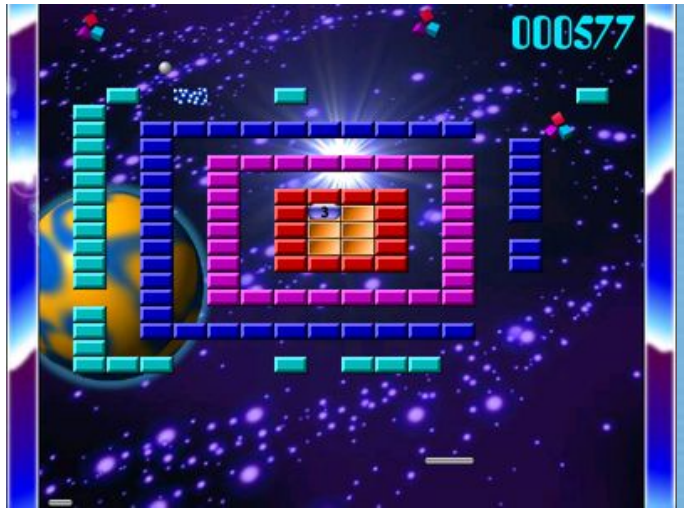
softserve experience matters

PyGame

PyGame intro

PyGame (the library) is a **Free** and **Open Source** python programming language library for making multimedia applications like games built on top of the excellent SDL (**Simple DirectMedia Layer**) library. Like SDL, pygame is highly portable and runs on nearly every platform and operating system. See: http://www.pygame.org



Installation

The best way to install pygame is with the pip tool.

```
sudo pip install pygame        #Linux and Mac OS X
```

```
pip install pygame              #Windows
```

```
python -m pip install pygame --user
```

→ We use the **--user** flag to tell it to install into the home directory, rather than globally.

```
pip install –user  pygame
```

And now you can try! →

```
python -m pygame.examples.aliens
```

You can find many examples of realization in package folder: *\site-packages\pygame\examples*

Let start

Let create our very first PyGame instance …

```
import pygame

pygame.init()
```

1. This will initiate PyGame, and allow you to then make various commands with PyGame

```
gameDisplay = pygame.display.set_mode((800,600))
```

2. We define our game's display, which is the main "display" for our game.
It's our canvas that we will draw things to.
Resolution of our game to be 800 px wide and 600 px tall.

```
pygame.display.set_caption('My first game')
```

3. We define our display's "caption."

```
clock = pygame.time.Clock()
```

4. We use clock to track time within the game, and this is mostly used for FPS, or "frames per second." For the most part, the average human eye can see ~30 FPS

Let start

Let create our very first PyGame instance ...

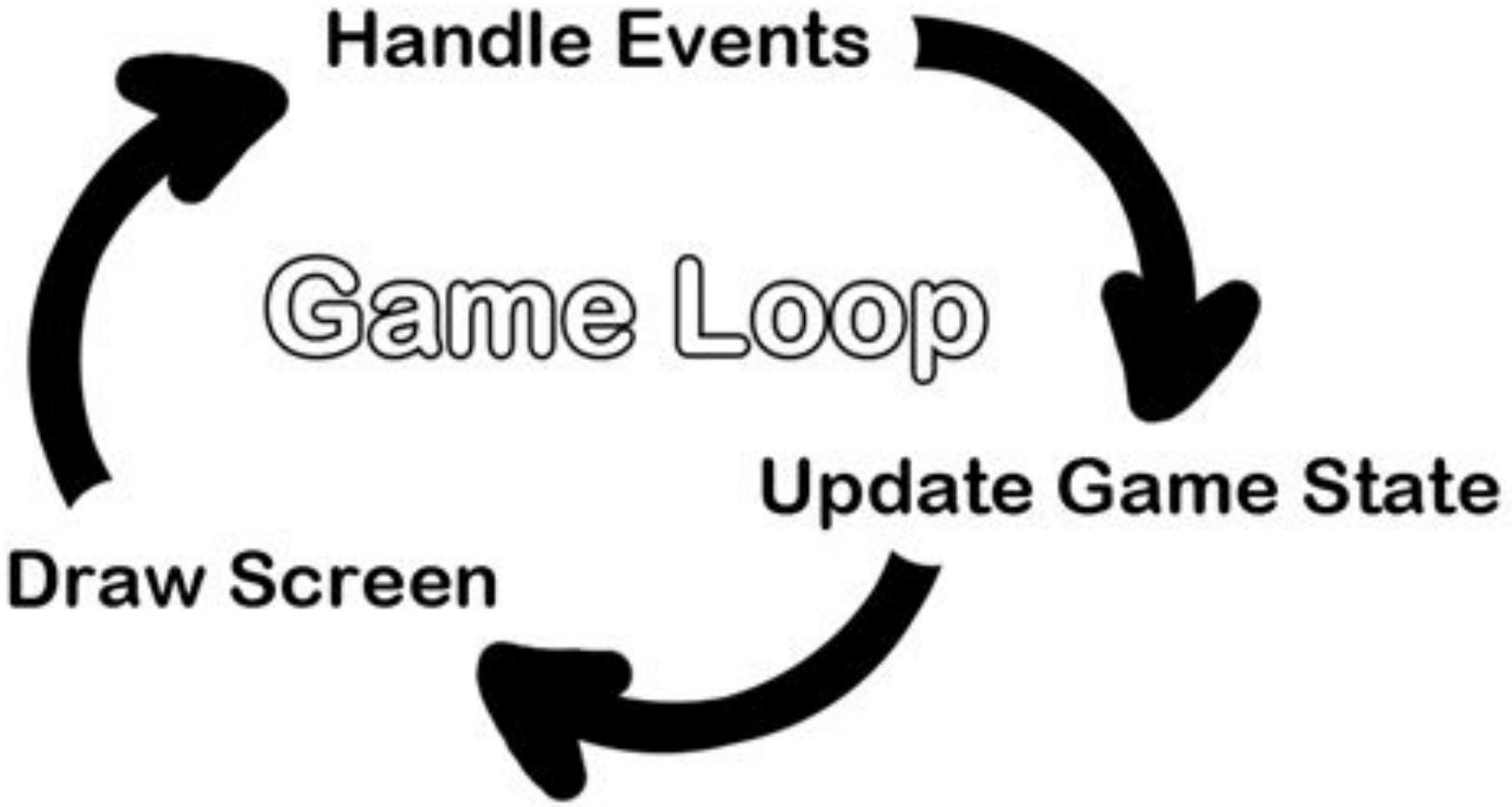
Першим аргументом передається необхідна роздільна здатність в вигляді кортежу з двох цілих чисел. Перше число - ширина в пікселях, друге - висота. Якщо передати (0,0), то роздільна здатність буде така ж як і була до того. Якщо нуль буде в висоті, чи в ширині, то не зміниться відповідно тільки висота чи ширина, але краще таким не зловживати.

```
gameDisplay = pygame.display.set_mode((800,600))
```

