PyEngine4

Release 4.0.0-indev

LavaPower

May 08, 2021

INFORMATIONS

Welcome to documentation of PyEngine4.

PyEngine4 is constantly under development, the documentation is therefore subject to change. Feel free to come back to it as soon as you have a problem.

Note: It is important to remember that PyEngine is an OpenSource project developed by non-professionals. You can also participate with Github.

Summary:

CHAPTER

ONE

INTRODUCTION

PyEngine was created by LavaPower.

PyEngine relies on PyGame to work. It was made to be used in 2D games of all types: Platformer, Pong, breakout games...

You can find tutorials, examples and documentation of the different classes.

Note: PyEngine is still very young and still very limited.

CHAPTER

TWO

DOWNLOAD AND INSTALLATION

- Have Python, PyGame and PyQt installed
- Download and decompress GitHub files (http://github.com/Lycos-Novation/PyEngine4)
- PyEngine is downloaded and installed

CHAPTER

THREE

FAQ

3.1 What is PyEngine?

PyEngine is a game engine for creating games 2D videos more easily.

3.2 Why create PyEngine?

To create a video game in python, there is already the very good PyGame.

But when I created my game, I had to create systems (such as the entity system) that are useful for all. So I chose to create PyEngine (which uses PyGame itself) (And then it allows a good training in Python).

3.3 What are the dependencies of PyEngine?

Apart from Python, PyEngine uses PyGame and PyQt. But games built with PyEngine only use PyGame.

3.4 What are the platforms where PyEngine can be used?

If you can use PyGame, PyQt and Python, you can use PyEngine.

3.5 I would like to participate in the development of PyEngine, how do I do it?

Send me a message by Discord (LavaPower / Lyos#2480) to see what you can do or go to GitHub

CHAPTER

FOUR

CHANGELOG

Caption : [+] Addition, [~] Modification, [-] Deletion, [#] Bug fix

4.1 V 1.1.0 : Kisure Update - 09/05/21 (INDEV)

- [+] Engine Settings
- [+] Utility classes : Vec2, Color, Math
- [+] requirements.txt
- [+] Components : ButtonComponent, TimeScaleComponent, MusicComponent, SoundComponent, AnimComponent
- [+] Assets : Sounds
- [+] Project Settings : Number of mixer channels
- [+] Pong assets in project_files (must be relinked to project)
- [+] Color picker in components widgets
- [+] Tag in GameObject
- [~] Open a project from a different version of PE4
- [~] Upgrade UI of ComponentsWidget and AssetsExplorer
- [~] Can now use more than one key for ControlComponent
- [~] Specify size of collision in CollisionComponent
- [~] Modification of Game Properties is now apply (title, width, height)
- [~] Textures have their own directory in build
- [~] You can now drag and drop scripts to ScriptComponent and textures to SpriteComponent or SpriteSheetComponent
- [#] Down_keys and Down_mousebutton in Engine can make crash
- [#] Missing Assets make crashes
- [#] Can't launch game if Sound or Texture path is None
- [#] Create gameobject without opened scene make crash

4.2 V 1.0.0 : Colombe Update - 04/04/21 (LATEST)

- First version

CHAPTER

FIVE

PROJECTWINDOW

The first window you will see when you launch PyEngine4 is the ProjectWindow.

This window list all the projects of PyEngine4. You can launch a project with a double click.

Note: A project made with other version of PyEngine can be launched but may produce problems.

You can also delete an existed project or create a new project with a name, an author and an icon (the last is not obligatory).

CHAPTER

SIX

SETTINGS

There are two types of settings : Project and Engine Settings.

6.1 Project

- engine_version: Version of PyEngine4
- width: Width of window
- height: Height of window
- mainScene: Name of the main scene (will be the first played scene
- numberMixerChannels: Number of audio channel for SoundComponent

6.2 Engine

- editor: Path of script editor.

CHAPTER

SEVEN

EDITOR

One of principal components of PyEngine is the Editor. He's composed of four other components.

7.1 Viewport

In the center of the screen, there is the viewport.

You can see the current selected scene here.

> **Warning:** The viewport haven't got any logic or game loop.

7.2 ProjectExplorer

At the bottom, there is the project explorer.

You can choose if you want see scripts, prefabs, textures or scenes. If you right click on the right part, you can create new assets or delete an old one.

7.2.1 Scripts

A Script in PyEngine4 is a python file where you create a logic for an object.

On creation, PyEngine4 ask a name for the script. If you double click on it, you will open the script.

