Python Object-Oriented Program with Libraries

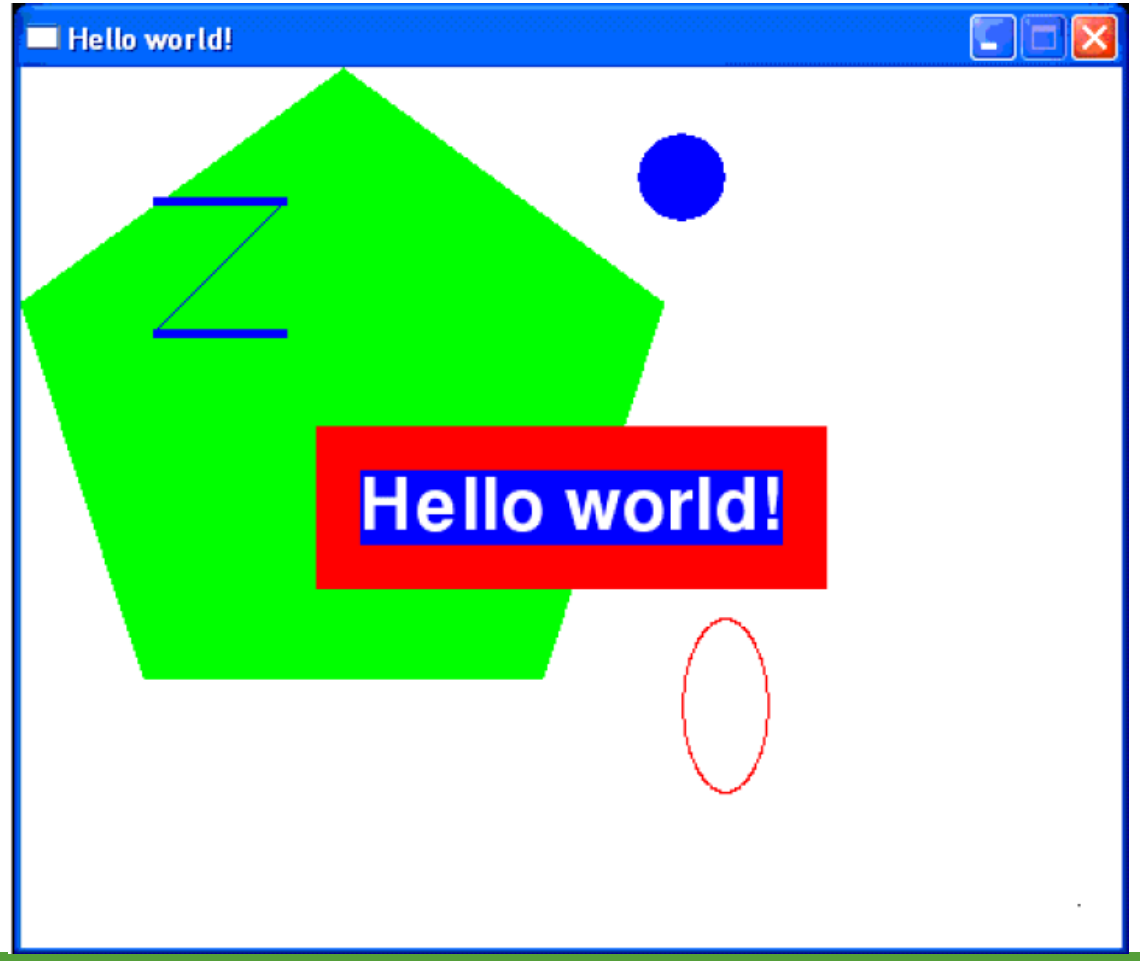
Unit 4: PyGame Tutorial

CHAPTER 2: TEXT

DR. ERIC CHOU

IEEE SENIOR MEMBER

Displaying Text



Fonts

- A font is a complete set of letters, numbers, symbols, and characters of a single style. Here is an example of the same sentence printed in different fonts:

Programming is fun!
Programming is fun!
Programming is fun!
PROGRAMMING IS FUN!
Programming is fun!
Programming is fun!

pygame.font.init() and pygame.font.quit()

- In pygame system. The display, font, and many other canvas painting function has their own controllers. The controllers have a init() function and a quit() function. The init() function is to initialize the initial condition. The quit() function is used to clean up the settings for the controllers.

pygame.font.init()

- Initialize the font module.

pygame.font.quit()

- Uninitialize the font module.

Set Up Fonts

```
# set up fonts
basicFont = pygame.font.SysFont(None, 48)
```

- We create a pygame.font.Font object (which we will just call Font objects for short) by calling the pygame.font.SysFont() function.
- The first parameter is the name of the font, but we will pass the None value to use the default system font.
- The second parameter will be the size of the font. In this example, we want the font size to be 48 points.