Другим параметром є тип необхідного нам дисплею.
Можна обирати між такими типами:

```
pygame.FULLSCREEN #Повноекранний режим (інакше віконний)
pygame.DOUBLEBUF #Подвійна буферизація (рекомендовано використовувати разом з наступними двома)
pygame.HWSURFACE #апаратне прискорення (тільки з FULLSCREEN)
pygame.OPENGL #створити дисплей, що доступний для малювання з допомогою OpenGL
pygame.RESIZABLE #створене вікно дозволяє користувачу змінювати свій розмір
pygame.NOFRAME #вікно не буде мати рамки та заголовка
```

Main game Loop



```
# Loop until the user clicks the close button.
done = False
# Used to manage how fast the screen updates
clock = pygame.time.Clock()

# -------- Main Program Loop -----------
while not done:
    # --- Main event loop
    for event in pygame.event.get(): # User did something
        if event.type == pygame.QUIT: # If user clicked close
            done = True # Flag that we are done so we exit this loop

    # --- Game logic should go here
    # --- Drawing code should go here
    # First, clear the screen to white. Don't put other drawing commands
    # above this, or they will be erased with this command.
    screen.fill(WHITE)
    # --- Go ahead and update the screen with what we've drawn.
    pygame.display.update()
    # --- Limit to 60 frames per second
    clock.tick(60)
```

Main program Loop

- First we declarate variable **done**
- Then, we run our "game loop," which will run while we have **done = false**
- We have a for loop within this while loop. This is going to be present in most PyGame scripts, where events are constantly being logged
- After our if statement we run a **pygame.display.update**. It'll update the entire surface as well
- **clock.tick(60)**. Basically, this is how many frames per second we are running. In this case, we are running 60 FPS.

```
done = False

while not done:

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

        #print(event)

    pygame.display.update()

    clock.tick(60)
```

Quit

pygame.quit() - will end our pygame instance.

Then we can run a simple quit(), which will exit Python and the application.

```
pygame.quit()
quit()
```