7.2.2 Prefabs

Coming Soon

7.2.3 Textures

A texture is a .PNG or .JPG file which can be used for a sprite.

On creation, PyEngine4 ask a name and a path to the file. If you double click on it, you will be able to change the path.

7.2.4 Sounds

A sounds is a .WAV, .OGG or .MP3 which can be used in AudioComponent or MusicComponent.

On create, PyEngine4 ask a name and a path to the file. If you double click on it, you will be able to change the path.

7.2.5 Scenes

A scene is a 2D world where you can put objects. You must have a scene for any game.

On creation, PyEngine4 ask a name for the scene.

7.3 SceneTree

On the left, you can see the Scene Tree.

It shows the objects which are in the current selected scene.

You can create a new game object with selected an already existed game object and make a right click. You will choose a name and the new game object will be the child of the existed.

You can also delete a game object with a right click.

7.4 Inspector

On the right, there is the inspector.

This panel will show the properties and components of the selected game object (or asset in the case of texture).

You can add a new component in a normal game object with the button on the bottom or with right click. You can also delete a component of a normal game object with right click.

CHAPTER

EIGHT

GAME

Class which represents the core of the game.

8.1 Variables

- engine: Utility class to control the game (Engine)
- title: Title of the Game (string)
- width: Width of Window (integer)
- height: Height of Window (integer)
- scenes: List of scenes (list of Scene)
- current_scene: Index of current displayed scene (integer)
- clock: Pygame's clock to control time of Game (Clock)
- screen: Window of Game (Surface)
- is_running: True if game is currently running (boolean)

Warning: Most of these variables mustn't be manipulated. Only current_scene, title, width, height, is_running and getting engine is safe.

8.2 Functions

- stop(): Stop Game
- get_fps(): Return current FPS

CHAPTER

NINE

SCENE

Class which represents a world for the game.

9.1 Variables

- bg_color: Color of background (Color)
- timescale: Scale of time (integer)
- game_objects: List of Game Objects own by the scene (List of GameObject)

Warning: game_objects mustn't be manipulated, use functions.

9.2 Functions

- add_game_objects(game_objects): Adding Game Objects to the scene (game_objects -> List of GameObjects)
- add_game_object(game_object): Add a Game Object to the scene (game_object -> GameObject)
- get_game_object(id): Getting Game Object by id or None if any Game Object is found (id -> integer)

CHAPTER

TEN

GAMEOBJECT

Class which represents an object of the game.

10.1 Variables

- parent: Parent of current object or None (GameObject)
- childs: List of childs game objects of current object (list of GameObject)
- components: List of components of current object (list of Components)
- id_: Id of object (integer)
- tag : Tag of object (string)

Warning: Most of these variables mustn't be manipulated. Only id_ and parents are safe on reading and tag is safe reading and writing.

10.2 Functions

- add_child(child): Add a child to object (child -> GameObject)
- add_component(comp): Add a component to object (comp -> Component)
- get_component(name): Return a component by his name or None if any component is found (name -> string)

CHAPTER

ELEVEN

ENGINE

Utility class which can onlly be used in script.

11.1 Variables

- game: Game object (Game)
- pg_constants: Constants from PyGame (module)
- down_keys: List of keys which are currently down (list)
- down_mousebuttons: List of mouse buttons which are currently down (list)

11.2 Functions

- get_game_object(id): Return game object which have specific id (id -> integer)
- get_current_scene(): Return scene which is currently selected
- stop_game(): Stop game

CHAPTER

TWELVE

ANIMCOMPONENT

This component add animations to game object.

Warning: This component use the TransformComponent and it can't work without it.

12.1 Editor

In the editor, you can add, remove and manage animations but also change frames per seconds and the current playing animation.

12.2 Script

12.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- fps: Number of frames per seconds (int)
- playing: Name of current animation (string)
- anims: Dictionnary of animations (dict<string, AnimSprite/AnimSpriteSheet>)
- current_sprite: Index of current displayed sprite (int)

12.2.2 Functions

No public function in this component.

CHAPTER

THIRTEEN

AUTOCOMPONENT

This component create automatic movement and rotation to game object.

Warning: This component use the TransformComponent and it can't work without it.

13.1 Editor

In the editor, you can change the movement and the rotation with differents spins. There is also a checkbox to activate/desactivate the component.

13.2 Script

13.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- move: Movement of component (Vec2)
- rotation: Rotation of component in degrees (integer)
- active: True if component is active

13.2.2 Functions

No public function in this component.

