Python game development

Alejandro García @AlexOteiza
agrs700@gmail.com

Why Python?

- **Very** productive
- Less code
- Cross-platform
- Integrable with other languages
- Fun

Game frameworks

- 2D:
  - **Pygame**
  - Pyglet
  - ika

    …
- 3D:
  - Panda3D
  - Blender Game Engine
  - Python-Ogre

    …

https://wiki.python.org/moin/PythonGames

- SDL Interface
- Windows, Mac and Linux compatible
- Perfect for prototyping and even comercial games

What is a videogame?

Main elements

- Screen graphics
- User input
- Playing sounds
- Game logic

Graphics

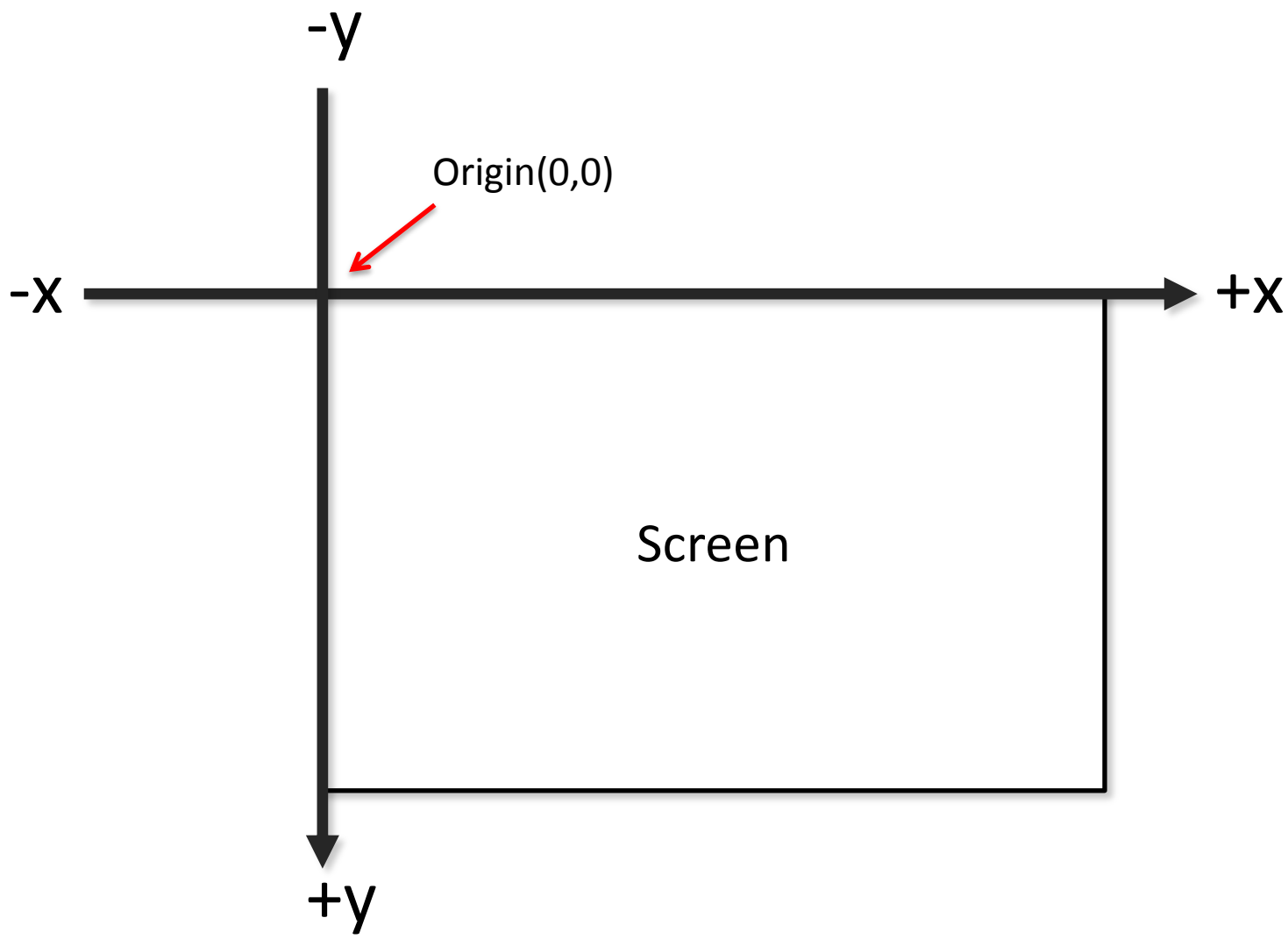
Opening window

```
import pygame


pygame.init()
screen_img = pygame.display.set_mode((640,480))
```