Demo Program: Display Text

font0.py

Go PyCharm!!!

```python
import pygame, sys
width, height = 240, 480
cycle_time    = 200

def draw_message(surface, myfont, color, position, message):
    label = myfont.render(message, 1, color)
    surface.blit(label, position)

def drawWindow(title):
    pygame.init()
    pygame.font.init()  # initialize the font module
    root = pygame.display.set_mode((width, height))
    pygame.display.set_caption(title)
    myfont = pygame.font.SysFont("Calibri", 36, True, False)
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.font.quit()   #unitialize the font module
                pygame.quit()
                sys.exit()
        draw_message(root, myfont, (255, 255, 0), (20, 100), "Hello World!")
        pygame.time.delay(cycle_time)
        pygame.display.update()
    pygame.font.quit()   #unitialize the font module
    pygame.quit()

if __name__ == "__main__":
    drawWindow("Simple Font Demo")
```

Prepare Font

Draw A Simple Label

Text Rendering

LECTURE 1

The Render Method

```
# set up the text
text = basicFont.render('Hello world!', True, WHITE,
BLUE)
textRect = text.get_rect()
```

- The Font object that we have stored in the basicFont variable has a method called render().
- This method will create a Surface object with the text drawn on it.
- The first parameter to render() is the string of the text to draw.
- The second parameter is a boolean for whether or not we want anti-aliasing.

The Render Method

```
# set up the text
text = basicFont.render('Hello world!', True, WHITE,
BLUE)
textRect = text.get_rect()
```

- The 3rd parameter is the color of the text.
- The 4th parameter is the color of the background.
- The second line creates a rectangle around the text.

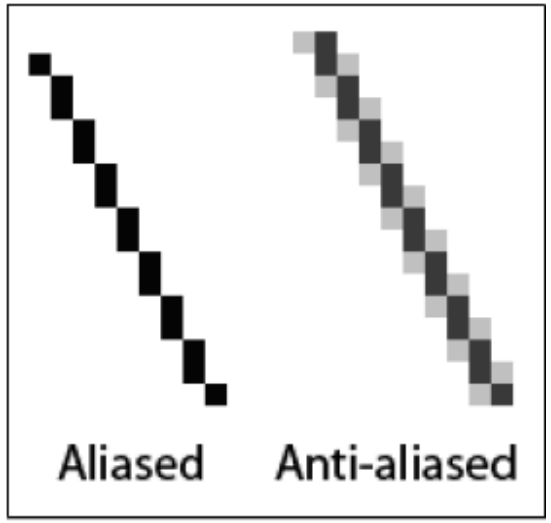
Blit the Text Onto the Surface

```
# draw the text onto the surface
windowSurface.blit(text, textRect)
```

- The blit() method will draw the contents of one Surface object onto another Surface object.
- This will draw the "Hello world!" text (which was drawn on the
- Surface object stored in the text variable) and draws it to the Surface object stored in the windowSurface variable.
- Remember that the text object had the "Hello world!" text drawn on it.

Anti-Aliasing

- Anti-aliasing is a technique for making a
- drawing look less blocky.
- Anti-aliasing can make your text and lines look blurry but smoother.
- It takes a little more computation time to do anti-aliasing, so although the graphics may look better, your program may run slower (but only just a little).



Demo Program: Display Text and Clear

font1.py

Go PyCharm!!!

Demo Program: font1.py

```
import pygame, sys
width, height = 240, 480
cycle_time    = 200

def draw_message(surface, myfont, color, position, message):
    label = myfont.render(message, 1, color)
    surface.blit(label, position)

def clear_window(surface):
    global width, height
    pygame.draw.rect(surface, (0, 0, 0), (0, 0, width, height))
```

```python
def drawWindow(title):
    pygame.init()
    pygame.font.init()  # initialize the font module
    root = pygame.display.set_mode((width, height))
    pygame.display.set_caption(title)
    myfont = pygame.font.SysFont("Calibri", 36, True, False)
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.font.quit()   #unitialize the font module
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_c:
                    clear_window(root)
                    continue
                elif event.key == pygame.K_SPACE:
                    draw_message(root, myfont, (255, 255, 0), (20, 100), "Hello World!")
                    continue
        pygame.time.delay(cycle_time)
        pygame.display.update()
    pygame.font.quit()   #unitialize the font module
    pygame.quit()

if __name__ == "__main__":
    drawWindow("Simple Font Demo")
```

Text Font Attributes

LECTURE 1

Attributes

```
textRect.centerx = windowSurface.get_rect().centerx
textRect.centery = windowSurface.get_rect().centery
```

- The pygame.Rect data type (which we will just call Rect for short) makes working with rectangle-shaped things easy.
- Use your chart!

