Pscreen to Pygame conversions

By Jason Witherell

Last Updated: 1/13/2016

This document is meant to assist in the pscreen to pygame "conversion process". I tried to (as concisely as possible) show how each "construct" works in the two libraries. If there are missing items, let me know and I'll try to update this document.

A special thanks to Paul Yost (Yost Engineering) for developing his excellent pscreen wrapper – it's how I learned python ☺

Pscreen	Pygame	Description
`pscreen.loadScreen((800,600))`	`pygame.init()` `screen = pygame.display.set_mode((800, 600))`	*Creates the window and a pygame surface to draw to (representing the back-buffer)*
`pscreen.unloadScreen()`	`pygame.quit()`	*Destroys the window and shuts down.*
`pscreen.updateScreen()`	`pygame.display.flip()`	*Should be called at the end of the game loop.*
`pscreen.pixelSet(x, y, color)`	`surf.set_at((x, y), color)`	*Sets an individual pixel to a given color. In pygame, you can do this to any surface (in pscreen, just the screen)*
`pscreen.pixelGet(x, y) -> color`	`surf.get_at((x, y)) -> color`	*Gets the color at the given spot (in pygame on surf, in pscreen the screen)*
`pscreen.line(x1, y1, x2, y2, color)`	`pygame.draw.line(surf, color, (x1, y1), (x2, y2), width=1)`	*Draws a line*
`…the rest of the pscreen drawing commands are just wrappers…they should be pretty straightforward to map to pygame`	`See http://www.pygame.org/docs/ for the drawing commands (under the 'draw' section at the top)`	

pscreen.keyIsPressed(keyName) **# e.g.** **pscreen.keyIsPressed("escape")**	**# You must call this exactly once per game loop** **pygame.event.get()** **# …or one of these two. If just doing** **# "device-polling" (this is all that** **# pscreen supports), it doesn't matter** **# which.** **pygame.event.poll()** **pygame.event.pump()** **# Then to actually test whether a key is** **# down do this at least once per frame** **keysPressed = pygame.key.get_pressed()** **# Finally, to check the state of a key** **# see if the Boolean in the "slot" of** **# interest is True** **if keysPressed[pygame.K_ESCAPE]:** **…** **# You can see the full list of key codes** **# at http://www.pygame.org/docs/ref/key.html**	***.Pygame also supports another type of input handling ("event handling"). See the examples from my ETGG1802 course for examples…***
pscreen.mouseGetButtonL() -> Boolean	**# You must process waiting events (as we** **# did on the row above for keyboards).** **# Make sure this method is only called** **# once!** **# Then, to test a mouse button:** **mPress = pygame.mouse.get_pressed()** **# This is a list of 3 booleans** **# (left, middle, right). So to test** **# whether the left mouse is down, do:** **if mPress[0]:** **…**	*As for keyboard events, pygame supports an "event-handling" mode. See the ETGG1802 examples for more.*
pscreen.mouseGetX() -> int **pscreen.mouseGetY() -> int**	**# Make sure you've processed events…** **pygame.mouse.get_pos() -> (x, y)**	
pscreen.spriteLoad(filename, slot)	**pygame.image.load(filename) -> surface**	

`pscreen.spriteRender(x, y, slot, rotation=0, scale=1, flipH=False, flipV=False, alpha=255)`	`# I'm assuming img is a pygame surface.`
	`# If a transform is processed (e.g.`
	`# scaling), use the "newImg" in future`
	`# calls (not "img")`
	
	`# Optionally scale:`
	`new_size = (scale * img.get_width(),`
	`scale * img.get_height())`
	`newImg = pygame.transform.scale(img,`
	`new_size)`
	
	`# Optionally rotate:`
	`newImg = pygame.transform.rotate(img,`
	`rotation)`
	
	`# Optionally flipH`
	`newImg = pygame.transform.flip(True,`
	`False)`
	
	`# Optionally flipV`
	`newImg = pygame.transform.flip(False,`
	`True)`
	
	`# Optionally change the alpha. Note:`
	`# only works with images *without*`
	`# a per-pixel alpha channel.`
	`img.set_alpha(alpha)`
	
	`# Calculate the upper-left corner`
	`leftx = x – img.get_width() // 2`
	`topy = y – img.get_height() // 2`
	
	`# Finally, blit it to another surface`
	`surf.blit(img, (leftx, topy))`
		This code centers the sprite at the given x, y position. Pygame blits with the upper-left as the 'anchor' – so you must calculate the true upper-left based on where you want the center to appear. Pscreen just does this internally