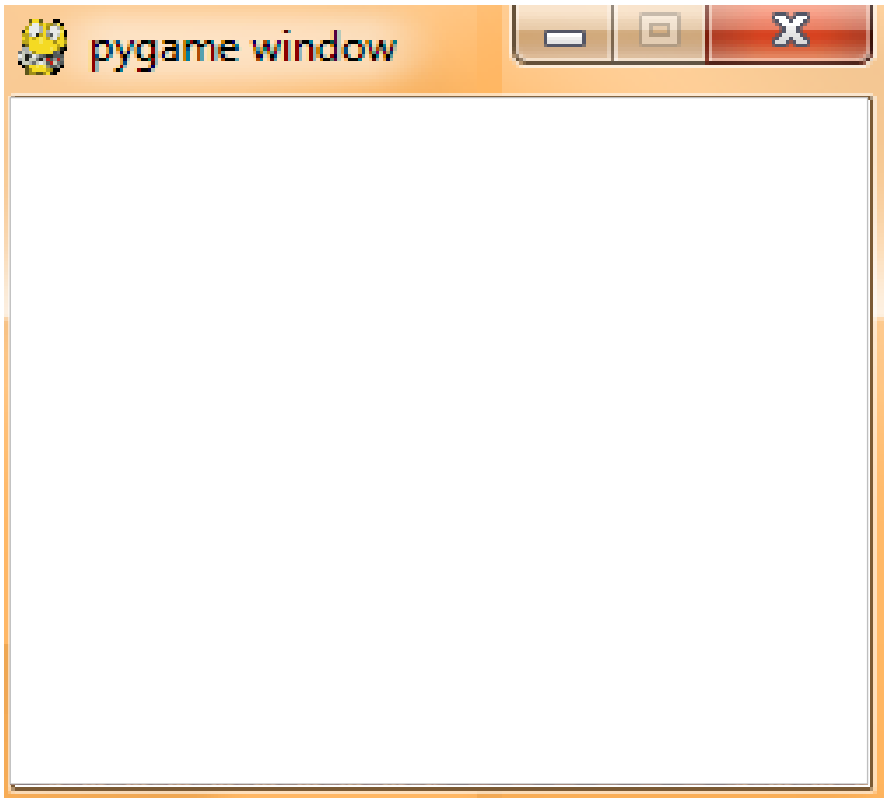
Events

```
import pygame

pygame.init()
screen_img = pygame.display.set_mode((640,480))
game_running = True
while game_running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            game_running = False

pygame.quit()
```