Attributes

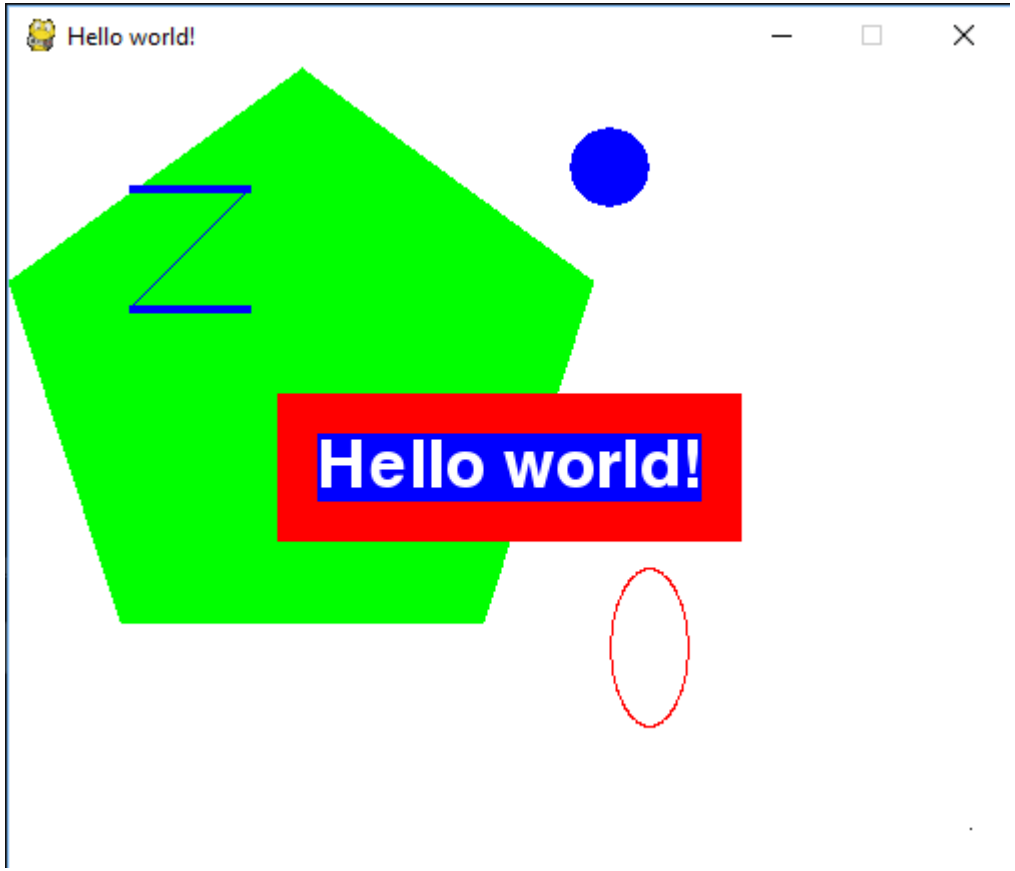
pygame.Rect Attribute	Description
`myRect.left`	The int value of the X-coordinate of the left side of the rectangle.
`myRect.right`	The int value of the X-coordinate of the right side of the rectangle.
`myRect.top`	The int value of the Y-coordinate of the top side of the rectangle.
`myRect.bottom`	The int value of the Y-coordinate of the bottom side of the rectangle.
`myRect.centerx`	The int value of the X-coordinate of the center of the rectangle.

Attributes

`myRect.centery`	The int value of the Y-coordinate of the center of the rectangle.
`myRect.width`	The int value of the width of the rectangle.
`myRect.height`	The int value of the height of the rectangle.
`myRect.size`	A tuple of two ints: (width, height)
`myRect.topleft`	A tuple of two ints: (left, top)
`myRect.topright`	A tuple of two ints: (right, top)
`myRect.bottomleft`	A tuple of two ints: (left, bottom)
`myRect.bottomright`	A tuple of two ints: (right, bottom)
`myRect.midleft`	A tuple of two ints: (left, centery)
`myRect.midright`	A tuple of two ints: (right, centery)
`myRect.midtop`	A tuple of two ints: (centerx, top)
`myRect.midbottom`	A tuple of two ints: (centerx, bottom)

Demo Program:

pygameHelloWorld.py



1. All static text/object drawing.
2. Not dynamic text drawing.

```python
import pygame, sys
from pygame.locals import *

# Set up pygame.
pygame.init()

# Set up the window.
windowSurface = pygame.display.set_mode((500, 400), 0, 32)
pygame.display.set_caption('Hello world!')

# Set up the colors.
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)

# Set up the fonts.
basicFont = pygame.font.SysFont(None, 48)

# Set up the text.
text = basicFont.render('Hello world!', True, WHITE, BLUE)
textRect = text.get_rect()
textRect.centerx = windowSurface.get_rect().centerx
textRect.centery = windowSurface.get_rect().centery

# Draw the white background onto the surface.
windowSurface.fill(WHITE)

# Draw a green polygon onto the surface.
pygame.draw.polygon(windowSurface, GREEN, ((146, 0), (291, 106), (236, 277), (56, 277), (0, 106)))
```

```python
# Draw some blue lines onto the surface.
pygame.draw.line(windowSurface, BLUE, (60, 60), (120, 60), 4)
pygame.draw.line(windowSurface, BLUE, (120, 60), (60, 120))
pygame.draw.line(windowSurface, BLUE, (60, 120), (120, 120), 4)

# Draw a red ellipse onto the surface.
pygame.draw.circle(windowSurface, BLUE, (300, 50), 20, 0)

# Draw a red ellipse onto the surface.
pygame.draw.ellipse(windowSurface, RED, (300, 250, 40, 80), 1)

# Draw the text's background rectangle onto the surface.
pygame.draw.rect(windowSurface, RED, (textRect.left - 20, textRect.top - 20, textRect.width + 40, textRect.height + 40))

# Get a pixel array of the surface.
pixArray = pygame.PixelArray(windowSurface)
pixArray[480][380] = BLACK
del pixArray

# Draw the text onto the surface.
windowSurface.blit(text, textRect)

# Draw the window onto the screen.
pygame.display.update()

# Run the game loop.
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
```

Demo Program: Display and Clear

font2.py

```
import pygame, sys
width, height = 240, 480
cycle_time = 200

def draw_message(surface, myfont, color, position, message):
    label = myfont.render(message, 1, color)
    surface.blit(label, position)
    return label.get_rect()

def clear_block(surface, x, y, w, h):
    pygame.draw.rect(surface, (0, 0, 0), (x, y, w, h))

def clear_window(surface):
    global width, height
    clear_block(surface, 0, 0, width, height)
```

```python
def drawWindow(title):
    pygame.init()
    pygame.font.init()  # initialize the font module
    root = pygame.display.set_mode((width, height))
    pygame.display.set_caption(title)
    myfont = pygame.font.SysFont("Calibri", 36, True, False)
    texton = True
    label_rect = 0
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.font.quit()  # unitialize the font module
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_SPACE:
                    if (texton):
                        label_rect = draw_message(root, myfont, (255, 255, 0), (20, 100), "Hello World!")
                        print(label_rect.left)
                        print(label_rect.top)
                        print(label_rect.width)
                        print(label_rect.height)
                        texton = not texton
                    else:
                        clear_block(root, label_rect.left+20, label_rect.top+100, label_rect.width, label_rect.height)
                        texton = not texton
                    continue
        pygame.time.delay(cycle_time)
        pygame.display.update()
    pygame.font.quit()  # unitialize the font module
    pygame.quit()

if __name__ == "__main__":
    drawWindow("Display and Clear 2")
```

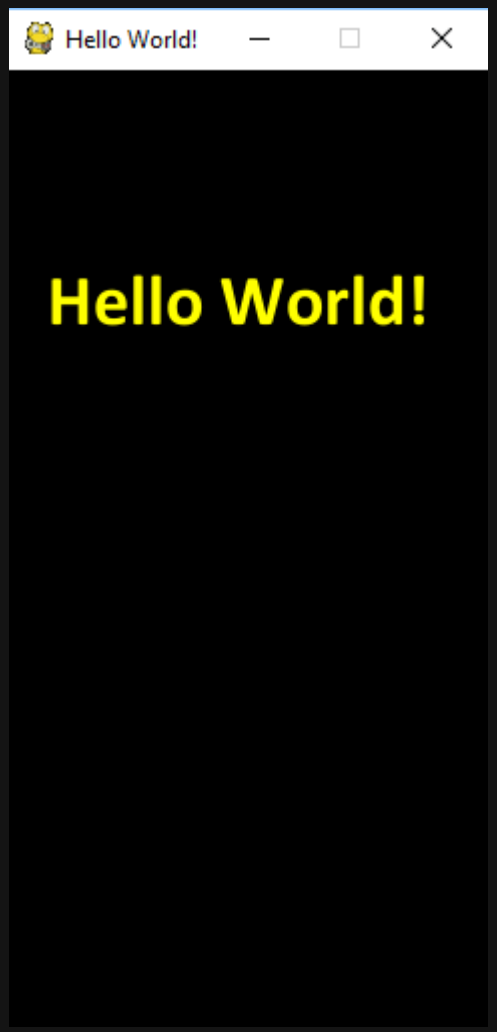
Flashing Text

LECTURE 1

Demo Program:
fontflash2.py

Go PyCharm!!!

```python
# fontflash2.py
import pygame

width, height = 240, 480

def main():
    pygame.init()
    pygame.font.init()  # initialize the font module
    root = pygame.display.set_mode((width, height))
    pygame.display.set_caption('Hello World!')

    texton = True
    myfont = pygame.font.SysFont("Calibri", 36, True, False)
    for i in range(20):
        if (texton):
            label = myfont.render("Hello World!", 1, (255, 255, 0))
            root.blit(label, (20, 100))
            texton = False
        else:
            pygame.draw.rect(root, (0, 0, 0), ((0, 0, 240, 480)))
            texton = True
        pygame.display.update()
        pygame.time.delay(200)
    pygame.font.quit()   #unitialize the font module
    pygame.quit()

if __name__ == "__main__":
    main()
```



Text Label

LECTURE 1

Demo Program: Group into Label Class

font3.py

Go PyCharm!!!

```python
import pygame # pygame_text.py  (Eric Chou)

class Label:
    def __init__(self, surface, myfont, color, position, message):
        self.surface = surface
        self.myfont = myfont
        self.color = color
        self.position = position
        self.message = message
        self.box = None
    def draw(self):
        label = self.myfont.render(self.message, 1, self.color)
        self.surface.blit(label, self.position)
        self.box = label.get_rect()
    def getBox(self):
        return self.box
    def clear(self):
        pygame.draw.rect(self.surface, (0, 0, 0),  \
                         (self.box.left+self.position[0], \
                         self.box.top+self.position[1],  \
                         self.box.width,  \
                         self.box.height) \
                        )
```

```python
def drawWindow(title):    # partial listing for font3.py
    pygame.init()
    pygame.font.init()  # initialize the font module
    root = pygame.display.set_mode((width, height))
    pygame.display.set_caption(title)
    myfont = pygame.font.SysFont("Calibri", 36, True, False)
    textonA= True
    textonB= True
    labelA = Label(root, myfont, (255, 255, 0), (20, 100), "Start Label")
    labelB = Label(root, myfont, (255, 128, 128), (20, 200), "Lower Label")
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.font.quit()  # unitialize the font module
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_SPACE:
                    if (textonA):
                        labelA.draw()
                        textonA = not textonA
                    else:
                        labelA.clear()
                        textonA = not textonA
                    continue
                elif event.key == pygame.K_b:
                    if (textonB):
                        labelB.draw()
                        textonB = not textonB
                    else:
                        labelB.clear()
                        textonB = not textonB
                    continue
        pygame.time.delay(cycle_time)
        pygame.display.update()
    pygame.font.quit()  # unitialize the font module
    pygame.quit()
```

Running Text

LECTURE 1

Running Text

1. Clear the previous location.
2. Add assign a new label body to the label holder.
3. Provide vertical running or horizontal running modes.
4. Provide looping or non-looping modes.

