Pygame Part 3: A Simple Game

Objectives

- Create rectangles
- Draw rectangles onto surfaces
- Process keyboard events
- Blit images from files
- Render text images
- Play sounds and music clips
- Check when two rectangles overlap
- Detect mouse clicks
- Read mouse cursor position

In this set of worksheets we'll write a simple shooting game.
What more can I say? Let's get on with it!

The Game

Finally we're ready to send a spaceship into the box where we kept the alien bug! (Remember the cheesy bouncing_alien.py?)

Before you start coding, you want to go to our web site https://bilbo.ccis.edu/plone/Members/yliow/ccpc/fall06 and download a zipped folder containing image, sound and music files. You may use the image, sound and musci files provided. But you're welcome to use yours. There are lots of free image and sound files on the internet for writing games. Just use google to find them. I also leave the graphical design of the game up to you (screen size, etc.)

Here are the requirements of the game:

- You control a ship by moving it left and right near the bottom of the screen.
- You can fire lasers.
- There is one bug. It moves left and right near the top of the screen.
- Some bad news: the bug gets closer and closer to you by moving down a pixel randomly.
- Oh and too bad ... you don't have a shield.

You can add stuff to it later.

We'll write the program slowly, building and growing the program until all the requirements are satisfied. I will give you the solutions to all the iterations. But you should definitely spend some time trying to build your own solution for each iteration and look at the solution only when you're stuck.

Now some advice for you: Be neat. Be organized. Always backup.

I strongly suggest you have a folder for all your Python programs and experimentation. In that folder, you want to have a folder for each major project. Keep all your files for this project (program files, images files, sound files, etc.) in your project folder.

In this set of worksheets, while helping you write the game, I'll be introducing new features of pygame. Most of these will be introduced through small experimentation code. You

probably want to keep these experimentation code outside your project folder.

Game Project Part 1

Go ahead and create a simple program that opens a screen. Make sure you can close it. This is old stuff so I won't give you any help. You decide on the screen size, background color, etc.

Rects

It's been a while since we looked at the bouncing alien. You recall that an image occupies a rectangular region on the screen.

First let me show you how to create a rect. But before that, first you need a basic pygame program:

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

BLACK = (0, 0, 0)

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    pygame.display.flip()
```

We'll be adding/modifying this program in this section.

Run the program by double-clicking on the program's icon. Don't forget that errors are sent to the file stderr.txt and outputs from print statements are sent to stdout.txt. You might need to close your pygame window before opening stdout.txt and stderr.txt. You can open these two files using notepad.

First here's how you create a rect (I'll say rect instead o rectangular region from now on ... because I'm lazy :)) and then I'll print it:

```
... as before ...

BLACK = (0,0,0)
```

When you see "... as before..." it means this portion of code is similar to the previous program.

```
rect = pygame.Rect(0, 0, 5, 5)
print rect

while 1:
    ... as before ...
```

Run the program. Close it. Take a look at stdout.txt. You should see something like this:

```
<rect(0, 0, 5, 5)>
```

This is a rect whose top-left corner is at (0,0) and it has a width of 5 pixels and height of 5 pixels.

Exercise. Modify the above program to create with top-left corner at (10, 20) and width of 30 and height of 40. Run the program, close the window, and check stdout.txt. Make sure you have:

```
<rect(10, 20, 30, 40)>
```

Note that the a `rect` variable has `x`, `y`, `w`, `h`. You can think of `x` as a variable inside the variable `rect`. To access the `x` in `rect`, you say `rect.x`.

Jargon. The `x`, `y`, `w`, `h` are called **instance variables** of `rect`. The word **instance** is the same as object. This tells you that `rect` is an object.

You can actually change the values of these instance variables:

```
... as before ...

rect = pygame.Rect(10, 20, 100, 200)
print rect

rect.x = 42
rect.y = rect.y + 2
rect.w = 1
rect.h = 1000
print rect

while 1:
```

```
    ... as before ...
```

You can use the rect for simple games. You also need them for blitting images. Frequently in a game you need to make sure the "thing" on the screen stays ***in*** the screen. This means that the `x` of the rect must be positive and the right side of the rect is less than the `WIDTH` of the screen. Of course the right side is `rect.x + rect.w`. (Correct? Get it?) Because this occurs so frequently, pygame has a special instance variable for this quantity. Try this:

```
... as before ...

rect = pygame.Rect(10, 20, 100, 200)
print rect
print rect.right

while 1:
    ... as before ...
```

Do you see that `rect.right` is just `rect.x + rect.width`.

Exercise. There is also an instance variable `left`, i.e. for a rect variable `rect`, there is a variable `rect.left`. Modify the above program to print `rect.left`. What is `rect.left`?

Exercise. Repeat the above exercise with two other instance variables: `top` and `bottom`.

Exercise. Go back to good old bouncing_alien.py and look for references to the four instance variables `top`, `bottom`, `left`, and `right`. Read the code and understand how they are used.

Drawing a Rect

Here's the basic program we'll use in this section:

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

BLACK = (0, 0, 0)

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    pygame.display.flip()
```

OK ... painting a rect is ***really*** easy:

```
... as before ...

RED = (255, 0, 0)
rect = pygame.Rect(10, 10, 100, 200)
pygame.draw.rect(surface, RED, rect)

while 1:
    ... as before ...
```

Exercise. Try to figure out what's happening here:

```
... as before ...

rect = pygame.Rect(10, 10, 100, 200)
pygame.draw.rect(surface, RED, rect, 5)

while 1:
    ... as before ...
```

Moving a Rect

There's already a move function for rects. Try this and see if you understand what's happening:

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

BLACK = (0, 0, 0)

rect = pygame.Rect(5, 8, 10, 20)
print rect
change = [1, 2]
rect = rect.move(change)
print rect

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    pygame.display.flip()
```

Note that this `move` changes both the x and y value of the rect. Of course you can also manipulate the x and y values of the rect directly.

OK. Try this simply animation:

```
... as before ...

BLACK = (0, 0, 0)
RED = (255, 0, 0)

rect = pygame.Rect(0, 0, 20, 10)
speed = [1, 0]
```

```
while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    surface.fill(BLACK)

    rect = rect.move(speed)
    pygame.draw.rect(surface, RED, rect)

    pygame.display.flip()
```

Oops!!! The red brick disappeared! Why? Because it keeps going to the right even when it hits the right wall.

Exercise. Modify the above program so that when the red brick hits the right wall, it stops there.

Image and Rects

Most of this section is just a review.

Using the bouncing_alien.py as an example, here's a simple program that loads and blits an image onto a rectangular region.