CHAPTER

FOURTEEN

BASICPHYSICCOMPONENT

This component add gravity to game object.

Warning: This component use the TransformComponent and it can't work without it.

14.1 Editor

In the editor, you can change gravity with a spin.

14.2 Script

14.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- gravity: Current gravity of component (integer)
- max_gravity: Maximum gravity of component (integer)
- grounded: True if object can't be move by gravity (boolean)
- time: Time must be elipsed beform apply gravity (float)

Note: You may not change grounded and time.

Warning: grounded only work with the CollisionComponent

14.2.2 Functions

No public function in this component.

CHAPTER

FIFTEEN

BUTTONCOMPONENT

This component add button to game object.

Warning: This component use the TransformComponent and it can't work without it.

15.1 Editor

In the editor, you can change callback, backgroud color, size, text and informations from font (name, size, bold, italic, underline, color, antialias).

15.2 Script

15.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- bg: Background color (Color)
- callback: Component and its function which will be call on click (string)
- size: Size of component (Vec2)
- text: Text will be displayed (string)
- font_name: Name of font used (string)
- font_size: Size of font used (integer)
- font_bold: True if text is in bold (boolean)
- font_italic: True if text is in italic (boolean)
- font_underline: True if text is underline (boolean)
- font_color: Color of font used (Color)
- font_antialias: True if antialias is active (boolean)
- transformed_font: Final font
- render: Final render.

Note: callback use a specific format : "NameComponent - NameFunction". For example: "myScript - click" is a valid callback.

Warning: You mustn't change transformed_font directly but use update_font function.

Warning: You mustn't change render directly but use update_render function.

15.2.2 Functions

- update_font(): Update font of component
- update_render(): Update render of component

CHAPTER

SIXTEEN

COLLISIONCOMPONENT

This component add collision and collision callback to game object.

Warning: This component use the TransformComponent and a component with a render and it can't work without them.

16.1 Editor

In the editor, you can change the size of collision, if the object is solid with a checkbox and specify a callback by giving the name of component and the function will be called.

16.2 Script

16.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- solid: True if object is solid (boolean)
- callback: Component and its function which will be call on collide (string)
- size: Size of the collider (Vec2)

Note: callback use a specific format : "NameComponent - NameFunction". For example: "myScript - collide" is a valid callback.

Note: A function which will be used as a callback must accept two arguments : the object which collide with and the cause of collision.

16.2.2 Functions

- can_go(position, cause="UNKNOWN"): Return if object can go to a position. (position -> Vec2, cause -> string)

CHAPTER

SEVENTEEN

TRANSFORMCOMPONENT

This component allow controlling game object.

17.1 Editor

In the editor, you can change keys, control type and speed.

Note: Keys with pygame's names and space to separate keys.

Note: List of Control type : - FOURDIRECTION : Object can move in four directions (Up, Down, Left, Right) - DOWNUP : Object can move in two directions (Up, Down) - LEFTRIGHT : Object can move in two directions (Left, Right) - CLASSICJUMP : Basic platformer controler. Move in X axis and can make a jump. (Use BasicPhysicComponent)

17.2 Script

17.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- keys: Dictionnary of keys used. Use UPJUMP, DOWN, RIGHT, LEFT as keys. (dictionnary)
- control_type: Type of control (string)
- speed: Speed of object (integer)

17.2.2 Functions

No public function in this component.

CHAPTER

EIGHTEEN

MUSICCOMPONENT

This component add music to game object.

18.1 Editor

In the editor, you can change music by selection or drag and drop, volume, if is currently playing and if it loops.

18.2 Script

18.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- music: Current music (string)
- volume: Current volume from 0 to 100 (integer)
- play: True if is current playing (boolean)
- loop: True if it loops (boolean)

18.2.2 Functions

No public function in this component.

CHAPTER

NINETEEN

SOUNDCOMPONENT

This component add sounds to game object.

19.1 Editor

In the editor, you can change sound by selecting or drag and drop and volume.

19.2 Script

19.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- sound: Current sound (string)
- volume: Current volume from 0 to 100 (integer)

19.2.2 Functions

- play(): Play the current sound

CHAPTER

TWENTY

SPRITECOMPONENT

This component add sprite to game object.

> **Warning:** This component use the TransformComponent and it can't work without it.

20.1 Editor

In the editor, you can change sprite by selecting json file or drag and drop.