Demo Program:

font4.py

Go PyCharm!!!

```python
import pygame, sys
from pygame_text import *
width, height = 640, 480
cycle_time = 100

def move_label(root, myfont, color, label, x, y, vertical=True, loop=True):
    global height
    label.clear()
    if (vertical):
        if (not loop): y = y + 5
        else: y = (y+5) % height
    else:
        if (not loop): x = x + 5
        else: x = (x+5) % width
    label = Label(root, myfont, color, (x, y), label.getMessage())
    label.draw()
    return (label, x, y)

def clear_block(surface, x, y, w, h):
    pygame.draw.rect(surface, (0, 0, 0), (x, y, w, h))

def clear_window(surface):
    global width, height
    clear_block(surface, 0, 0, width, height)
```

```python
def drawWindow(title):
    pygame.init()
    pygame.font.init()  # initialize the font module
    root = pygame.display.set_mode((width, height))
    pygame.display.set_caption(title)
    myfont = pygame.font.SysFont("Calibri", 36, True, False)
    x = 20
    y = 100
    x1 = 20
    y1 = 200
    c1 = (255, 255, 0)
    c2 = (0, 128, 255)
    labelA = Label(root, myfont, c1, (x, y), "Moving")
    labelB = Label(root, myfont, c2, (x1, y1), "Label")
    moving = False
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.font.quit()  # unitialize the font module
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_SPACE:
                    moving = True
                    continue

        if (not moving):
            labelA.draw()
            labelB.draw()
        else:
            (labelA, x, y) = move_label(root, myfont, c1, labelA, x, y, True, True)
            (labelB, x1, y1) = move_label(root, myfont, c2, labelB, x1, y1, False, True)

        pygame.time.delay(cycle_time)
        pygame.display.update()
    pygame.font.quit()  # unitialize the font module
    pygame.quit()
```


Moving Labels
Label
Moving


Moving Labels
Moving
Label

Using pygame_color.py and pygame_color_simple.py

Demo Program: font5.py

```
import pygame, sys
import pygame_color_simple as colors
from pygame_text import *
width, height = 640, 480
cycle_time = 100
```

Using the colors from pygame_color_simple.py

```
    c1 = colors.yellow
    c2 = colors.uci_blue
```

Using pygame_color.py

Demo Program: font6.py

Using pygame_color.py

```
1   import pygame, sys
2   from pygame_color import *
3   from pygame_text import *
4   width, height = 640, 480
5   cycle_time = 100
37      c1 = colors['yellow']
38      c2 = colors['dodgerblue1']
```

Fonts

LECTURE 1

Figure out Fonts Available in Your System

```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40)
In[2]: from pygame import *
```

```
In[4]: font.get_fonts()
Out[4]:
['arial',
 'arialblack',
 'bahnschrift',
```

Text Button

LECTURE 1

```
Rect(object) -> Rect
```

pygame.Rect.copy	— copy the rectangle
pygame.Rect.move	— moves the rectangle
pygame.Rect.move_ip	— moves the rectangle, in place
pygame.Rect.inflate	— grow or shrink the rectangle size
pygame.Rect.inflate_ip	— grow or shrink the rectangle size, in place
pygame.Rect.clamp	— moves the rectangle inside another
pygame.Rect.clamp_ip	— moves the rectangle inside another, in place
pygame.Rect.clip	— crops a rectangle inside another
pygame.Rect.union	— joins two rectangles into one
pygame.Rect.union_ip	— joins two rectangles into one, in place
pygame.Rect.unionall	— the union of many rectangles
pygame.Rect.unionall_ip	— the union of many rectangles, in place
pygame.Rect.fit	— resize and move a rectangle with aspect ratio
pygame.Rect.normalize	— correct negative sizes
pygame.Rect.contains	— test if one rectangle is inside another
pygame.Rect.collidepoint	— test if a point is inside a rectangle
pygame.Rect.colliderect	— test if two rectangles overlap
pygame.Rect.collidelist	— test if one rectangle in a list intersects
pygame.Rect.collidelistall	— test if all rectangles in a list intersect
pygame.Rect.collidedict	— test if one rectangle in a dictionary intersects
pygame.Rect.collidedictall	— test if all rectangles in a dictionary intersect

rectObj.inflate(x_offset, y_offset) function

- rectObj.inflate(x_offset, y_offset) returns another rectangle, so you are rebinding the name 'rectObj' to the new rectangle.
- If x_offset is greater than 0, the width will be increased by x_offset.
- If y_offset is greater than 0, the height will be increased by y_offset.

