Drawing Images:

```
alienPic = image.load("alien.png")          # done once
screen.blit(alienPic, (100,100,105,78))
```

Drawing images involves two things:

1) Loading the image.
The best way is to load the image in the same folder.
You could instead specify an absolute or a relative path.
Absolute is a poor idea because if you move your program you would have to change your code.

```
Absolute:    "C:\\python\\projects\\game1\\images\\alien.png"
Relative:    "images\\alien.png"
```

2) Copying the image to the screen.

BLIT is a very old term (probably a poor choice on their part).

Back in C, you would use BITBLT to copy a bitmap to the screen.

It stands for Bit Block Transfer because you were transferring a block of bits to the screen.

Everyone called it "bit-blit", and later just blit.

Blit needs the picture object (it's actually a Surface) and where it's going (a rectangle).

Sounds:

```
fireSound = mixer.Sound("fire.wav")          # done once
fireSound.play()
```

Sounds are very similar to images. Load once, play many times.

Text:

```
fontHello = font.SysFont("Times New Roman",30)  # done once

text = fontHello.render("Hello World!" , 1, (255, 0, 0))
screen.blit(text, (200,200,400,100))
```

Text is a bit of a mess.

For any font that you want to write in you need to create a Font object.

When you want to actually write you need to create a Surface (what any image is) by using font.render(), from there you blit() it like any picture.

Example#1

The following program uses the above information.

NOTE: You would need alien.png and fire.wav in the same directory as your code for this to work.

```
from pygame import *

init()
SIZE = 800, 600
screen = display.set_mode(SIZE)

fontHello = font.SysFont("Times New Roman", 30)   # Initialize a font
alienPic = image.load("alien.png")                # Load image from file
fireSound = mixer.Sound("fire.wav")               # Load shooting sound

def drawScene(screen):
    screen.fill((255,255,255))

    # Create the text and blit it on the screen (similar to images)
    text = fontHello.render("Hello World!", 1, (255,0,0))
    screen.blit(text,(200,200,400,100))

    # Draw image from file (to have a transparent background, you need to
```

```
create
    # the image as such using Photoshop or something other than MS
Paint
    # (save as .png or .gif)
    alien = (100,100,alienPic.get_width(), alienPic.get_height())
    screen.blit(alienPic, alien)

    # if mouse is pressed, then play a sound
    if mouse.get_pressed()[0]==1:
        fireSound.play()

    display.flip()


running = True

while running:
    for evnt in event.get():
        if evnt.type == QUIT:
            running = False

    # Allow the program to quit if ESC is pressed
    keys = key.get_pressed()
    if keys[K_ESCAPE]: break

    drawScene(screen)

quit()
```

Sizing Fonts

Also with fonts, it may be useful to find the size that the text will take.

For example, we had:

```
text = fontHello.render("Hello World!", 1, (255,0,0))
screen.blit(text, Rect(200,200,400,100))
```

However, if we wanted to put Hello World centered in a rectangle (200, 200, 400, 100), I could use the size to center the text in that rectangle.

Example #2

Centering text.

```
text = "Hello World"
renderedText = thisFont.render(text , 1, colour)
fontSize = thisFont.size(text) #tuple with [0] being the width, [1] the height

#setting the rectangle dimensions
rectangle = (200, 200, 400, 100)

# rectangle[2] is the width, setting the x value of where the text will start
startX = (rectangle[2] - fontSize[0])//2 + rectangle[0] #centering the text over the width

#rectangle[3] is the height, setting the y value of where the text will start
    startY = (rectangle[3] - fontSize[1])//2 + rectangle[1]

#create this new rectangle, using the size as the width and height
centeredRect = (startX, startY, fontSize[0], fontSize[1])

#drawing outline of the rectangle
draw.rect(screen, BLACK, rectangle, 2)

# blit hello world to the screen
screen.blit(renderedText, centeredRect)
```

The output would be as shown: