"""

PygButton v0.1.0

PygButton (pronounced "pig button") is a module that implements UI buttons for Pygame.

PygButton requires Pygame to be installed. Pygame can be downloaded from http://pygame.org

PygButton was developed by Al Sweigart (al@inventwithpython.com)

https://github.com/asweigart/pygbutton

EXEMPLARY, OR

CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR

SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON

ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING

NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF

ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those of the

authors and should not be interpreted as representing official policies, either expressed

or implied, of Al Sweigart.

"""

```
import pygame
from pygame.locals import *

pygame.font.init()
PYGBUTTON_FONT = pygame.font.Font('freesansbold.ttf', 14)

BLACK     = (  0,   0,   0)
WHITE     = (255, 255, 255)
DARKGRAY  = ( 64,  64,  64)
GRAY      = (128, 128, 128)
LIGHTGRAY = (212, 208, 200)

class PygButton(object):
    def __init__(self, rect=None, caption='', bgcolor=LIGHTGRAY, fgcolor=BLACK, font=None,
```

```
normal=None, down=None, highlight=None):
        """Create a new button object. Parameters:
           rect - The size and position of the button as a pygame.Rect object
              or 4-tuple of integers.
           caption - The text on the button (default is blank)
           bgcolor - The background color of the button (default is a light
              gray color)
           fgcolor - The foreground color (i.e. the color of the text).
              Default is black.
           font - The pygame.font.Font object for the font of the text.
              Default is freesansbold in point 14.
           normal - A pygame.Surface object for the button's normal
              appearance.
           down - A pygame.Surface object for the button's pushed down
              appearance.
           highlight - A pygame.Surface object for the button's appearance
              when the mouse is over it.

           If the Surface objects are used, then the caption, bgcolor,
           fgcolor, and font parameters are ignored (and vice versa).
           Specifying the Surface objects lets the user use a custom image
           for the button.
           The normal, down, and highlight Surface objects must all be the
           same size as each other. Only the normal Surface object needs to
           be specified. The others, if left out, will default to the normal
```

```
        surface.
        """
    if rect is None:
        self._rect = pygame.Rect(0, 0, 30, 60)
    else:
        self._rect = pygame.Rect(rect)

    self._caption = caption
    self._bgcolor = bgcolor
    self._fgcolor = fgcolor

    if font is None:
        self._font = PYGBUTTON_FONT
    else:
        self._font = font

    # tracks the state of the button
    self.buttonDown = False # is the button currently pushed down?
    self.mouseOverButton = False # is the mouse currently hovering over the button?
    self.lastMouseDownOverButton = False # was the last mouse down event over the mouse button? (Used to track clicks.)
    self._visible = True # is the button visible
    self.customSurfaces = False # button starts as a text button instead of having custom images for each surface

    if normal is None:
        # create the surfaces for a text button
```

```
            self.surfaceNormal = pygame.Surface(self._rect.size)
            self.surfaceDown = pygame.Surface(self._rect.size)
            self.surfaceHighlight = pygame.Surface(self._rect.size)
            self._update() # draw the initial button images
        else:
            # create the surfaces for a custom image button
            self.setSurfaces(normal, down, highlight)

    def handleEvent(self, eventObj):
        """All MOUSEMOTION, MOUSEBUTTONUP, MOUSEBUTTONDOWN event objects
        created by Pygame should be passed to this method. handleEvent() will
        detect if the event is relevant to this button and change its state.

        There are two ways that your code can respond to button-events. One is
        to inherit the PygButton class and override the mouse*() methods. The
        other is to have the caller of handleEvent() check the return value