Button class

- **draw()**: draw or redraw the button.
- **getBox()**: get the rect() (defined in pygame) area
  - **getBox().collidepoint(event.pos)** will check if the mouse click inside the button area.
- **connect(function)**: connect the action function with the button.
- **onClick()**: take action when the button area is clicked.

```
class Button(Label):
    def __init__(self, surface, myfont, fore_ground, back_ground, position, message,
                 padding_x=4, padding_y=4, width=0, height=0):
        super().__init__(surface, myfont, fore_ground, position, message)
        super().draw()
        self.buttonBox = self.box.inflate(padding_x, padding_y)
        self.buttonBox.centerx = position[0]+self.box.centerx
        self.buttonBox.centery = position[1]+self.box.centery
        self.fore_ground = fore_ground
        self.back_ground = back_ground
    def draw(self):
        self.surface.fill(self.back_ground, self.buttonBox)
        self.surface.blit(self.label, self.position)
    def getBox(self):
        return self.buttonBox
    def connect(self, func):
        self.func = func
    def onClick(self):
        self.func()
```

```python
import pygame, sys # font7.py
import pygame_color_simple as colors
from pygame_text import *
width, height = 640, 480
cycle_time = 100

def move_label(root, myfont, color, label, x, y, vertical=True, loop=True):
    global height
    label.clear()
    if (vertical):
        if (not loop): y = y + 5
        else: y = (y+5) % height
    else:
        if (not loop): x = x + 5
        else: x = (x+5) % width
    label = Label(root, myfont, color, (x, y), label.getMessage())
    label.draw()
    return (label, x, y)

def clear_block(surface, x, y, w, h):
    pygame.draw.rect(surface, colors.black, (x, y, w, h))

def clear_window(surface):
    global width, height
    clear_block(surface, 0, 0, width, height)

def quit_game():
    pygame.font.quit()  # unitialize the font module
    pygame.quit()
    sys.exit()
```

```python
def drawWindow(title):
    pygame.init()
    pygame.font.init()  # initialize the font module
    root = pygame.display.set_mode((width, height))
    pygame.display.set_caption(title)
    myfont  = pygame.font.SysFont("Calibri", 36, True, False)
    myfont2 = pygame.font.SysFont("Calibri", 24, True, False)
    x1 = 20
    y1 = 100
    x2 = 20
    y2 = 200
    x3 = 570
    y3 = 15
    c1 = colors.yellow
    c2 = colors.uci_blue
    c3 = colors.gray
    c4 = colors.black
    labelA  = Label(root, myfont, c1, (x1, y1), "Moving")
    labelB  = Label(root, myfont, c2, (x2, y2), "Label")
    buttonA = Button(root, myfont2, c4, c3, (x3, y3), "Quit", 30, 6)
    buttonA.connect(quit_game)
    buttonA.draw()      # default button
    moving = False
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                quit_game()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_SPACE:
                    moving = True
                    continue
            elif (event.type == pygame.MOUSEBUTTONDOWN and buttonA.getBox().collidepoint(event.pos)):
                buttonA.onClick()
        if (not moving):
            labelA.draw()
            labelB.draw()
        else:
            (labelA, x1, y1) = move_label(root, myfont, c1, labelA, x1, y1, True, True)
            (labelB, x2, y2) = move_label(root, myfont, c2, labelB, x2, y2, False, True)
        pygame.time.delay(cycle_time)
        pygame.display.update()
    pygame.font.quit()  # unitialize the font module
    pygame.quit()
```