```
import sys, pygame
pygame.init()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

BLACK = (0, 0, 0)

image = pygame.image.load("GalaxianAquaAlien.gif")
rect = image.get_rect()
print rect
surface.blit(image, rect)

while 1:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    pygame.display.flip()
```

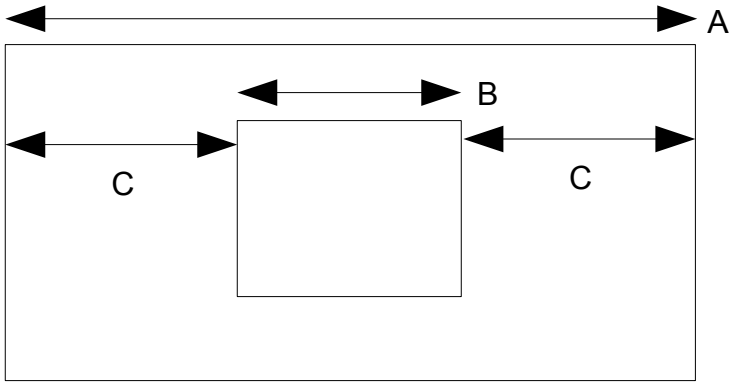
The only thing I will add is that the image variable can give you its own rect. Look at the statement:

```
rect = image.get_rect()
```

(You can of course go to your image file and check the width and height, and then create the rect in your code. But that's unnecessary.)

Note that `rect.w` and `rect.h` give the width and height of the image. `rect.x` and `rect.y` are both zero.

Exercise. By changing the `rect.x` and `rect.y`, blit the image in the middle of the screen. There a little bit of math here. Here's the hint: Suppose you have the following rects:



where the width of the bigger one is A and the width of the smaller one is B. Suppose the smaller rect is in the exact middle of the larger one. What is the length of C?

Why is this relevant? Because the larger rect is our surface, the smaller rect is the rect to be centered. To center the rect, you need to change the x value of the rect and the y value of the rect. The above diagram tells you the correct value of x for the centered rect. Use the same idea for the y value and you will be able to center the rect in the surface.

Here's something new: You can actually blit a ***portion*** of your image. Try this:

```
import sys, pygame
pygame.init()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

BLACK = (0, 0, 0)

image = pygame.image.load("GalaxianAquaAlien.gif")
rect = image.get_rect()
print rect
rect2 = pygame.Rect(10, 10, 100, 100)
print rect2
surface.blit(image, rect, rect2)

while 1:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    pygame.display.flip()
```

In the above example, I blit the rect2 portion of the image. Let me explain. Suppose you have a 8-by-8 image and you only want to blit the right half of the image:

				x	x	x	x
				x	x	x	x
				x	x	x	x
				x	x	x	x
				x	x	x	x
				x	x	x	x
				x	x	x	x
				x	x	x	x

(I've marked the portion of the image I want to blit with x's.) The rect with x's is (4,0,4,8). So this is how to blit it:

```
rect2 = pygame.Rect(4, 0, 4, 8)
surface.blit(image, rect, rect2)
```

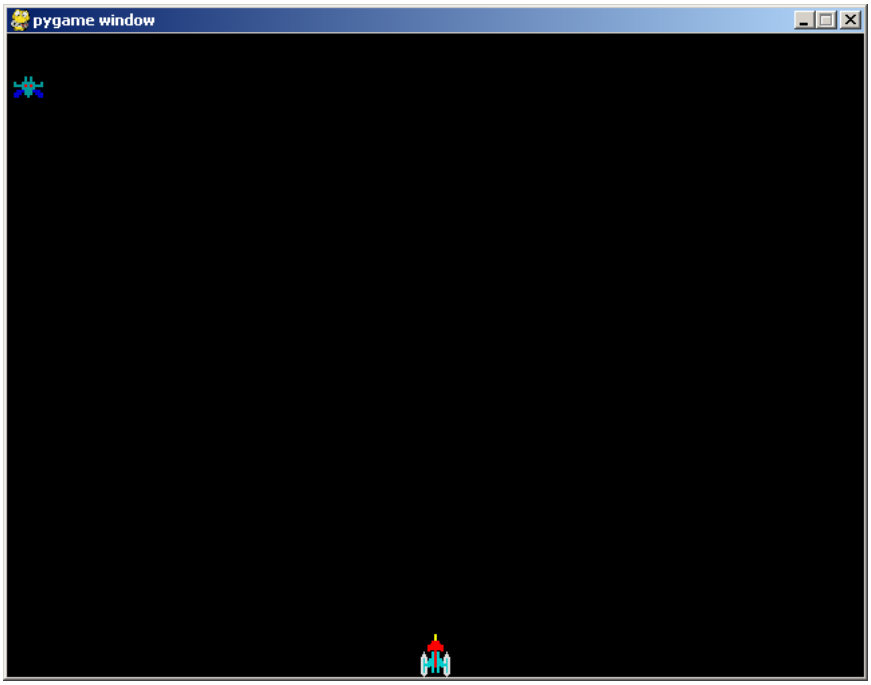
Note that there are two rects: `rect` is the rectangular region on the surface you want to blit to and `rect2` is the portion of the image you want to blit.

Game Project Part 2

Now draw the alien bug (with “GalaxianAquaAlien.gif”) and your ship (with “GalaxianGalaxip.gif”).

You should keep some space near the top for the score.

Your ship should appear near the bottom and in the middle of the of the screen. The alien appear near the left wall just below the area for the score. Everything should be stationary for the time being:



OK. Let's think about it. What variables do you need? You need the images for the alien and the flagship. Of course they need to have their positions, i.e. their rects.

Organize your variables! For me, I'm going to use the following variable names:

- `alien_image`
- `alien_rect`
- `ship_image`
- `ship_rect`

Of course I have variables `WIDTH` and `HEIGHT`. I'm reserving the a rect of height 24 and width `WIDTH` for the score. I'll use

a variable `SCORE_HEIGHT` for this height, i.e. 24.

The answer is on the next page ... but you should try your best to make yours work without referring to the solution.

Solution to Game Project Part 2

```
import pygame, sys
pygame.init()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

SCORE_HEIGHT = 24
BLACK = (0, 0, 0)

sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

# Create alien
alien_image = pygame.image.load("GalaxianAquaAlien.gif")
alien_rect = alien_image.get_rect()
alien_rect = alien_rect.move([0,SCORE_HEIGHT])

# Create flagship
ship_image = pygame.image.load("GalaxianGalaxip.gif")
ship_rect = ship_image.get_rect()
x = (WIDTH - ship_rect.w) / 2
y = HEIGHT - ship_rect.h
ship_rect = ship_rect.move([x,y])