        for the strings 'enter', 'move', 'down', 'up', 'click', or 'exit'.

        Note that mouseEnter() is always called before mouseMove(), and
        mouseMove() is always called before mouseExit(). Also, mouseUp() is
        always called before mouseClick().

        buttonDown is always True when mouseDown() is called, and always False
        when mouseUp() or mouseClick() is called. lastMouseDownOverButton is
        always False when mouseUp() or mouseClick() is called."""

        if eventObj.type not in (MOUSEMOTION, MOUSEBUTTONUP, MOUSEBUTTONDOWN)
```

```
or not self._visible:
            # The button only cares bout mouse-related events (or no events, if it is invisible)
            return []

        retVal = []

        hasExited = False
        if not self.mouseOverButton and self._rect.collidepoint(eventObj.pos):
            # if mouse has entered the button:
            self.mouseOverButton = True
            self.mouseEnter(eventObj)
            retVal.append('enter')
        elif self.mouseOverButton and not self._rect.collidepoint(eventObj.pos):
            # if mouse has exited the button:
            self.mouseOverButton = False
            hasExited = True # call mouseExit() later, since we want mouseMove() to be handled
before mouseExit()

        if self._rect.collidepoint(eventObj.pos):
            # if mouse event happened over the button:
            if eventObj.type == MOUSEMOTION:
                self.mouseMove(eventObj)
                retVal.append('move')
            elif eventObj.type == MOUSEBUTTONDOWN:
                self.buttonDown = True
                self.lastMouseDownOverButton = True
                self.mouseDown(eventObj)
```

```
            retVal.append('down')
    else:
        if eventObj.type in (MOUSEBUTTONUP, MOUSEBUTTONDOWN):
            # if an up/down happens off the button, then the next up won't cause mouseClick()
            self.lastMouseDownOverButton = False

    # mouse up is handled whether or not it was over the button
    doMouseClick = False
    if eventObj.type == MOUSEBUTTONUP:
        if self.lastMouseDownOverButton:
            doMouseClick = True
        self.lastMouseDownOverButton = False

        if self.buttonDown:
            self.buttonDown = False
            self.mouseUp(eventObj)
            retVal.append('up')

        if doMouseClick:
            self.buttonDown = False
            self.mouseClick(eventObj)
            retVal.append('click')

    if hasExited:
        self.mouseExit(eventObj)
        retVal.append('exit')
```

```
        return retVal


    def draw(self, surfaceObj):
        """Blit the current button's appearance to the surface object."""
        if self._visible:
            if self.buttonDown:
                surfaceObj.blit(self.surfaceDown, self._rect)
            elif self.mouseOverButton:
                surfaceObj.blit(self.surfaceHighlight, self._rect)
            else:
                surfaceObj.blit(self.surfaceNormal, self._rect)


    def _update(self):
        """Redraw the button's Surface object. Call this method when the button has changed
appearance."""
        if self.customSurfaces:
            self.surfaceNormal    = pygame.transform.smoothscale(self.origSurfaceNormal,
self._rect.size)
            self.surfaceDown      = pygame.transform.smoothscale(self.origSurfaceDown,
self._rect.size)
            self.surfaceHighlight = pygame.transform.smoothscale(self.origSurfaceHighlight,
self._rect.size)
            return

        w = self._rect.width # syntactic sugar
        h = self._rect.height # syntactic sugar

        # fill background color for all buttons
```

```
        self.surfaceNormal.fill(self.bgcolor)
        self.surfaceDown.fill(self.bgcolor)
        self.surfaceHighlight.fill(self.bgcolor)

        # draw caption text for all buttons
        captionSurf = self._font.render(self._caption, True, self.fgcolor, self.bgcolor)
        captionRect = captionSurf.get_rect()
        captionRect.center = int(w / 2), int(h / 2)
        self.surfaceNormal.blit(captionSurf, captionRect)
        self.surfaceDown.blit(captionSurf, captionRect)

        # draw border for normal button
        pygame.draw.rect(self.surfaceNormal, BLACK, pygame.Rect((0, 0, w, h)), 1) # black border around everything