20.2 Script

20.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- sprite: Resource file. Will search in resources folder. (string)
- render: Final render.

> **Warning:** You mustn't change render directly but use update_render function.

20.2.2 Functions

- update_render(): Update render of component

CHAPTER

TWENTYONE

SPRITESHEETCOMPONENT

This component add spritesheet to game object.

Warning: This component use the TransformComponent and it can't work without it.

21.1 Editor

In the editor, you can change spritesheet by selecting json file or drag and drop but also the numbers of sprite and the current index of sprite.

21.2 Script

21.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- sprite: Resource file. Will search in resources folder. (string)
- sprite_number: Number of sprite in width and height. (list of two integers)
- current_sprite: Index of current displayed sprite (integer)
- render: Final render.

Note: Current_sprite start at 0.

Warning: You mustn't change render directly but use update_render function.

21.2.2 Functions

- update_render(): Update render of component

CHAPTER

TWENTYTWO

TEXTCOMPONENT

This component add text to game object.

Warning: This component use the TransformComponent and it can't work without it.

22.1 Editor

In the editor, you can change text and informations from font (name, size, bold, italic, underline, color, antialias).

22.2 Script

22.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- text: Text will be displayed (string)
- font_name: Name of font used (string)
- font_size: Size of font used (integer)
- font_bold: True if text is in bold (boolean)
- font_italic: True if text is in italic (boolean)
- font_underline: True if text is underline (boolean)
- font_color: Color of font used (Color)
- font_antialias: True if antialias is active (boolean)
- transformed_font: Final font
- render: Final render.

Warning: You mustn't change transformed_font directly but use update_font function.

Warning: You mustn't change render directly but use update_render function.

22.2.2 Functions

- rendered_size(text): Return size of text which be rendered with the font of component (text -> string)
- update_font(): Update font of component
- update_render(): Update render of component

CHAPTER

TWENTYTHREE

TRANSFORMCOMPONENT

This component add position, rotation and scale to game object.

23.1 Editor

In the editor, you can change position, rotation or scale with differents spins.

23.2 Script

23.2.1 Variables

- game_object: Object which own this component (GameObject)
- engine: Utility class to control the game (Engine)
- rotation : Rotation of object in degrees (integer)
- position : Position X and Y of object (Vec2)
- scale : Scale X and Y of object (Vec2)

23.2.2 Functions

No public function in this component.

CHAPTER

TWENTYFOUR

COLOR

This class represents a Color.

24.1 Variables

- r: Red component of Color (integer)
- g: Green component of Color (integer)
- b: Blue component of Color (integer)
- a: Alpha component of Color (integer)

Note: These components must be between 0 and 255

24.2 Functions

- darker(force=1): Return a darker color. (force -> integer)
- lighter(force=1): Return a lighter color. (force -> integer)
- rgb(): Return a list with R, G, B values
- rgba(): Return a list with R, G, B, A values
- html(): Return html value
- from_rgb(r, g, b): Return a color with R, G, B value (r -> integer, g -> integer, b -> integer)
- from_rgba(r, g, b, a): Return a color with R, G, B, A value (r -> integer, g -> integer, b -> integer, a -> integer)
- from_color(color): Return a color from a color (copy) (color -> Color)
- from_html(html): Return a color from html value (html -> string)
- from_name(name): Return a color from its name (name -> string)

Note: List of defined names : “WHITE”, “BLACK”, “GRAY”, “RED”, “GREEN”, “BLUE”, “FUCHSIA”, “YELLOW”, “CYAN”, “LIME”, “BROWN”, “NAVY_BLUE”, “OLIVE”, “PURPLE”, “TEAL”, “SILVER”, “ORANGE”

Warning: All functions started by “from_xxx” are classmethods. To use it, you must write Color.<method> and not use a instance.

CHAPTER

TWENTYFIVE

MATH

This file regoups a list of differents functions.

25.1 Functions

- clamp(value, mini=None, mini=None): Clamp a value between range. (Every type is allowed if there are compatible with operators less than and greater than)

CHAPTER

TWENTYSIX

VEC2

This class represents a Vector 2D.

26.1 Variables

- x: X coords (integer)
- y: Y coords (integer)

26.2 Functions

- coords(): Return list with X and Y coords
- set_coords(x, y): Set x and y coords (x -> integer, y -> integer)
- normalized(): Return normalized Vec2
- zero: Return Vec2(0, 0)

Warning: Function zero is a classmethod. To use it, you must write Vec2.zero().