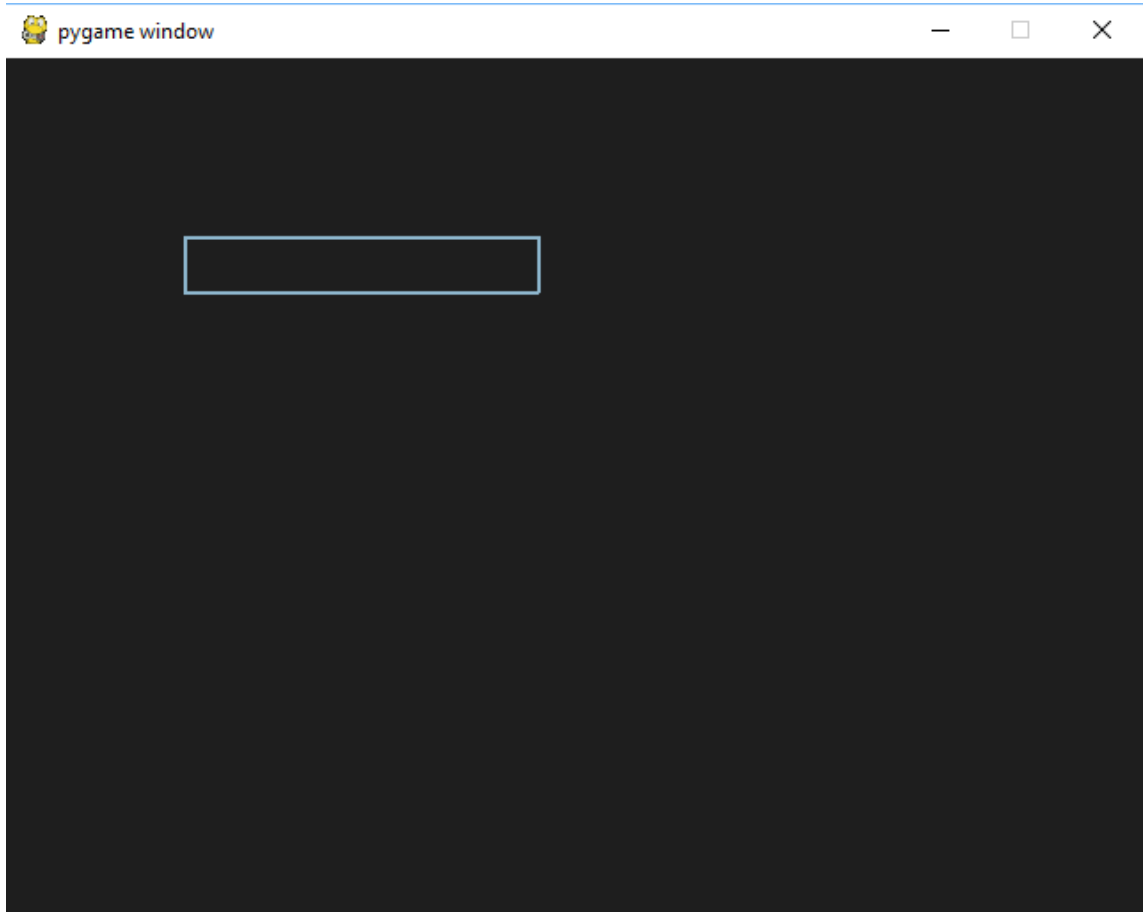
Text Input Box

LECTURE 1

Text Input (non-class type)

Demo Program: input_box1.py

Go PyCharm!!!



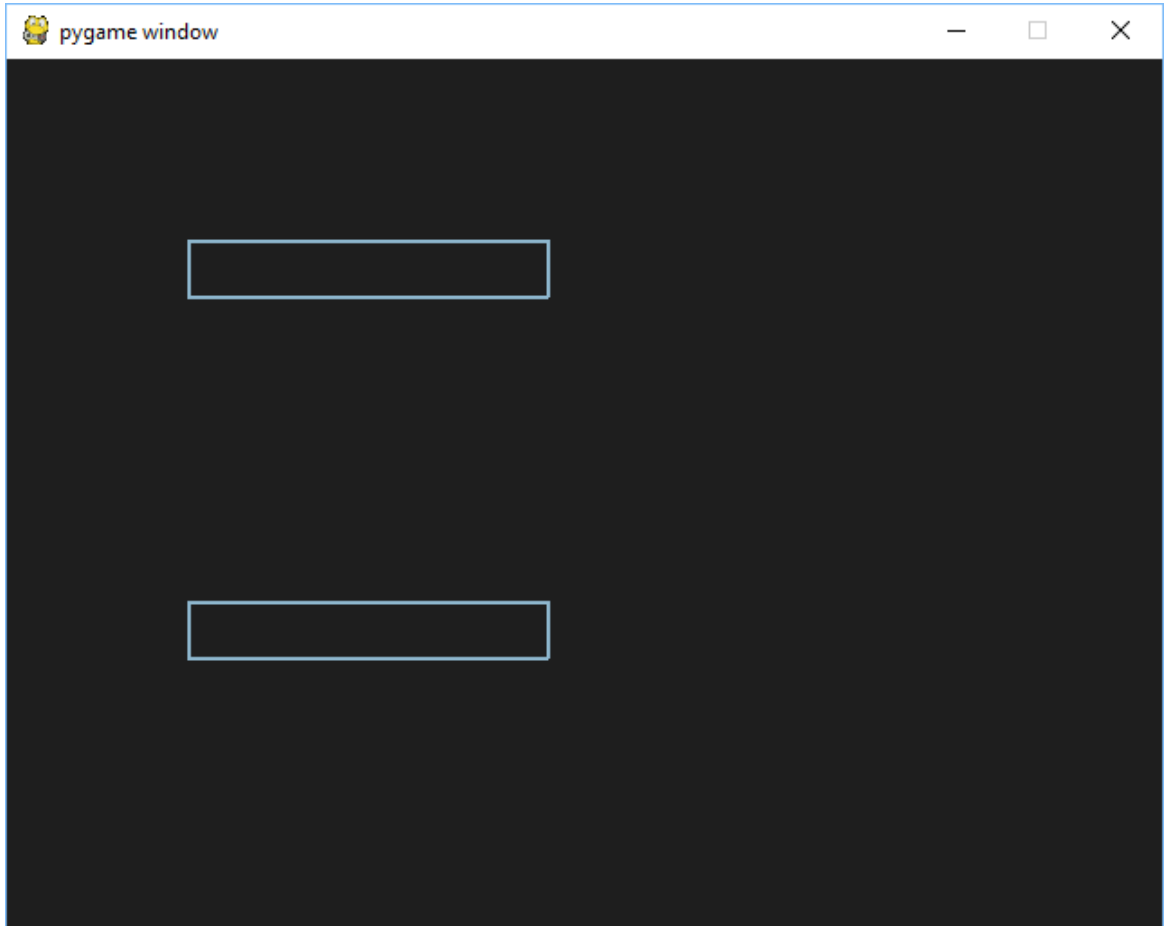
```python
import pygame

def main():
    pygame.init()
    screen = pygame.display.set_mode((640, 480))
    font = pygame.font.Font(None, 32)
    clock = pygame.time.Clock()
    input_box = pygame.Rect(100, 100, 140, 32)
    color_inactive = pygame.Color('lightskyblue3')
    color_active = pygame.Color('white')
    color = color_inactive
    active = False
    text = ''
    done = False

    while not done:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                done = True
            if event.type == pygame.MOUSEBUTTONDOWN:
                # If the user clicked on the input_box rect.
                if input_box.collidepoint(event.pos):
                    # Toggle the active variable.
                    active = not active
                else:
                    active = False
                # Change the current color of the input box.
                color = color_active if active else color_inactive
```

```
            if event.type == pygame.KEYDOWN:
             if active:
                 if event.key == pygame.K_RETURN:
                     print(text)
                     text = ''
                 elif event.key == pygame.K_BACKSPACE:
                     text = text[:-1]
                 else:
                     text += event.unicode

        screen.fill((30, 30, 30))
        # Render the current text.
        txt_surface = font.render(text, True, color)
        # Resize the box if the text is too long.
        width = max(200, txt_surface.get_width()+10)
        input_box.w = width
        # Blit the text.
        screen.blit(txt_surface, (input_box.x+5, input_box.y+5))
        # Blit the input_box rect.
        pygame.draw.rect(screen, color, input_box, 2)

        pygame.display.update()
        pygame.time.delay(30)
    pygame.quit()

if __name__ == '__main__':
    main()
```

