètre Rien

10.10 set_callback

Description Définit un Callback

Retourne Rien

Paramètres

- callback <WindowCallbacks> : Callback à définir
- function <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

STOPWINDOW Rien**CHANGEWORLD**

- <World> - Ancien Monde
- <World> - Nouveau Monde

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

Avertissement : Peut retourner une exception : `TypeError`

Avertissement : Peut retourner les exceptions : `NoObjectError`

Vous trouverez dans cette catégorie toutes les classes et fonctions relatant aux entités.

11.1 Entity

Cette classe correspond à une entité de votre jeu.

11.1.1 Constructeur

Description Crée l'objet Entity

Paramètre Rien

Voici ses attributs :

11.1.2 identity

Description Id de l'entité

Type int

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

11.1.3 system

Description Système gérant l'entité

Type EntitySystem

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

Voici ses méthodes :

11.1.4 attach_entity

Description Attache une entité à l'entité

Retourne Rien

Paramètre entity <Entity> : Entité à attacher

11.1.5 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

11.1.6 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.

11.1.7 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.

11.1.8 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.

11.2 Tilemap

Cette classe correspond à une tilemap de votre jeu.

Note : Cette classe utilise le fichier .json donné par Tiled

Note : Par défaut, les tiles ont une physique sans affectation gravitationnelle

11.2.1 Constructeur

Description Crée l'objet Tilemap

Paramètres

- position <Vec2> : Position de la tilemap
- file <str> : Chemin vers le fichier .json
- Scale <int> (1) : Scale de la tilemap

Voici ses attributs :

11.2.2 identity

Description Id de l'entité

Type int

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

11.2.3 scale

Description Scale de la tilemap

Type int

11.2.4 tiles

Description Liste des tiles formant la tilemap

Type List[Entity]

11.2.5 system

Description Système gérant l'entité

Type EntitySystem

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

Vous trouverez dans cette catégorie toutes les classes et fonctions relatant aux systèmes de monde.

12.1 EntitySystem

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 :

12.1.1 position

Description Position de la caméra

Type Vec2

12.1.2 offset

Description Offset de la caméra

Type Vec2

12.1.3 zoom

Description Zoom de la caméra

Type int

12.1.4 entity_follow

Description Entité suivie par la caméra

Type Entity ou None

12.2 EntitySystem

Cette classe correspond au système d'entité du monde

Voici ses méthodes :

12.2.1 get_entity

Description Récupère une entité

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

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

12.2.2 get_all_entities

Description Récupère toutes les entités

Retourne <Tuple[Entity]> : Liste des entités

Paramètres Rien

12.2.3 add_entity

Description Ajoute une entité au système

Retourne <Entity> : Entité ajoutée

Paramètre entity <Entity> : Entité à ajouter

Avertissement : Peut retourner une exception : NoObjectError

12.2.4 has_entity

Description Savoir si une entité est enregistrée

Retourne <bool> : Vrai si l'entité existe

Paramètre entity <Entity> : Entité

12.2.5 remove_entity

Description Supprime l'enregistrement d'une entité

Retourne Rien

Paramètre entity <Entity> : Entité à supprimer

Note : Peut retourner une ValueError

12.3 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 :

12.3.1 loop

Description Vrai si la queue est en mode boucle

Type boolean

12.3.2 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 :

12.3.3 next_song

Description Passe à la musique suivante

Retourne Rien

Paramètre Rien

12.3.4 clear_queue

Description Vide la queue

Retourne Rien

Paramètre Rien

12.3.5 play

Description Lance la musique

Retourne Rien

Paramètre Rien

Avertissement : Peut retourner une exception : NoObjectError

12.3.6 add

Description Ajoute une musique à la queue

Retourne Rien

Paramètre file <string> : Chemin vers le fichier de la musique

12.3.7 stop

Description Arrête la musique

Retourne Rien

Paramètre Rien

12.3.8 pause

Description Met en pause la musique

Retourne Rien

Paramètre Rien

12.3.9 unpaused

Description Relance la musique

Retourne Rien

Paramètre Rien

12.4 SoundSystem

Cette classe correspond au système de bruitage du monde

Voici ses attributs :

12.4.1 volume

Description Volume du système

Type int

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

12.4.2 number_channel

Description Nombre de son jouable en simultané

Type int

Voici ses méthodes :

12.4.3 play

Description Joue un bruitage

Retourne Rien

Paramètre file <str> : Chemin vers le son

12.5 UISystem

Cette classe correspond au système d'interface utilisateur du monde

Voici ses méthodes :

12.5.1 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.

12.5.2 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

12.5.3 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.

12.5.4 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

Vous trouverez dans cette catégorie toutes les classes et fonctions relatant aux composants d'entité.

13.1 AnimComponent

Cette classe permet de d'animer le sprite de l'entité

13.1.1 Constructeur

Description Crée l'objet AnimComponent

Paramètres

- timer <int> : Temps en frames pour changer de sprites
- images <Tuple[string]> : Liste des chemins des sprites

Note : Il est nécessaire d'avoir un SpriteComponent pour ajouter un AnimComponent

Voici ses attributs :

13.1.2 time

Description Temps en frames pour changer de sprites

Type int

13.1.3 images

Description Liste des chemins des sprites

Type Tuple[string]

13.1.4 entity

Description Entité lié au composant

Type entity

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

13.2 ControlComponent

Cette classe permet de contrôler l'entité

13.2.1 Constructeur

Description Crée l'objet ControlComponent

Paramètres

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

Voici ses attributs :

13.2.2 speed

Description Vitesse du mouvement

Type int

13.2.3 entity

Description Entité lié au composant

Type entity

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

Voici ses méthodes :

13.2.4 set_control

Description Définit un controle

Retourne Rien

Paramètres

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

13.2.5 get_control

Description Récupère un controle

Retourne <const> : Touche correspondant au controle

Paramètre control <Controls> : Controle à récupérer

13.3 LifeComponent

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

13.3.1 Constructeur

Description Crée l'objet LifeComponent

Paramètres

— maxlife <int> (100) : Vie maximum

— callback <function> (None) : Fonction s'activant quand la vie arrive à 0

Voici ses attributs :

13.3.2 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.

13.3.3 maxlife

Description Vie maximum

Type int

13.3.4 callback

Description Fonction s'activant quand la vie arrive à 0

Type function

13.3.5 entity

Description Entité lié au composant

Type entity

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

13.4 MoveComponent

Cette classe permet de faire bouger automatiquement une entité

13.4.1 Constructeur

Description Crée l'objet MoveComponent

Paramètres

— direction <Vec2> : Direction du mouvement

Voici ses attributs :

13.4.2 direction

Description Vecteur direction du mouvement

Type Vec2

13.4.3 entity

Description Entité lié au composant

Type entity

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

13.5 PhysicsComponent

Cette classe permet de définir une physique à l'entité

13.5.1 Constructeur

Description Crée l'objet PhysicsComponent

Paramètres

- affectbygravity <bool> (True) : Affecté par la gravité ou non
- gravity_force <int> (5) : Puissance de la gravité
- callback <function> (None) : Fonction appelé lors de la collision

Voici ses attributs :

13.5.2 gravity

Description Force de la gravité appliqué

Type int

13.5.3 callback

Description Fonction appelée lors d'une collision

Type Function

Note : La fonction doit avoir deux arguments : l'objet avec lequel il a eu collision et les informations de la collision

Note : Les informations de la cause sont représentées par une classe ayant comme attributs : - La cause de la collision (cause) - Le côté de la collision (cote)

13.5.4 entity

Description Entité liée au composant

Type entity

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

13.6 PositionComponent

Cette classe permet de définir une position à l'entité

13.6.1 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 :