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    surface.fill(BLACK)

    surface.blit(alien_image, alien_rect)
    surface.blit(ship_image, ship_rect)

    pygame.display.flip()
    pygame.time.delay(100)
```

Game Project Part 3

Now make the alien move left and right within the screen. This should be easy. Your ship is still stationary.

Make sure it works.

Next randomly make the alien move down a pixel. Let's just say that roughly in 1 out of 100 moves it moves down by one pixel. This is how you can achieve that: Each time you move the alien, generate a random number between 0 and 99. If the number is 0, make the alien move down by 3 pixel.

Instead of coding 3 directly into the code, I'm going to create a constant variable `ALIEN_Y_INCREMENT` setting it to 3.

(If you want to be bad to the player, you can make the alien move by more pixels when it moves down.)

Note that at some point the alien will “pass through” the player's ship: There is no collision yet. That's OK. We'll do that later.

You also need not worry about the alien going “through” the bottom of the screen.

Solution to Part 3

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

SCORE_HEIGHT = 24
BLACK = (0, 0, 0)

sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

# Create alien
alien_image = pygame.image.load("GalaxianAquaAlien.gif")
alien_rect = alien_image.get_rect()
alien_rect = alien_rect.move([0,SCORE_HEIGHT])
alien_speed = [1, 0]
ALIEN_Y_INCREMENT = 3

# Create flagship
ship_image = pygame.image.load("GalaxianGalaxip.gif")
ship_rect = ship_image.get_rect()
x = (WIDTH - ship_rect.w) / 2
y = HEIGHT - ship_rect.h
ship_rect = ship_rect.move([x,y])

def move(d, v, m):
    d = d + v
    if d < 0:
        d = 0
        v = -v
    elif d > m:
        d = m
        v = -v
    return d, v

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    surface.fill(BLACK)

    # Move alien horizontally
    alien_rect.x, alien_speed[0] = move(alien_rect.x, \
                                alien_speed[0], WIDTH - alien_rect.w)

    # Probabilistically move alien vertically down
```

```
    if random.randrange(100) == 0:
        alien_rect.y = alien_rect.y + ALIEN_Y_INCREMENT

    surface.blit(alien_image, alien_rect)
    surface.blit(ship_image, ship_rect)

    pygame.display.flip()
```

On to collision!!!

How to do Basic Collision Detection

When do things collide?

In the real world, this happens when the space occupy by two things overlap.

In the case of games, since things move quickly and we don't have perfect vision, usually it's good enough to say that two things collide when the rects of their images overlap.

How do you tell when two rects collide? With pygame that's easy. (Actually this is easy even without pygame; it's just some simple math).

Suppose `rect` and `rect2` are two rects. Then calling `rect.colliderect(rect2)` will return `1` if the two rects `rect` and `rect2` overlap. Otherwise it will return `0`. Don't forget that `1` is like `True` while `0` is like `False`.

Try this example:

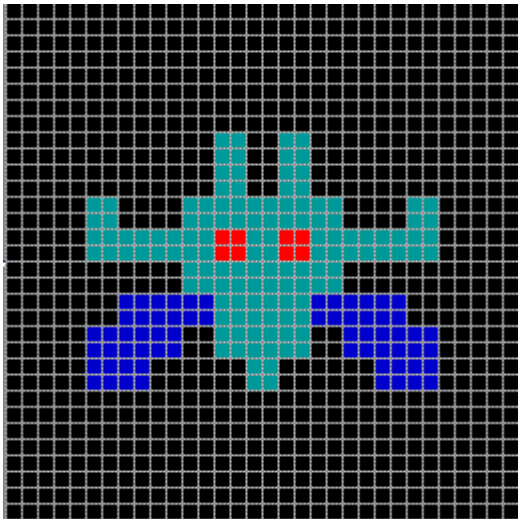
```
import pygame, sys
pygame.init()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

BLACK = (0,0,0)
RED = (255,0,0)

rect = pygame.Rect(0,0,20,10)
rect2 = pygame.Rect(5,5,10,10)
print rect.colliderect(rect2)

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
```

```
    pygame.display.flip()
```

There are two rects. First check (in your head) that `rect` and `rect2` overlap each other. Next run the program. Close it. Check the output in stdout.txt – you should get `1`.

Exercise. Using the above example, create two ***non-overlapping*** rects, run the program, check the output. Make sure you get `0`.

You should remember that the rect can be much larger than the region actually occupied by the image. For instance if you look at “GalaxianAquaAlien.gif” you will see that the actual alien is smaller than the whole image file (alien and the background):



This means that the rects of two images can collide even though the images (excluding the backgrounds) do not really overlap. But if the game is moving fast enough, the player(s) can't really tell the difference.

Game Project Part 4

Now let's make the program know when the alien collides with the ship. (Sorry ... you can't defend yourself yet .. you can't move and you can't fire lasers ... :))

For the time being, when there is a collision, just make the alien stop moving.

You need to remember when NOT to move the alien when there is a collision. So just create a variable to remember if you have a collision.

Solution to Game Project Part 4

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

SCORE_HEIGHT = 24
BLACK = (0, 0, 0)

sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

# Create alien
alien_image = pygame.image.load("GalaxianAquaAlien.gif")
alien_rect = alien_image.get_rect()
alien_rect = alien_rect.move([0,SCORE_HEIGHT])
alien_speed = [1, 0]

# Create flagship
ship_image = pygame.image.load("GalaxianGalaxip.gif")
ship_rect = ship_image.get_rect()
x = (WIDTH - ship_rect.w) / 2
y = HEIGHT - ship_rect.h
ship_rect = ship_rect.move([x,y])

def move(d, v, m):
    d = d + v
    if d < 0:
        d = 0
        v = -v
    elif d > m:
        d = m
        v = -v
    return d, v

collides = False
while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    surface.fill(BLACK)

    if collides == False:
        alien_rect.x, alien_speed[0] = move(alien_rect.x, \
                                    alien_speed[0], WIDTH - alien_rect.w)
        if random.randrange(100) == 0:
            alien_rect.y = alien_rect.y + 3
```

```
    collides = alien_rect.colliderect(ship_rect)

    surface.blit(alien_image, alien_rect)
    surface.blit(ship_image, ship_rect)

    pygame.display.flip()
```

On to moving and firing your ship!!!

Keyboard

You know that when you write a console program (remember the black text based screen?) you read the keyboard with `raw_input` and `input`.

When you have a pygame program, keyboard input is slightly differently.

When you have an input (this includes key press, mouse movement, etc.) an **event** is created and stored somewhere (never mind where). You have to remove the event from where it's stored and check what kind of event it is and process it accordingly in your program.

Now look at this code segment from the previous program::

```
...
while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    ...
```

This tells you that `pygame.event.get()` is a list – it's a list of all the events that's currently stored based on user input. The `for`-loop runs across all the events with the variable `event`. In the body of the for-loop, you check the type of the event and make the program act accordingly. In the above code, you halt the program if the event is one where the user closes the program. If it's not an event that associated with closing a program, then nothing is done – that event is ignored.

All that is old stuff. Now for something new ...

Look at this program:

```
import pygame, sys
pygame.init()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)
```

```
sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

BLACK = (0,0,0)
RED = (255,0,0)

rect = pygame.Rect(0,0,20,10)

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        if event.type == pygame.KEYDOWN:
            pygame.draw.rect(surface, RED, rect)

    pygame.display.flip()
```

Save it and run it. Press any key on your keyboard. What do you see? Check the code.

The program shows you how to detect a key press.

One thing you should realize is that when you press a key, a ***single*** keydown event is created. If you continue holding the key down, no extra keydown event is created even if you hold it for a minute.

Run this program and hold a key down for a 10 seconds. How many rects are drawn?