Handles events

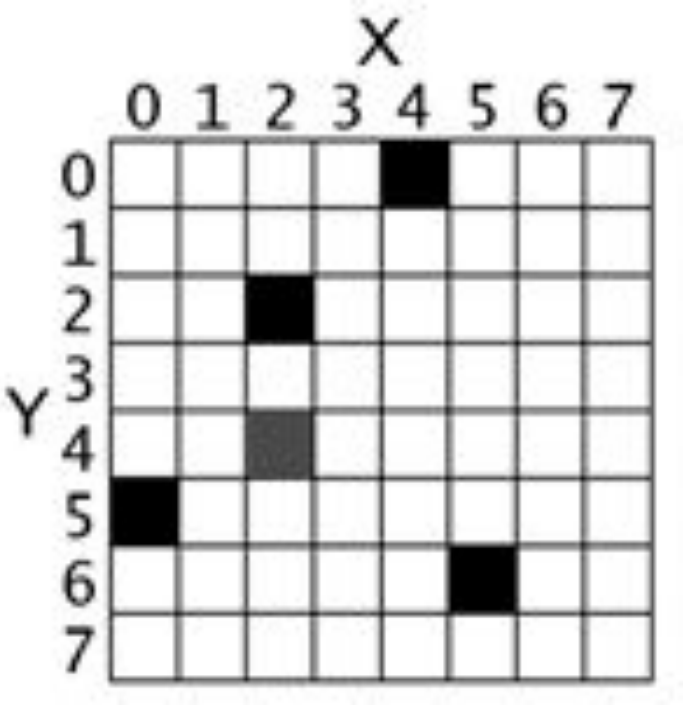
The main module for dealing with user input is the pygame.event module. Once you start your program, initialize variables, and draw your opening graphics, you probably need to respond to user input to make things happen. That's where the Pygame event loop comes in. Its job is to watch all the different inputs (keyboard, mouse, joysticks, etc.) and process them as they occur.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        print("User asked to quit.")
    elif event.type == pygame.KEYDOWN:
        print("User pressed a key.")
    elif event.type == pygame.KEYUP:
        print("User let go of a key.")
    elif event.type == pygame.MOUSEBUTTONDOWN:
        print("User pressed a mouse button")
```

Event Type	Parameters
QUIT	None
ACTIVEEVENT	gain, state
KEYDOWN	unicode, key, mod
KEYUP	key, mod
MOUSEMOTION	pos, rel, buttons
MOUSEBUTTONUP	pos, button
MOUSEBUTTONDOWN	pos, button
JOYAXISMOTION	joy, axis, value
JOYBALLMOTION	joy, ball, rel
JOYHATMOTION	joy, hat, value
JOYBUTTONUP	joy, button
JOYBUTTONDOWN	joy, button
VIDEORESIZE	size, w, h
VIDEOEXPOSE	None
USEREVENT	Code

Pixel Coordinates and Colors

The Pygame framework represents **Cartesian Coordinates** as a tuple of two integers, such as (4, 0) or (2, 2).
The first integer is the X coordinate and the second is the Y coordinate.



In Pygame, we represent colors with tuples of three integers. **RGB values.**

Color	RGB Values
Aqua	(0, 255, 255)
Black	(0, 0, 0)
Blue	(0, 0, 255)
Fuchsia	(255, 0, 255)
Gray	(128, 128, 128)
Green	(0, 128, 0)
Lime	(0, 255, 0)
Maroon	(128, 0, 0)
Navy Blue	(0, 0, 128)
Olive	(128, 128, 0)
Purple	(128, 0, 128)
Red	(255, 0, 0)
Silver	(192, 192, 192)
Teal	(0, 128, 128)
White	(255, 255, 255)
Yellow	(255, 255, 0)

Draw primitives

A program can draw things like rectangles, polygons, circles, ellipses, arcs, and lines : http://www.pygame.org/docs/ref/draw.html

pygame.draw.rect(Surface, color, Rect, width=0): return Rect
pygame.draw.line(Surface, color, start_pos, end_pos, width=1) : return Rect
pygame.draw.polygon(Surface, color, pointlist, width=0) : return Rect
pygame.draw.circle(Surface, color, pos, radius, width=0) : return Rect

```
pygame.draw.rect(screen, RED, [55, 50, 20, 25])
pygame.draw.line(screen, GREEN, [0, 0], [100, 100], 5)
```

Drawing Text

- There are three things that need to be done.
- First, the program creates a variable that holds information about the font to be used, such as what typeface and how big.
- Second, the program creates an image of the text. One way to think of it is that the program carves out a "stamp" with the required letters that is ready to be dipped in ink and stamped on the paper.
- The third thing that is done is the program tells where this image of the text should be stamped (or "blit'ed") to the screen.

```
# Select the font to use, size, bold, italics
font = pygame.font.SysFont('Calibri', 25, True, False)

# Render the text. "True" means anti-aliased text.
# Black is the color. The variable BLACK was defined
# above as a list of [0, 0, 0]
# Note: This line creates an image of the letters,
# but does not put it on the screen yet.
text = font.render("My text",True,BLACK)

# Put the image of the text on the screen at 250x250
screen.blit(text, [250, 250])
```

Documentation

- http://pygame.org/wiki/GettingStarted
- http://www.pygame.org/docs/
- http://programarcadegames.com/index.php?lang=en
- https://pythonprogramming.net/pygame-python-3-part-1-intro/
- http://inventwithpython.com/pygame/

softserve experience matters

Thank you!

USA HQ
Toll Free: 866-687-3588
Tel: +1-512-516-8880

Ukraine HQ
Tel: +380-32-240-9090

Bulgaria
Tel: +359-2-902-3760