API Documentation

API Documentation

August 29, 2015

Contents

1 Module engine

A Python3 game engine and base game object, by John Aycock <aycock@ucalgary.ca>.

1.1 Functions

init_screen(*width*, *height*)

This should be called once only, before anything in this module is used.

init_engine(*delay*=`0.01`)

(Re)initializes the game engine. Only one game engine may exist at any one time. The optional parameter specifies a delay added to each game time step, in seconds; the value may be a floating point number.

add_random_event(*prob*, *fn*)

Defines a callback function that is invoked with probability prob at each time step. Multiple random event callback functions may be registered at the same time. The probability must be a float in the range [0.0, 1.0].

set_keyboard_handler(*fn*)

Sets callback function to invoke when a key is pressed. The function is passed the name of the key pressed as a string. Only one keyboard handler may be registered at a time.

set_mouse_handler(*fn*)

Sets callback function to invoke when the mouse button is pressed. The function is passed the x and y coordinates where the mouse was clicked. Only one mouse handler may be registered at a time.

register_collision(*class1*, *class2*, *fn*)

Instructs the game engine to invoke the callback routine fn when a collision is detected between an instance of class1 and an instance of class2. Note that there is no ordering guaranteed for how game objects are tested for collision, so both combinations of class1/class2 and class2/class1 will need to be registered.

add_obj(*obj*)

Adds a GameObject-derived object instance to the game.

del_obj(*obj*)

Removes a GameObject-derived object instance from the game.

exit_engine()

Instructs the game engine to exit on the next time step.

engine()

Starts the game engine running.

1.2 Variables

Name	Description
___doc___	**Value:** ...
___package___	**Value:** `None`

1.3 Class GameObject

This is a base game object, suitable for subclassing or wrapping fish.

1.3.1 Methods

___init___(*self*, *x*, *y*, *deltax*, *deltay*, *shape*, *color*)

Instantiates a game object at position (x, y) with the given shape and color, to move by (deltax, deltay) each time step.

heading(*self*)

Returns the direction the object should be facing. By default, this is towards where the object will be moving.

draw(*self*)

Draws the object at its current (x, y) coordinates.

delete(*self*)

Invoked to delete an object.

erase(*self*)

Removes the object's image on screen.

update(*self*)

Invoked to update the object's image on screen, a draw-new-then-erase-old sequence.

move(*self*)

Invoked to move the object's (x, y) position on each time step.

isstatic(*self*)

Returns a Boolean value: True (static, unmoving object), or False (the default, a moving object).

isoob(*self*)

Returns True/False to indicate if the object is out of bounds or not. By default, the screen height/width and the object's (x, y) position are used to determine this.

step(*self*)

Called by the game engine each time step to allow the game object to update accordingly. The object's age (in game time steps) is updated and, if it's a moving object, invokes methods to perform the move and update. Moving out of bounds causes the object to be deleted from the game.

Index