```
... as before ...

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        if event.type == pygame.KEYDOWN:
            x = random.randrange(640)
            y = random.randrange(480)
            rect = pygame.Rect(x,y,20,10)
            pygame.draw.rect(surface, RED, rect)

    pygame.display.flip()
```

To get your computer to generate extra keydown events when a key is held down, you have to set the repeat rate for the keyboard. You also have to set the repeat delay. Run the

following example and hold a key down:

```
... as before ...

pygame.key.set_repeat(1000, 100)

while 1:

    ... as before ...
```

After you press a key, a keydown event is created. If you continue to hold the key down, after 1000 milliseconds (i.e. 1 second), new keydown events are created at the rate of 1 event for every 100 milliseconds.

OK. That was easy. BUT ... you only know that a key was pressed. That's not enough. Frequently you need to know ***which*** key was pressed.

In the next program, I will draw a red rect when the left arrow key is pressed and a blue rect when the right arrow key is pressed. When the key pressed is neither the left arrow nor the right arrow key, nothing happens.

OK. Enough talking. Try this out:

```
... as before ...

BLACK = (0, 0, 0)
RED = (255, 0, 0)
BLUE = (0, 0, 255)

pygame.key.set_repeat(1000, 100)

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        if event.type == pygame.KEYDOWN:
            x = random.randrange(640)
            y = random.randrange(480)
            rect = pygame.Rect(x, y, 20, 10)

            keypressed = pygame.key.get_pressed()
            if keypressed[pygame.K_LEFT]:
                color = RED
```

```
            if keypressed[pygame.K_RIGHT]:
                color = BLUE

            pygame.draw.rect(surface, color, rect)

    pygame.display.flip()
```

The `pygame.key.get_pressed()` checks all the keys on your keyboard and returns a ***tuple of boolean values*** (`True` or `False`). The variables `pygame.K_LEFT` and `pygame.K_RIGHT` are constant variables that act as index values into the `keypressed` tuple. `keypressed[pygame.K_LEFT]` gives you `True` if the left arrow key was pressed.

Exercise. Here's a quick review of tuples:

```
t = (True, True, False)
print t[0], t[2]
if t[1]:
    print "t is True at index 1"
```

Here's a list of all the constant variable names for other keys:

```
Variable name Key
------------- -----------
K_BACKSPACE   backspace
K_TAB         tab
K_RETURN      return
K_PAUSE       pause
K_ESCAPE      escape
K_SPACE       space
K_0           0
K_1           1
K_2           2
K_a           a
K_b           b
K_c           c
K_d           d
K_DOWN        down arrow
K_RIGHT       right arrow
K_LEFT        left arrow
K_F1          F1
K_F2          F2
```

Exercise. Modify the above program by adding the following

feature: so that if the spacebar is pressed, a green rect is drawn at a random location.

Exercise. When you have time, go to pygame's web site and look for a list of ***all*** the constant variable names.

Game Project Part 5

Now make your ship move left and right ... ***within the screen of course***!!!

The left arrow key moves the ship left by 1 pixel and the right arrow key moves the ship right.

Don't forget ... the ship (or rather the rect of the ship's image) must stay within the screen.

I know ... I know ... you can't fire and it looks miserable moving your ship left and right helplessly. Lasers are coming soon.

Solution to Game Project Part 5

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

pygame.key.set_repeat(10, 10)

SCORE_HEIGHT = 24
BLACK = (0, 0, 0)

sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

# Create alien
alien_image = pygame.image.load("GalaxianAquaAlien.gif")
alien_rect = alien_image.get_rect()
alien_rect = alien_rect.move([0,SCORE_HEIGHT])
alien_speed = [1, 0]

# Create flagship
ship_image = pygame.image.load("GalaxianGalaxip.gif")
ship_rect = ship_image.get_rect()
x = (WIDTH - ship_rect.w) / 2
y = HEIGHT - ship_rect.h
ship_rect = ship_rect.move([x,y])
ship_speed = [0, 0]

def move(d, v, m):
    d = d + v
    if d < 0:
        d = 0
        v = -v
    elif d > m:
        d = m
        v = -v
    return d, v

collides = False
while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            keypress = pygame.key.get_pressed()
            if keypress[pygame.K_LEFT]:
                ship_speed = [-1, 0]
            elif keypress[pygame.K_RIGHT]:
```

```
                ship_speed = [1, 0]

    surface.fill(BLACK)

    if collides == False:
        alien_rect.x, alien_speed[0] = move(alien_rect.x,\
                                alien_speed[0], WIDTH - alien_rect.w)
        if random.randrange(100) == 0:
            alien_rect.y = alien_rect.y + ALIEN_Y_INCREMENT

        ship_rect.x, ship_speed[0] = move(ship_rect.x, ship_speed[0],\
                                    WIDTH - ship_rect.w)
        ship_speed = [0, 0]

    collides = alien_rect.colliderect(ship_rect)

    surface.blit(alien_image, alien_rect)
    surface.blit(ship_image, ship_rect)

    pygame.display.flip()
```

Now for ..... ***LASERS!!!***

Game Project Part 6

Now add a single laser (... DON'T BE GREEDY! ... start small ...) for your ship to use.

Here's how you should do it.

We're going to draw the laser as a red-colored rect. (Poor man's laser???)

First of all, note that initially the laser is not fired – it's not drawn. It's only when you press the spacebar, then the laser starts moving.

That means that we need a variable for the laser. The variable tells us whether the laser is to be moved and to be drawn. And in fact whether we should check for its collision with the alien. I'm calling my variable `laser_alive`. If `laser_alive` is `False`, it's not present in the screen. If `laser_alive` is `True`, then it ***IS*** present in the screen. (You can also set the laser's speed to [0,0] when it's not moving and use that to detect that the laser is not present on the screen.)

You get to choose the speed of the laser. Of course the laser move straight up. (After pressing the spacebar, where should the laser's rect be? Think about it ...)

Don't forget that while a laser is in motion, you cannot fire another. Furthermore when the laser reaches the top (the bottom of the score area), the laser should disappear – i.e. the `laser_alive` should be set to `False` – so that the laser can be fired again from the ship.

Solution to Game Project Part 5

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

pygame.key.set_repeat(10, 10)

SCORE_HEIGHT = 24
BLACK = (0, 0, 0)
RED = (255, 0, 0)
sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

# Create alien
alien_image = pygame.image.load("GalaxianAquaAlien.gif")
alien_rect = alien_image.get_rect()
alien_rect = alien_rect.move([0,SCORE_HEIGHT])
alien_speed = [1, 0]
ALIEN_Y_INCREMENT = 3

# Create flagship
ship_image = pygame.image.load("GalaxianGalaxip.gif")
ship_rect = ship_image.get_rect()
x = (WIDTH - ship_rect.w) / 2
y = HEIGHT - ship_rect.h
ship_rect = ship_rect.move([x,y])
ship_speed = [0, 0]

# Ship's laser
laser_rect = pygame.Rect(0, 0, 4, 8)
laser_speed = [0, 0]
laser_alive = False

def move(d, v, m):
    d = d + v
    if d < 0:
        d = 0
        v = -v
    elif d > m:
        d = m
        v = -v
    return d, v

collides = False
while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
```

```
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            keypress = pygame.key.get_pressed()
            if keypress[pygame.K_LEFT]:
                ship_speed = [-1, 0]
            elif keypress[pygame.K_RIGHT]:
                ship_speed = [1, 0]
            elif keypress[pygame.K_SPACE]:
                if not laser_alive:
                    laser_alive = True
                    laser_speed = [0, -2]
                    laser_rect.x = ship_rect.x + \
                                   (ship_rect.w - laser_rect.w) / 2
                    laser_rect.y = ship_rect.y - laser_rect.w

    surface.fill(BLACK)

    if collides == False:
        alien_rect.x, alien_speed[0] = move(alien_rect.x, \
                                alien_speed[0], WIDTH - alien_rect.w)
        if random.randrange(100) == 0:
            alien_rect.y = alien_rect.y + ALIEN_Y_INCREMENT

        ship_rect.x, ship_speed[0] = move(ship_rect.x, ship_speed[0],\
                                  WIDTH - ship_rect.w)
        ship_speed = [0, 0]

        if laser_alive:
            laser_rect.y, laser_speed[1] = move(laser_rect.y, \
                                laser_speed[1], HEIGHT - laser_rect.h)
            if laser_rect.y < SCORE_HEIGHT:
                laser_alive = False

    collides = alien_rect.colliderect(ship_rect)

    surface.blit(alien_image, alien_rect)
    surface.blit(ship_image, ship_rect)
    if laser_alive:
        pygame.draw.rect(surface, RED, laser_rect)
    pygame.display.flip()
```

Yoohooooo!!!

Now to get the laser to collide with the alien ...

Game Project Part 6

Now modify your program so that everything stops when either the alien collide with your ship or when your laser collides with the alien. This should be easy.

Don't worry about keeping score.

Solution to Game Project Part 6

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

pygame.key.set_repeat(10, 10)

SCORE_HEIGHT = 24
BLACK = (0, 0, 0)
RED = (255, 0, 0)
sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

# Create alien
alien_image = pygame.image.load("GalaxianAquaAlien.gif")
alien_rect = alien_image.get_rect()
alien_rect = alien_rect.move([0,SCORE_HEIGHT])
alien_speed = [1, 0]
ALIEN_Y_INCREMENT = 3

# Create flagship
ship_image = pygame.image.load("GalaxianGalaxip.gif")
ship_rect = ship_image.get_rect()
x = (WIDTH - ship_rect.w) / 2
y = HEIGHT - ship_rect.h
ship_rect = ship_rect.move([x,y])
ship_speed = [0, 0]

# Ship's laser
laser_rect = pygame.Rect(0, 0, 4, 8)
laser_speed = [0, 0]
laser_alive = False

def move(d, v, m):
    d = d + v
    if d < 0:
        d = 0
        v = -v
    elif d > m:
        d = m
        v = -v
    return d, v

collides = False
while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
```

```
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            keypress = pygame.key.get_pressed()
            if keypress[pygame.K_LEFT]:
                ship_speed = [-1, 0]
            elif keypress[pygame.K_RIGHT]:
                ship_speed = [1, 0]
            elif keypress[pygame.K_SPACE]:
                if not laser_alive:
                    laser_alive = True
                    laser_speed = [0, -2]
                    laser_rect.x = ship_rect.x + \
                                   (ship_rect.w - laser_rect.w)/2
                    laser_rect.y = ship_rect.y - laser_rect.w

    surface.fill(BLACK)

    if collides == False:
        alien_rect.x, alien_speed[0] = move(alien_rect.x, \
                                alien_speed[0], WIDTH - alien_rect.w)
        if random.randrange(100) == 0:
            alien_rect.y = alien_rect.y + ALIEN_Y_INCREMENT

        ship_rect.x, ship_speed[0] = move(ship_rect.x, ship_speed[0],\
                                    WIDTH - ship_rect.w)
        ship_speed = [0, 0]

        if laser_alive:
            laser_rect.y, laser_speed[1] = move(laser_rect.y, \
                               laser_speed[1], HEIGHT - laser_rect.h)
            if laser_rect.y < SCORE_HEIGHT:
                laser_alive = False

    collides = alien_rect.colliderect(ship_rect) or \
               laser_rect.colliderect(alien_rect)

    surface.blit(alien_image, alien_rect)
    surface.blit(ship_image, ship_rect)
    if laser_alive:
        pygame.draw.rect(surface, RED, laser_rect)
    pygame.display.flip()
```

Now for some sounds ... explosion and all that!

Sound and music

Recall quickly playing sound (you have already seen this) ...

```
import pygame, sys
pygame.init()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

pygame.mixer.music.load("gameplay.mid")
music_on = False

sound1 = pygame.mixer.Sound("gameover.wav")
sound2 = pygame.mixer.Sound("start.wav")

sound1.play()
sound2.play()

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    pygame.display.flip()
    pygame.time.delay(100)
```

No big deal.

Now for music. In Pygame you can only play ***one*** music file at a time. Notice in the above program, you can play two sound files at the same time. There are many effects you can have for music files. I'll just show you how to play and stop the music.

```
import pygame, sys
pygame.init()

sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)
```

```
pygame.mixer.music.load("demo.mid")

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if event.type == pygame.KEYDOWN:
            keypressed = pygame.key.get_pressed()
            if keypressed[pygame.K_LEFT]:
                pygame.mixer.music.play(-1)
            if keypressed[pygame.K_RIGHT]:
                pygame.mixer.music.stop()
```

In this program, the music will start playing when you press the left key and it will stop with you press the right key.

Note that when press the left arrow when the music is playing, the music stops playing and then is restarted. In other words you can only play one music file at one time.

Game Project Part 7

Now add explosion to your game whenever there is a collision. You may use the file gexplode.wav.

Also, play a laser sound when the laser is fire. You may use the file laser.wav.

Hint: You do not want to play the explosion again and again and again ... You need to remember if the explosion has been played – you need a variable to remember that.