13.6.2 position

Description Position de l'entité

Type Vec2

13.6.3 entity

Description Entité lié au composant

Type entity

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

13.7 SpriteComponent

Cette classe permet de définir un sprite à l'entité

Avertissement : Non compatible avec le TextComponent

13.7.1 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 :

13.7.2 scale

Description Scale du composant

Type int

13.7.3 size

Description Taille du sprite

Type Tuple[int, int] ou Vec2

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

13.7.4 rotation

Description Rotation du sprite

Type int

13.7.5 sprite

Description Chemin vers le sprite

Type string

13.7.6 entity

Description Entité lié au composant

Type entity

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

13.8 TextComponent

Cette classe permet de définir un texte à l'entité

Avertissement : Non compatible avec le SpriteComponent

13.8.1 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 :

13.8.2 scale

Description Scale du composant

Type int

13.8.3 background

Description Couleur de fond

Type None ou Color

13.8.4 color

Description Couleur du texte

Type Color

13.8.5 font

Description Police du texte

Type Font

13.8.6 text

Description Texte à afficher

Type string

13.8.7 rendered_size

Description Taille du texte rendu

Type Tuple[int, int]

Note : La taille du texte rendu ne peut et ne doit pas être modifiée

13.8.8 entity

Description Entité lié au composant

Type entity

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

Vous trouverez dans cette catégorie toutes les classes et fonctions relatant aux widgets.

14.1 Entry

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.

14.1.1 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
- color <None|Color> (Colors.BLACK.value) : Couleur du texte
- font <None|Font> (Font()) : Police du texte

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

Voici ses attributs :

14.1.2 text

Description Text entré dans le widget

Type string

14.1.3 sprite

Description Image de l'Entry

Type None ou string

14.1.4 width

Description Largeur de l'Entry

Type int

14.2 Button

Cette classe permet de d'afficher un bouton.

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

14.2.1 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 :

14.2.2 sprite

Description Image du bouton

Type None ou string

14.2.3 size

Description Taille du bouton

Type Tuple[int, int]

14.2.4 command

Description Fonction lancé à l'appui du bouton

Type None ou Function

14.3 Label

Cette classe permet d’afficher un texte.

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

Avertissement : Label ne supporte pas les sauts de ligne. Utilisez MultilinesLabel

14.3.1 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 :

14.3.2 background

Description Couleur de fond

Type None ou Color

14.3.3 color

Description Couleur du texte

Type Color

14.3.4 font

Description Police du texte

Type Font

14.3.5 text

Description Texte à afficher

Type string

14.4 Image

Cette classe permet d'afficher une image.

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

14.4.1 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 :

14.4.2 size

Description Taille de l'image

Type Tuple[int, int]

14.4.3 sprite

Description Chemin vers l'image

Type string

14.5 Widget

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 :

14.5.1 identity

Description Id du widget

Type int

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

14.5.2 system

Description Système gérant le widget

Type UISystem

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

14.5.3 position

Description Position du widget

Type Vec2

Voici ses méthodes :

14.5.4 is_show

Description Permet de savoir si le widget est affiché

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

Paramètre Rien

14.5.5 show

Description Affiche un widget

Retourne Rien

Paramètre Rien

14.5.6 hide

Description Cache un widget

Retourne Rien

Paramètre Rien

14.6 Console

Cette classe permet d'afficher un console.

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

14.6.1 Constructeur

Description Crée l'objet Console

Paramètres

- window <Window> : Fenetre du widget
- position <Vec2> : Position du widget
- width <int> : Largeur de la console

Note : La console a de base deux commandes : - print : permettant d'afficher quelque chose dans le retour - debug : permettant d'activer ou désactiver le debug de la fenêtre

Voici ses méthodes :

14.6.2 reply

Description Affiche du texte dans le retour du widget

Retourne Rien

Paramètre texte <str> : Texte à afficher

14.6.3 add_command

Description Ajoute une commande à la console

Retourne Rien

Paramètres

- name <str> : Nom de la commande
- function <function> : Fonction de la commande

Note : Function doit avoir comme arguments : - console <Console> : La console exécutant la commande - window <Window> : La fenêtre de jeu - args <List[str]> : Liste des arguments données

14.6.4 delete_command

Description Supprime une commande

Retourne Rien

Paramètre name <str> : Nom de la commande

14.7 AnimatedImage

Cette classe permet d'afficher une image avec une animation.

