

# Documentation/Installation Instructions

**Pscreen is a Python module that provides simple 2D drawing and graphics functions as well as mouse and keyboard I/O.**

## Installation Instructions

Download the full pscreen installer from this website and run it. Alternatively, if the pyglet and pygame libraries are already installed, you can download just the individual installer for pscreen. They can be obtained from [www.pygame.org](http://www.pygame.org) and [www.pyglet.org](http://www.pyglet.org), respectively.

## General Functions

**LoadScreen**(resolution=(800, 600), fullscreen=False, selectedBackend="pygame"): Initialize the window. The possible choices for selectedBackend are "pygame-basic", "pygame", and "pyglet".

**ClearScreen**(color=(255, 255, 255, 255)): Fill the entire screen with the specified color. Should be placed at the top of the game loop.

**UpdateScreen**(): Update the contents of the window. Should be placed at the end of the game loop.

**UnloadScreen**(): Unload the screen and all associated resources.

## Drawing Functions

**PixelSet**(x, y, color=(255, 255, 255, 255)): Color in the pixel at the specified position.

**PixelGet**(x, y): Return a tuple describing the color at the specified position.

**Line**(x1, y1, x2, y2, color=(255, 255, 255, 255), linewidth=1): Draw a line from (x1, y1) to (x2, y2).

**Rectangle**(x1, y1, x2, y2, color=(255, 255, 255, 255), linewidth=1): Draw a rectangle using (x1, y1) and (x2, y2) as two corners. If linewidth is zero, the rectangle will be filled.

**Triangle**(x1, y1, x2, y2, x3, y3, color=(255, 255, 255, 255), linewidth=1): Draw a triangle using (x1, y1), (x2, y2) and (x3, y3) as the corners. If linewidth is zero, the triangle will be filled.

**Circle**(x, y, r, color=(255, 255, 255, 255), linewidth=1): Draw a circle at position (x, y) with radius r. If linewidth is zero, the circle will be filled.

**Ellipse**(x, y, w, h, color=(255, 255, 255, 255), linewidth=1): Draw an ellipse at position (x, y) with width w and height h. If linewidth is zero, the ellipse will be filled.

## Keyboard Functions

**KeyIsPressed**(key): Returns true if this key is currently pressed. key can be a string; for

example, "a", "return", and "escape" are all valid.

**KeyIsNotPressed(key)**: Returns true if this key is NOT currently pressed.

**KeyGetPressedList()**: Return a list of currently pressed keys.

### *Mouse Functions*

**MouseGetButtonL()**: Returns True if the left mouse button is held down.

**MouseGetButtonR()**: Returns True if the right mouse button is held down.

**MouseGetButtonM()**: Returns True if the middle mouse button is held down.

**MouseGetX()**: Returns the x position of the mouse.

**MouseGetY()**: Returns the y position of the mouse.

**MouseGetPosition()**: Returns a tuple containing the x and y positions of the mouse.

**MouseGetButtons()**: Returns a 3-element tuple describing the state of each mouse button, such as (left, middle, right). Each element will be True or False.

### *Font Functions*

**FontSelect(fontName="Arial", fontSize="24")**: Set the current font. Must be done prior to rendering text.

**FontWrite(x, y, text, color=(255, 255, 255, 255))**: Draw text at the specified position. Currently, the alpha value (the last color parameter) is not supported in pygame.

### *Music Functions*

**MusicLoad(filename)**: Load filename as the current background music. Multiple formats are supported, including .mp3, .wav, and .ogg.

**MusicPlay()**: Play the currently loaded background music.

**MusicPause()**: Pause the currently playing background music.

**MusicUnPause()**: Unpause the currently loaded background music.

**MusicFade(seconds)**: Fade out the currently playing background music over the given time.

**MusicSetVolume(volumePercent=40)**: Set the volume of the currently loaded background music.

**MusicGetVolume()**: Return the volume of the currently loaded background music.

**MusicStop()**: Stop the currently playing background music.

### *Sound Functions*

**SoundLoad(filename, slot)**: Load the specified sound into one of 256 slots (slot should be 0 - 255).

**SoundPlay(slot)**: Play the sound stored in the given slot.

**SoundStop(slot)**: Stop the sound stored in the given slot.

**SoundSetVolume**(slot, volumePercent): Set the volume of the sound in the given slot.

### ***Sprite Functions***

**SpriteLoad**(filename, slot): Load the specified sprite into one of 256 slots (slot should be 0 - 255).

**SpriteRender**(x, y, slot, rotation=0, scale=1, flipH=False, flipV=False, alpha=255): Render the sprite in the specified slot at the specified position. Rotation specifies a counterclockwise rotation in angles; scale specifies a scaling factor in both dimensions; flipH and flipV flip the sprite horizontally and vertically, respectively. The alpha parameter is not currently supported in pygame.

**SpriteSimpleRender**(x, y, slot): A simpler, quicker method of rendering a sprite.

**SpriteWidth**(slot): Return the width of the sprite in the specified slot.

**SpriteHeight**(slot): Return the height of the sprite in the specified slot.

### ***Framerate Functions***

**FrameGetLimit**(): Returns the current frames per second.

**FrameSetLimit**(fps): Cap the framerate at the specified frames per second.

**FrameDelay**(): Ticks the internal clock object; this should be called during every iteration of the game loop. The function returns a value that indicates the amount of time elapsed since the last update.