Solution to Game Project Part 7

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

pygame.key.set_repeat(10, 10)

SCORE_HEIGHT = 24
BLACK = (0, 0, 0)
RED = (255, 0, 0)
sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

# Create alien
alien_image = pygame.image.load("GalaxianAquaAlien.gif")
alien_rect = alien_image.get_rect()
alien_rect = alien_rect.move([0,SCORE_HEIGHT])
alien_speed = [1, 0]
ALIEN_Y_INCREMENT = 3

# Create flagship
ship_image = pygame.image.load("GalaxianGalaxip.gif")
ship_rect = ship_image.get_rect()
x = (WIDTH - ship_rect.w) / 2
y = HEIGHT - ship_rect.h
ship_rect = ship_rect.move([x,y])
ship_speed = [0, 0]

# Ship's laser
laser_rect = pygame.Rect(0, 0, 4, 8)
laser_speed = [0, 0]
laser_alive = False

# Laser sound
laser_sound = pygame.mixer.Sound("laser.wav")

# Explosion sound
explosion_sound = pygame.mixer.Sound("gexplode.wav")
explosion_played = False

def move(d, v, m):
    d = d + v
    if d < 0:
        d = 0
        v = -v
    elif d > m:
        d = m
        v = -v
```

```
    return d, v

collides = False
while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            keypress = pygame.key.get_pressed()
            if keypress[pygame.K_LEFT]:
                ship_speed = [-1, 0]
            elif keypress[pygame.K_RIGHT]:
                ship_speed = [1, 0]
            elif keypress[pygame.K_SPACE]:
                if laser_alive == False:
                    laser_alive = True
                    laser_speed = [0, -2]
                    laser_rect.x = ship_rect.x + \
                                   (ship_rect.w - laser_rect.w)/2
                    laser_rect.y = ship_rect.y - laser_rect.w
                    laser_sound.play()

    surface.fill(BLACK)

    if not collides:
        alien_rect.x, alien_speed[0] = move(alien_rect.x, \
                                      alien_speed[0], WIDTH - alien_rect.w)
        if random.randrange(100) == 0:
            alien_rect.y = alien_rect.y + ALIEN_Y_INCREMENT

        ship_rect.x, ship_speed[0] = move(ship_rect.x, ship_speed[0], \
                                     WIDTH - ship_rect.w)
        ship_speed = [0, 0]

        if laser_alive:
            laser_rect.y, laser_speed[1] = move(laser_rect.y, \
                                       laser_speed[1], HEIGHT - laser_rect.h)
            if laser_rect.y < SCORE_HEIGHT:
                laser_alive = False

    collides = alien_rect.colliderect(ship_rect) or \
               laser_rect.colliderect(alien_rect)

    if collides == True and explosion_played == False:
        explosion_sound.play()
        explosion_played = True

    surface.blit(alien_image, alien_rect)
    surface.blit(ship_image, ship_rect)
    if laser_alive:
        pygame.draw.rect(surface, RED, laser_rect)
    pygame.display.flip()
```

Text

Drawing text onto your surface is pretty easy. You basically create an image in your program – it's not loaded from a file – and blit it.

To create the image, you need to create a font variable, specify what kind of font and size you want. While creating the image, you need to specify the color of the text image. For our example, I will only use the default font.

Here's an example. Run it.

```
import pygame, sys
pygame.init()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

WHITE = (255, 255, 255)

font = pygame.font.Font(None, 100)
image = font.render("hello world", 1, WHITE)
rect = image.get_rect()
surface.blit(image, rect)

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    pygame.display.flip()
```

Look at this statement:

```
font = pygame.font.Font(None, 100)
```

The `None` tells Python to use the default font. (You can change it for instance to Courier font). The 100 tells Python that you want your font size to be 100. Now for the statement:

```
image = font.render("hello world", 1, WHITE)
```

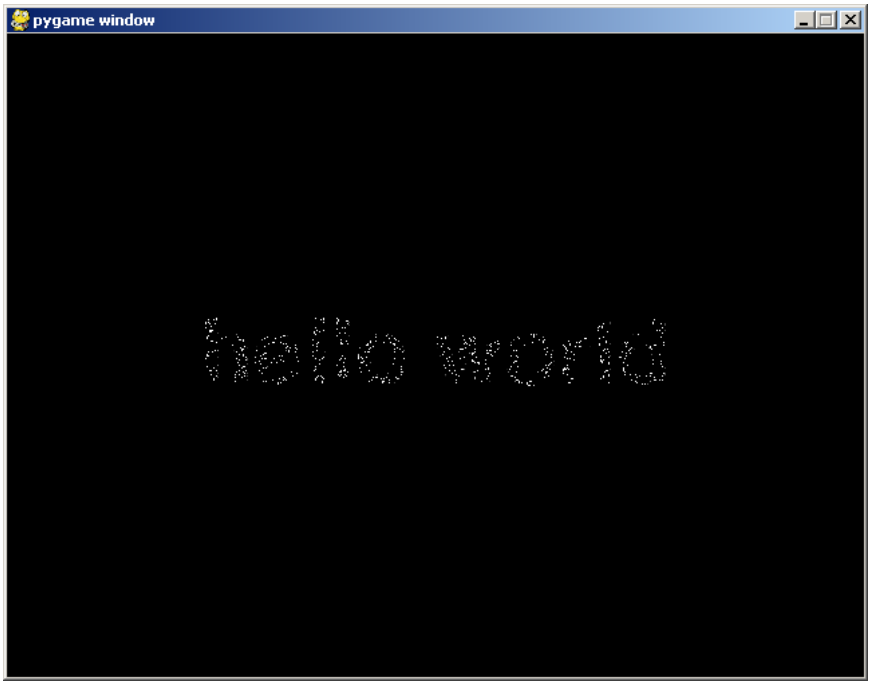
Everything is clear except for the `1`. The `1` tells Python to "smooth" the edges of the image; 0 tells Python not to smooth the edges.

Exercise. Modify the program to display your name in a color of your choice.

Exercise. Recall that you can blit a portion of the image. Write a wipe-across hello world: When you run the program, you see the hello world wiped across from left to right:

(You can choose any text and any color.)

Exercise. Repeat the above so that the text appear pixel-by-pixel at random places on the text image:



Exercise. (Optional ... and challenging) And of course you have to try the falling characters in the Matrix movie!!!

Exercise. Modify the above program so that the text bounces in the screen and its color changes smoothly. (Much, much, ... later when we get to do OpenGL programming, we'll do something similar but with a 3D hello world floating in 3D space.)

Solution to Wipe-Across Hello World

```
import pygame, sys
pygame.init()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

WHITE = (255, 255, 255)

font = pygame.font.Font(None, 100)
image = font.render("hello world", 1, WHITE)
rect = image.get_rect()

surface_rect = rect.move((WIDTH - rect.w)/2, \
                         (HEIGHT - rect.h)/2)
image_rect = pygame.Rect(0, 0, 0, rect.h)