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

14.7.1 Constructeur

Description Crée l'objet AnimatedImage

Paramètres

- position <Vec2> : Position du widget
- sprites <List[str]> : Chemin vers les images
- timer <int> : Timer pour l'animation
- size <None|Tuple[int, int]> (None) : Taille de l'image

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

Voici ses attributs :

14.7.2 size

Description Taille de l'image

Type Tuple[int, int]

14.7.3 timer

Description Timer pour l'animation

Type int

14.7.4 sprite

Description Chemin vers l'image actuelle

Type str

14.7.5 sprites

Description Chemin vers les images

Type List[str]

14.8 Selector

Cette classe permet d'afficher un sélecteur de texte.

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

14.8.1 Constructeur

Description Crée l'objet Selector

Paramètres

- position <Vec2> : Position du widget
- textes <List[str]> : Textes à sélectionner

Voici son attribut :

14.8.2 strings

Description Texte à sélectionner

Type List[str]

Voici sa méthode :

14.8.3 get

Description Récupère le texte sélectionné

Retourne str : Texte sélectionné

Paramètres Rien

Vous trouverez dans cette catégorie toutes les classes et fonctions relatant aux network.

15.1 NetworkManager

Cette classe permet de gérer le network de PyEngine

15.1.1 Constructeur

Description Crée l'objet NetworkManager

Paramètres Aucun

Voici ses attributs :

15.1.2 client

Description Client du jeu

Type Client

Note : Pour le définir, il faut utiliser la méthode create_client

15.1.3 server

Description Serveur du jeu

Type Server

Note : Pour le définir, il faut utiliser la méthode `create_server`

Voici ses méthodes :

15.1.4 `create_client`

Description Créer le client du jeu

Retourne Rien

Paramètres

- `ip <string>` : IP du Serveur où se connecter
- `port <string>` : Port du Serveur où se connecter
- `callback <function>` : Fonction appelé lors de la reception d'un packet

Note : La fonction `callback` doit avoir comme arguments : - Le type du packet - L'id de l'auteur du packet - Le message du packet

15.1.5 `create_server`

Description Créer le serveur du jeu

Retourne Rien

Paramètres

- `port <string>` : Port du Serveur

15.1.6 `stop_client`

Description Arrête le client du jeu

Retourne Rien

Paramètres Rien

15.1.7 `stop_server`

Description Arrête le serveur du jeu

Retourne Rien

Paramètres Rien

15.2 Packet

Cette classe représente un message transitant sur le network

15.2.1 Constructeur

Description Crée l'objet Packet

Paramètres

- type <str> (None) : Type du packet
- author <int> (None) : Id de l'auteur du packet
- message <str> (None) : Message du Packet

Note : Cette classe ne doit être utilisée que pour envoyer des messages au serveur ou aux clients.

15.3 Client

Cette classe représente le client de PyEngine

Avertissement : Cette classe doit être construite via le NetworkManager.

Voici ses méthodes :

15.3.1 stop

Description Arrête le client

Retourne Rien

Paramètres Rien

Avertissement : Il faut privilégier la fonction stop_client du NetworkManager.

15.3.2 send

Description Envoie un packet au serveur

Retourne Rien

Paramètres

- type <str> : Type du packet
- author <int> : Id de l'auteur du packet
- message <str> : Message du packet

Note : Sans modification, le serveur se contentera de renvoyer le packet à tous les autres clients

15.4 Server

Cette classe représente le serveur de PyEngine

Avertissement : Cette classe doit être construite via le NetworkManager.

Voici ses méthodes :

15.4.1 stop

Description Arrête le serveur

Retourne Rien

Paramètres Rien

Avertissement : Il faut privilégier la fonction stop_server du NetworkManager.

15.4.2 sendto

Description Envois un message à un client

Retourne Rien

Paramètres

- nb <int> : ID du client à qui envoyer le message
- packet <Packet> : Packet représentant le message

15.4.3 sendall

Description Envois un message à tous les clients sauf à l’auteur

Retourne Rien

Paramètres

- packet <Packet> : Packet représentant le message

Vous trouverez dans cette catégorie toutes les classes et fonctions relatant aux utilitaires.

16.1 Color

Cette classe permet de symboliser une couleur

16.1.1 Constructeur

Description Crée l'objet Color

Paramètres

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

Voici ses méthodes :

16.1.2 get

Description Retourne la couleur en tuple

Retourne <Tuple[int, int, int]>

Paramètre Rien

16.1.3 set

Description Permet de définir une couleur

Retourne <Color>

Paramètre color <Color>

16.1.4 to_hex

Description Retourne la couleur en hexadécimal

Retourne <string>

Paramètre Rien

16.1.5 from_hex

Description Définit la couleur à partir de l'hexadécimal

Retourne Rien

Paramètre <string>

16.1.6 darker

Description Retourne un couleur plus foncé

Retourne <Color>

Paramètre Rien

16.1.7 lighter

Description Retourne une couleur plus clair

Retourne <Color>

Paramètre Rien

16.1.8 __add__

Description Correspond à l'addition

Retourne <Color>

Paramètre <Color>

16.1.9 __sub__

Description Correspond à la soustraction

Retourne <Color>

Paramètre <Color>

16.1.10 __repr__

Description Correspond à la représentation en string

Retourne <string>

Paramètre Rien

16.1.11 __eq__

Description Correspond à la comparaison “==”

Retourne <bool>

Paramètre <Color>

16.2 Config

Cette classe permet de symboliser un fichier de config

16.2.1 Constructeur

Description Crée l’objet Config

Paramètre file <string> : Chemin vers le fichier config

Note : Si le fichier n’existe pas, il faudra utiliser la méthode create pour le créer

Voici son attribut :

16.2.2 file

Description Chemin du fichier config

Type string

Voici ses méthodes :

16.2.3 get

Description Retourne la valeur de la config

Retourne <Any> : Valeur de la config

Paramètre key <string> : Clé de la valeur

16.2.4 set

Description Définit une valeur de la config

Retourne Rien

Paramètres

- key <string> : Clé de la valeur
- value <Any> : Valeur

16.2.5 save

Description Sauvegarde la config

Retourne Rien

Paramètres Rien

16.2.6 create

Description Crée la config

Retourne Rien

Paramètre dic <Dict> : Dictionnaire représentant la config

16.3 Font

Cette classe permet de symboliser une police d'écriture

16.3.1 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 :

16.3.2 name

Description Nom de la police

Type string

16.3.3 size

Description Taille de la police

Type int

16.3.4 bold

Description Vrai si la police est en gras

Type boolean

16.3.5 italic

Description Vrai si la police est en italique

Type boolean

Voici ses méthodes :

16.3.6 `renderer_size`

Description Correspond à la taille du rendu d'un texte

Retourne `Tuple[int, int]`

Paramètre `text <string>` : Texte à rendre

16.3.7 `__repr__`

Description Correspond à la représentation en string

Retourne `<string>`

Paramètre Rien

16.3.8 `__eq__`

Description Correspond à la comparaison "=="

Retourne `<bool>`

Paramètre ``

16.4 Lang

Cette classe permet de symboliser un fichier de langage

16.4.1 Constructeur

Description Crée l'objet Lang

Paramètre `file <string>` : Chemin vers le fichier config

Avertissement : Si le fichier n'existe pas, il retourne une `ValueError`

Voici son attribut :

16.4.2 `file`

Description Chemin du fichier lang

Type `string`

Voici ses méthodes :

16.4.3 get_translate

Description Retourne la traduction à partir de la clé

Retourne <string> : Traduction

Paramètres

- key <string> : Clé de la traduction
- default <string> : Traduction par défaut

Note : Cette fonction retournera la traduction par défaut si la clé n'existe pas dans le fichier.