        pygame.draw.line(self.surfaceNormal, WHITE, (1, 1), (w - 2, 1))
        pygame.draw.line(self.surfaceNormal, WHITE, (1, 1), (1, h - 2))
        pygame.draw.line(self.surfaceNormal, DARKGRAY, (1, h - 1), (w - 1, h - 1))
        pygame.draw.line(self.surfaceNormal, DARKGRAY, (w - 1, 1), (w - 1, h - 1))
        pygame.draw.line(self.surfaceNormal, GRAY, (2, h - 2), (w - 2, h - 2))
        pygame.draw.line(self.surfaceNormal, GRAY, (w - 2, 2), (w - 2, h - 2))

        # draw border for down button
        pygame.draw.rect(self.surfaceDown, BLACK, pygame.Rect((0, 0, w, h)), 1) # black border around everything
        pygame.draw.line(self.surfaceDown, WHITE, (1, 1), (w - 2, 1))
        pygame.draw.line(self.surfaceDown, WHITE, (1, 1), (1, h - 2))
        pygame.draw.line(self.surfaceDown, DARKGRAY, (1, h - 2), (1, 1))
```

```
    pygame.draw.line(self.surfaceDown, DARKGRAY, (1, 1), (w - 2, 1))
    pygame.draw.line(self.surfaceDown, GRAY, (2, h - 3), (2, 2))
    pygame.draw.line(self.surfaceDown, GRAY, (2, 2), (w - 3, 2))

    # draw border for highlight button
    self.surfaceHighlight = self.surfaceNormal


def mouseClick(self, event):
    pass # This class is meant to be overridden.
def mouseEnter(self, event):
    pass # This class is meant to be overridden.
def mouseMove(self, event):
    pass # This class is meant to be overridden.
def mouseExit(self, event):
    pass # This class is meant to be overridden.
def mouseDown(self, event):
    pass # This class is meant to be overridden.
def mouseUp(self, event):
    pass # This class is meant to be overridden.


def setSurfaces(self, normalSurface, downSurface=None, highlightSurface=None):
    """Switch the button to a custom image type of button (rather than a
    text button). You can specify either a pygame.Surface object or a
    string of a filename to load for each of the three button appearance
    states."""
```

```
        if downSurface is None:
            downSurface = normalSurface
        if highlightSurface is None:
            highlightSurface = normalSurface

        if type(normalSurface) == str:
            self.origSurfaceNormal = pygame.image.load(normalSurface)
        if type(downSurface) == str:
            self.origSurfaceDown = pygame.image.load(downSurface)
        if type(highlightSurface) == str:
            self.origSurfaceHighlight = pygame.image.load(highlightSurface)

        if self.origSurfaceNormal.get_size() != self.origSurfaceDown.get_size() !=
self.origSurfaceHighlight.get_size():
            raise Exception('foo')

        self.surfaceNormal = self.origSurfaceNormal
        self.surfaceDown = self.origSurfaceDown
        self.surfaceHighlight = self.origSurfaceHighlight
        self.customSurfaces = True
        self._rect = pygame.Rect((self._rect.left, self._rect.top, self.surfaceNormal.get_width(),
self.surfaceNormal.get_height()))



    def _propGetCaption(self):
        return self._caption
```

```
    def _propSetCaption(self, captionText):
        self.customSurfaces = False
        self._caption = captionText
        self._update()

    def _propGetRect(self):
        return self._rect

    def _propSetRect(self, newRect):
        # Note that changing the attributes of the Rect won't update the button. You have to
re-assign the rect member.
        self._update()
        self._rect = newRect

    def _propGetVisible(self):
        return self._visible

    def _propSetVisible(self, setting):
        self._visible = setting

    def _propGetFgColor(self):
        return self._fgcolor

    def _propSetFgColor(self, setting):
```

```
    self.customSurfaces = False
    self._fgcolor = setting
    self._update()

def _propGetBgColor(self):
    return self._bgcolor

def _propSetBgColor(self, setting):
    self.customSurfaces = False
    self._bgcolor = setting
    self._update()

def _propGetFont(self):
    return self._font

def _propSetFont(self, setting):
    self.customSurfaces = False
    self._font = setting
    self._update()

caption = property(_propGetCaption, _propSetCaption)
rect = property(_propGetRect, _propSetRect)
visible = property(_propGetVisible, _propSetVisible)
fgcolor = property(_propGetFgColor, _propSetFgColor)
```

```
bgcolor = property(_propGetBgColor, _propSetBgColor)

font = property(_propGetFont, _propSetFont)
```