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    # If portion of image to blit is not the
    # full image rect expand it.
    if image_rect.w < rect.w:
        image_rect.w = image_rect.w + 1
        surface.blit(image, surface_rect, image_rect)

    pygame.display.flip()
    pygame.time.delay(100)
```

Solution to the Pixel-by-Pixel Hello World

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

WHITE = (255, 255, 255)

font = pygame.font.Font(None, 100)
image = font.render("hello world", 1, WHITE)
rect = image.get_rect()

xoffset = (WIDTH - rect.w) / 2
yoffset = (HEIGHT - rect.h) / 2

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    x = random.randrange(rect.w)
    y = random.randrange(rect.h)
    image_rect = pygame.Rect(x, y, 1, 1)
    surface_rect = rect.move(xoffset + x, yoffset + y)
    surface.blit(image, surface_rect, image_rect)

    pygame.display.flip()
```

Solution to Bouncing Hello World

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

BLACK = (0, 0, 0)

def move(d, v, m):
    d = d + v
    if d > m:
        d = m
        v = -v
    elif d < 0:
        d = 0
        v = -v
    return d, v

R = random.randrange(256)
G = random.randrange(256)
B = random.randrange(256)
Rspeed = random.randrange(1, 3)
Gspeed = random.randrange(1, 3)
Bspeed = random.randrange(1, 3)

font = pygame.font.Font(None, 100)

image = font.render("hello world", 1, BLACK)
imagerect = image.get_rect()

x = random.randrange(WIDTH - imagerect.x)
y = random.randrange(HEIGHT - imagerect.h)
xspeed = random.randrange(1, 3)
yspeed = random.randrange(1, 3)

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    surface.fill(BLACK)

    R, Rspeed = move(R, Rspeed, 255)
    G, Gspeed = move(G, Gspeed, 255)
    B, Bspeed = move(B, Bspeed, 255)
    color = (R, G, B)

    x, xspeed = move(x, xspeed, WIDTH - 1 - imagerect.w)
```

```
    y, yspeed = move(y, yspeed, HEIGHT - 1 - imagerect.h)

    surfacerect = imagerect.move([x, y])

    image = font.render("hello world", 1, color)
    surface.blit(image, surfacerect)

    pygame.display.flip()
    pygame.time.delay(10)
```

Game Project Part 8

Now add the following to your program:

If you manage to kill the alien, print a congratulatory message on the screen and at the top, print a score. You can figure out how you want to give the score. For instance you might want to give a higher score if the alien is shot early rather than late. I'll just use alien_rect.y: the smaller the value of alien_rect.y, the higher the score.

Hint: You probably want to print this message after the explosion. You also want to the player to see the message. So you should insert a delay before you stop the game. One easy way to stop the game is simply to exit the game loop using `break`.

You can simple display the message or you can use any of the effects such as wipe-across.

Another hint: The computation of my score implies that the score is an integer. But you know that to produce a text image requires a string. You can get a string from an integer using the `str` function. Check this out:

```
score = 1234
text = "Score: " + str(score)
print text
```

Solution to Game Project Part 8

```
import pygame, sys, random
pygame.init()
random.seed()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

pygame.key.set_repeat(10, 10)

SCORE_HEIGHT = 24
BLACK = (0, 0, 0)
RED = (255, 0, 0)
sys.stdout = file("stdout.txt", "w")
sys.stderr = file("stderr.txt", "w")

# Create alien
alien_image = pygame.image.load("GalaxianAquaAlien.gif")
alien_rect = alien_image.get_rect()
alien_rect = alien_rect.move([0,SCORE_HEIGHT])
alien_speed = [1, 0]
ALIEN_Y_INCREMENT = 3

# Create flagship
ship_image = pygame.image.load("GalaxianGalaxip.gif")
ship_rect = ship_image.get_rect()
x = (WIDTH - ship_rect.w) / 2
y = HEIGHT - ship_rect.h
ship_rect = ship_rect.move([x,y])
ship_speed = [0, 0]

# Ship's laser
laser_rect = pygame.Rect(0, 0, 4, 8)
laser_speed = [0, 0]
laser_alive = False

# Laser sound
laser_sound = pygame.mixer.Sound("laser.wav")

# Explosion sound
explosion_sound = pygame.mixer.Sound("gexplode.wav")
explosion_played = False

def move(d, v, m):
    d = d + v
    if d < 0:
        d = 0
        v = -v
    elif d > m:
        d = m
        v = -v
```

```
    return d, v

collides = False
while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            keypress = pygame.key.get_pressed()
            if keypress[pygame.K_LEFT]:
                ship_speed = [-1, 0]
            elif keypress[pygame.K_RIGHT]:
                ship_speed = [1, 0]
            elif keypress[pygame.K_SPACE]:
                if not laser_alive:
                    laser_alive = True
                    laser_speed = [0, -2]
                    laser_rect.x = ship_rect.x + \
                                   (ship_rect.w - laser_rect.w)/2
                    laser_rect.y = ship_rect.y - laser_rect.w
                    laser_sound.play()

    surface.fill(BLACK)

    if not collides:
        alien_rect.x, alien_speed[0] = move(alien_rect.x, \
                                alien_speed[0], WIDTH - alien_rect.w)
        if random.randrange(100) == 0:
            alien_rect.y = alien_rect.y + ALIEN_Y_INCREMENT

        ship_rect.x, ship_speed[0] = move(ship_rect.x, ship_speed[0],\
                                    WIDTH - ship_rect.w)
        ship_speed = [0, 0]

        if laser_alive:
            laser_rect.y, laser_speed[1] = move(laser_rect.y, \
                                laser_speed[1], HEIGHT - laser_rect.h)
            if laser_rect.y < SCORE_HEIGHT:
                laser_alive = False

    collides = alien_rect.colliderect(ship_rect) or \
               laser_rect.colliderect(alien_rect)

    if collides == True and explosion_played == False:
        explosion_sound.play()
        explosion_played = True
        pygame.time.delay(100)

    surface.blit(alien_image, alien_rect)
    surface.blit(ship_image, ship_rect)
    if laser_alive:
        pygame.draw.rect(surface, RED, laser_rect)
    pygame.display.flip()
```

```
    # Display message if there is a collision and then break
    if collides == True:
        if alien_rect.colliderect(ship_rect):
            message = "The alien had you for lunch"
            score = 0
        else:
            message = "You saved the world!"
            score = 500 - alien_rect.y

        # Draw the score
        WHITE = (255, 255, 255)
        font = pygame.font.Font(None, SCORE_HEIGHT)
        image = font.render("Score: " + str(score), 1, WHITE)
        rect = image.get_rect()
        surface.blit(image, rect)