Text Input (class InputBox)

Demo Program: input_box2.py

Go Charm!!!



```python
class InputBox:
    def __init__(self, x, y, w, h, text=''):
        self.rect = pygame.Rect(x, y, w, h)
        self.color = COLOR_INACTIVE
        self.text = text
        self.txt_surface = FONT.render(text, True, self.color)
        self.active = False
    def handle_event(self, event):
        if event.type == pygame.MOUSEBUTTONDOWN:
            # If the user clicked on the input_box rect.
            if self.rect.collidepoint(event.pos):
                # Toggle the active variable.
                self.active = not self.active
            else:
                self.active = False
            # Change the current color of the input box.
            self.color = COLOR_ACTIVE if self.active else COLOR_INACTIVE
        if event.type == pygame.KEYDOWN:
            if self.active:
                if event.key == pygame.K_RETURN:
                    print(self.text)
                    self.text = ''
                elif event.key == pygame.K_BACKSPACE:
                    self.text = self.text[:-1]
                else:
                    self.text += event.unicode
                # Re-render the text.
                self.txt_surface = FONT.render(self.text, True, self.color)
    def update(self):
        # Resize the box if the text is too long.
        width = max(200, self.txt_surface.get_width()+10)
        self.rect.w = width
    def draw(self, screen):
        # Blit the text.
        screen.blit(self.txt_surface, (self.rect.x+5, self.rect.y+5))
        # Blit the rect.
        pygame.draw.rect(screen, self.color, self.rect, 2)
```

```python
def main():
    clock = pygame.time.Clock()
    input_box1 = InputBox(100, 100, 140, 32)
    input_box2 = InputBox(100, 300, 140, 32)
    input_boxes = [input_box1, input_box2]
    done = False

    while not done:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                done = True
            for box in input_boxes:
                box.handle_event(event)

        for box in input_boxes:
            box.update()

        screen.fill((30, 30, 30))
        for box in input_boxes:
            box.draw(screen)

        pygame.display.flip()
        clock.tick(30)

if __name__ == '__main__':
    main()
    pygame.quit()
```

Text Display Project

LECTURE 1

Display Text with Button, Label and TextBox

- Label is a single Text Message.
- TextBox is a rectangle area with multiple line of text messages. Usually, it is scrollable. But in game system, it is not required to be scrollable.
- Button is a Label which can be clicked to take action.

```python
import pygame
def __pygamebox(title, message):
    pygame.quit()  # clean out anything running
    pygame.display.init()
    pygame.font.init()
    root = pygame.display.set_mode((460, 140))
    pygame.display.set_caption(title)
    font = pygame.font.Font(None, 18)
    foreg, backg, liteg = (0, 0, 0), (180, 180, 180), (210, 210, 210)
    ok = font.render('Quit', 1, foreg, liteg)
    okbox = ok.get_rect().inflate(200, 10)
    okbox.centerx = root.get_rect().centerx
    okbox.bottom = root.get_rect().bottom - 10
    root.fill(backg)
    root.fill(liteg, okbox)
    root.blit(ok, okbox.inflate(-200, -10))
    pos = [10, 10]
    for text in message.split('\n'):
        if text:
            msg = font.render(text, 1, foreg, backg)
            root.blit(msg, pos)
        pos[1] += font.get_height()
    pygame.display.flip()
    stopkeys = pygame.K_ESCAPE, pygame.K_SPACE, pygame.K_RETURN, pygame.K_KP_ENTER
    while 1:
        event = pygame.event.wait()
        if event.type == pygame.QUIT or \
                (event.type == pygame.KEYDOWN and event.key in stopkeys) or \
                (event.type == pygame.MOUSEBUTTONDOWN and okbox.collidepoint(event.pos)):
            break
    pygame.quit()

__pygamebox("Font 1", "Hello! \nHow are you?")
```

Demo Program:
font100.py

Go PyCharm!!!