16.5 Loggers

Cette classe permet de symboliser les loggers du jeu

Avertissement : Cette classe ne doit pas être construite. Une variable nommé « loggers » est déjà disponible.

Voici ses méthodes :

16.5.1 create_logger

Description Créer un logger

Retourne <Logger> : Logger créé

Paramètres

- name <string> : Nom du logger à créer
- file <string> (**None**) [Chemin du fichier si le] logger doit enregistrer les logs
- stream <bool> (**False**) [Vrai si le logger doit] afficher les logs dans la console

16.5.2 get_logger

Description Récupère un logger

Retourne <Logger> : Logger

Paramètre name <string> : Nom du Logger

Avertissement : Si le logger n'existe pas, il retourne une KeyError

16.5.3 get_all

Description Récupère tous les loggers

Retourne <Tuple[Logger]> : Tous les loggers

Paramètres Rien

16.5.4 to_all

Description Envoie un message à tous les loggers

Retourne Rien

Paramètres

- action <string> : Urgence du message
- message <string> : Message à envoyer

Note : Les actions sont « debug », « info », « critical », « warning » et « error »

16.6 Vec2

Cette classe permet de symboliser un vecteur de dimension 2

16.6.1 Constructeur

Description Crée l'objet Vec2

Paramètres

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

Voici ses attributs :

16.6.2 coords

Description Coordonnées du vecteur

Type Tuple[int, int]

16.6.3 fcoords

Description Coordonnées du vecteur

Type Tuple[float, float]

Voici ses méthodes :

16.6.4 length

Description Longueur du vecteur

Type int

16.6.5 normalized

Description Retourne le vecteur normalisé

Retourne <Vec2>

Paramètres Rien

16.6.6 `__add__`

Description Correspond à l'addition

Retourne `<Vec2>`

Paramètre `<Vec2int>`

16.6.7 `__sub__`

Description Correspond à la soustraction

Retourne `<Vec2>`

Paramètre `<Vec2int>`

16.6.8 `__mul__`

Description Correspond à la multiplication

Retourne `<Vec2>`

Paramètre `<Vec2int>`

16.6.9 `__truediv__`

Description Correspond à la division

Retourne `<Vec2>`

Paramètre `<Vec2int>`

16.6.10 `__iter__`

Description Correspond à l'interateur (for i in Vec2)

Retourne `<int>`

Paramètre Rien

16.6.11 `__repr__`

Description Correspond à la représentation en string

Retourne `<string>`

Paramètre Rien

16.6.12 `__eq__`

Description Correspond à la comparaison "=="

Retourne `<bool>`

Paramètre `<Vec2>`

16.6.13 `__neg__`

Description Correspond à la négation

Retourne `<Vec2>`

Paramètre Rien

16.7 Others

Ce fichier n'est pas une classe mais un regroupement de fonctions.

Voici ses fonctions :

16.7.1 `clamp`

Description Force une valeur à être dans un interval

Retourne `<int>` : Valeur finale

Paramètres

- `val <int>` : Valeur de base
- `mini <int>` (None) : Valeur minimale
- `maxi <int>` (None) : Valeur maximale

16.7.2 `wrap_text`

Description Retourne le texte avec une largeur maximale

Retourne `<str>` : Texte finale

Paramètres

- `text <str>` : Texte à wrapper
- `font ` : Police du texte
- `width <int>` : Largeur du texte

Note : Cette fonction se base sur les espaces. Sans espaces, elle est inutile.

16.7.3 `get_images_from_gif`

Description Récupère les frames d'un gif animé

Retourne `<List[str]>` : Liste des chemins des frames

Paramètre Chemin vers le gif