        # Draw a message
        font = pygame.font.Font(None, 48)
        image = font.render(message, 1, WHITE)
        rect = image.get_rect()
        surface_rect = rect.move((WIDTH - rect.w)/2, \
                       (HEIGHT - rect.h)/2)
        image_rect = pygame.Rect(0, 0, 0, rect.h)

        while 1:
            if image_rect.w < rect.w:
                image_rect.w = image_rect.w + 1
                surface.blit(image, surface_rect, image_rect)
            else:
                break
            pygame.display.flip()
            pygame.time.delay(10)
        pygame.time.delay(3000)
        break
```

TADA!

Well there you go ... that's your first game.

Obviously you can improve on this in many ways. Here are some obvious possibilities:

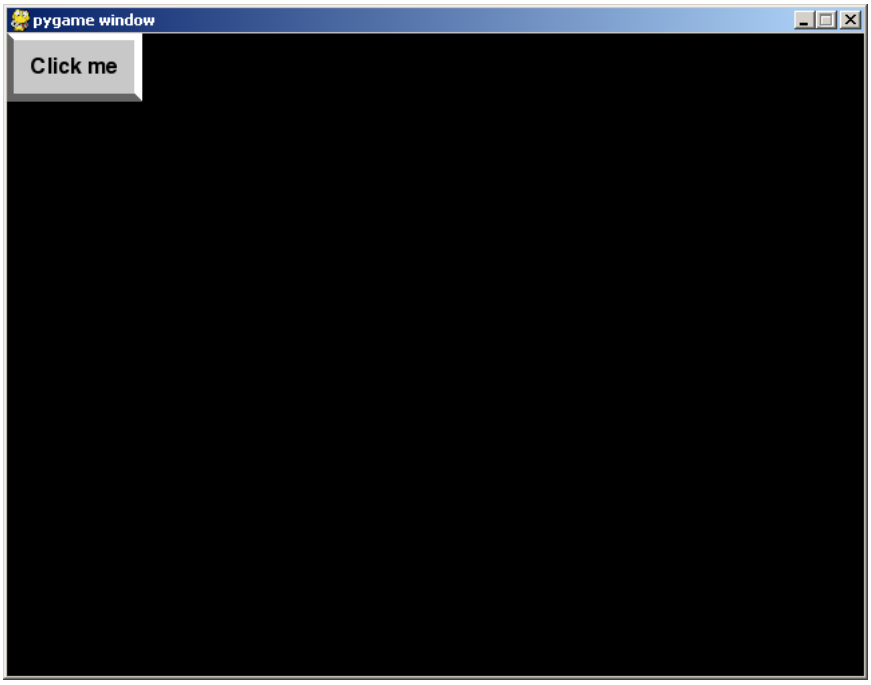
- Improve the collision detection. For instance the rect used for detecting collision with an alien bug should actually be smaller.
- Have more aliens.
- Let aliens have laser cannons too.
- Make aliens randomly zoom down on you.
- Give the player three ships.
- Have an explosion animation.
- Allow one and two players to play the game.
- Have different background music.
- Have a greeting and demo scene.
- Have levels: Each time the player kills a whole squadron of aliens, bring a new squadron that moves faster and aims much better at your ship.
- Store high scores.
- Etc.

It's also a good idea to clean up your code, put more comments, etc.

Have fun!

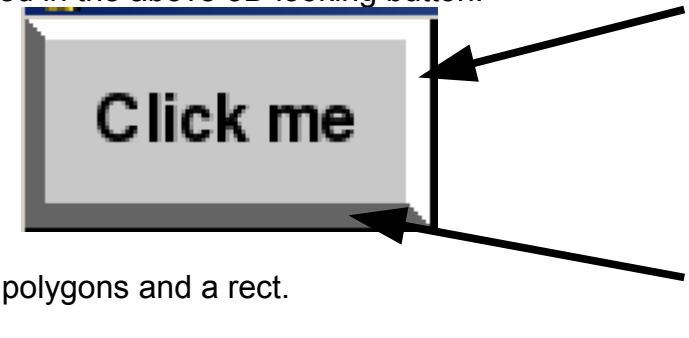
Mouse and a Button

One last thing before I'm done. I'll just show you how to read mouse inputs. Another thing showed in the example is drawing polygons. The example involves creating a button. When you run the program you see this:



When you press the button with your mouse, the button appears to be pressed and then pops up after a second.

The trick to used in the above 3D looking button:



is to draw two polygons and a rect.

To draw a polygonal, for instance a triangle with corners at (0,0), (100, 100), (150, 200) in red, you do this:

```
RED = (255,0,0)
```

```
points = [(0,0),(100,100),(150,200),(0,0)]
pygame.draw.polygon(surface, RED, points)
```

I'll leave it to you to study and experiment with the code. You will find the documentation from pygame (see previous worksheets) useful.

Here you go:

```
import pygame, sys
pygame.init()

WIDTH, HEIGHT = 640, 480
SIZE = (WIDTH, HEIGHT)
surface = pygame.display.set_mode(SIZE)

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)

# up button
points1 = [(0,0),(100,0),(100,50),(95,45),(95,5),(5,5),(0,0)]
points2 = [(0,0),(0,50),(100,50),(95,45),(5,45),(5,5),(0,0)]
color1 = (255,255,255)
color2 = (100,100,100)
color3 = (200,200,200)

# down button
dpoints1 = [(1,1),(101,1),(101,51),(96,46),(96,6),(6,6),(1,1)]
dpoints2 = [(1,1),(1,51),(101,51),(96,46),(6,46),(6,6),(1,1)]
dcolor1 = (255,255,255)
dcolor2 = (100,100,100)
dcolor3 = (200,200,200)

# label for button
font = pygame.font.Font(None, 24)
image = font.render("Click me", 1, BLACK)
rect = image.get_rect()
rect = rect.move((100 - rect.w)/2, (50 - rect.h)/2)
drect = rect.move(1,1)

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        if event.type == pygame.MOUSEBUTTONDOWN:
            x, y = pygame.mouse.get_pos()
            surface.fill(BLACK)
            pygame.draw.polygon(surface, dcolor1, dpoints1)
            pygame.draw.polygon(surface, dcolor2, dpoints2)
            pygame.draw.rect(surface, color3, pygame.Rect(6,6,91,41))
            surface.blit(image, drect)
            pygame.display.flip()
```

```
            if 0 <= x <= 100 and 0 <= y <= 50:
                print "pressed"
            pygame.time.delay(1000)

    surface.fill(BLACK)
    pygame.draw.polygon(surface, color1, points1)
    pygame.draw.polygon(surface, color2, points2)
    pygame.draw.rect(surface, color3, pygame.Rect(5,5,90,40))
    surface.blit(image, rect)

    pygame.display.flip()
    pygame.time.delay(10)
```

Exercise. What is the difference between the points of the polygons for the button when it's up and when it's down?

Exercise. Modify the program so that when the button is clicked, you play a button click sound. I'll leave it to you to find a suitable sound file.

Exercise. Modify your game so that options in your game are implemented as buttons (for instance one-player or two-player, etc.)

Exercise. As an exercise on drawing polygons, write a program with three points bouncing within the screen. Using the three moving points, draw a moving triangle. Make the color change gradually.

By the way, there are tools that will help you develop GUI-based programs (GUI = graphical user interface), i.e. programs with buttons, forms, etc. So if you are going to write a huge program with many GUI elements like buttons, forms, etc., the first thing is to investigate which tool works best for you and then learn to use it. Developing a complete GUI package is extremely time-consuming. Of course if you have only a couple of buttons, you can hand-code your own buttons.