Coordinates system

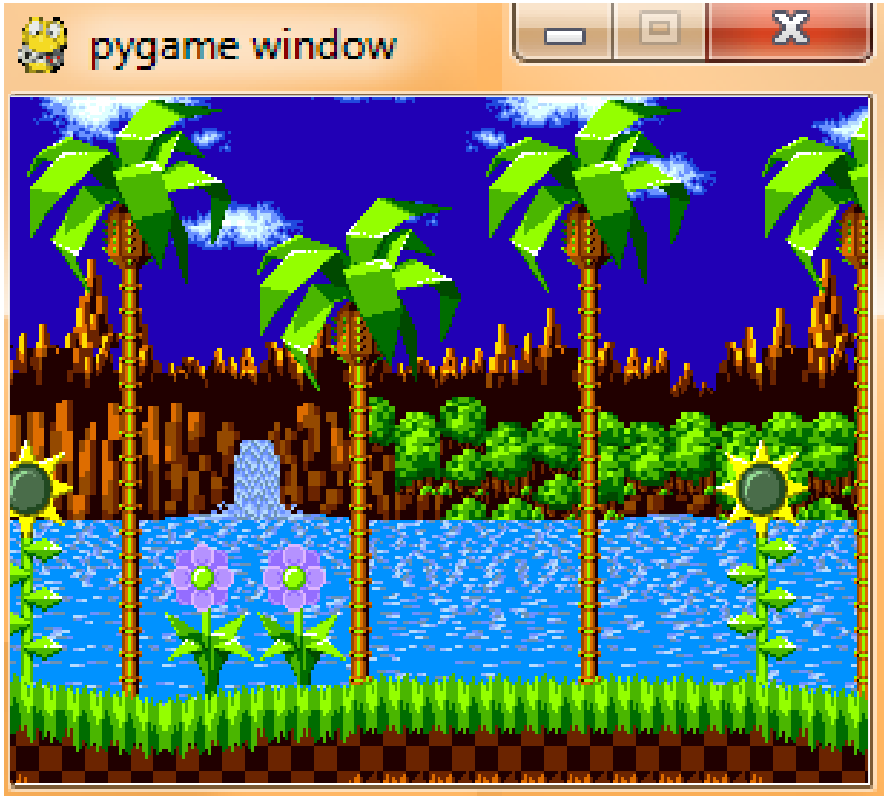


Drawing

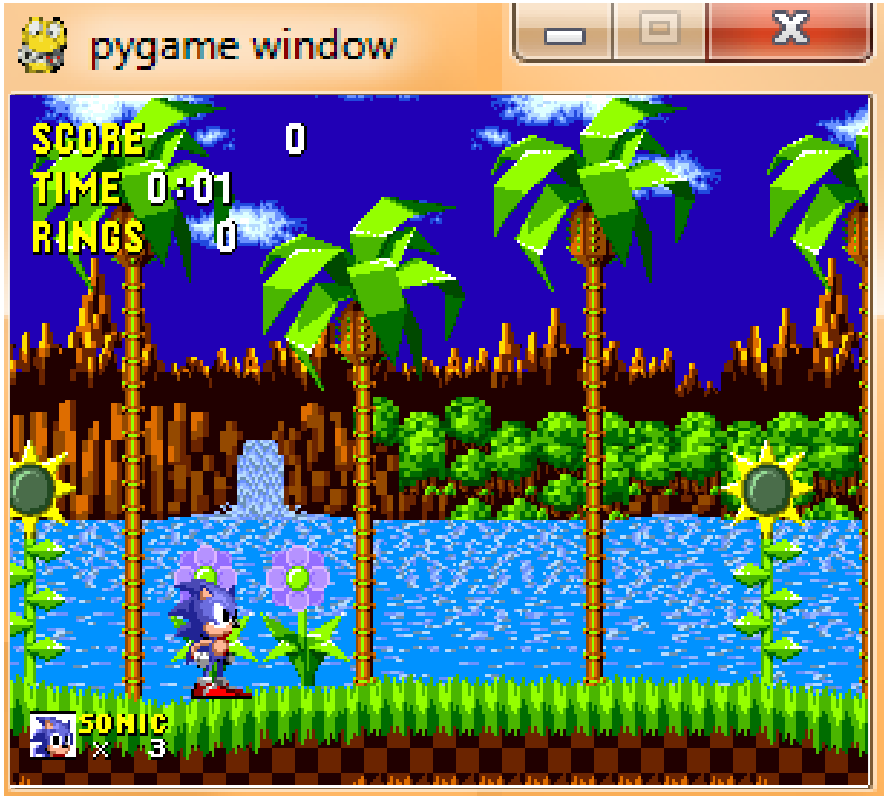


Drawing

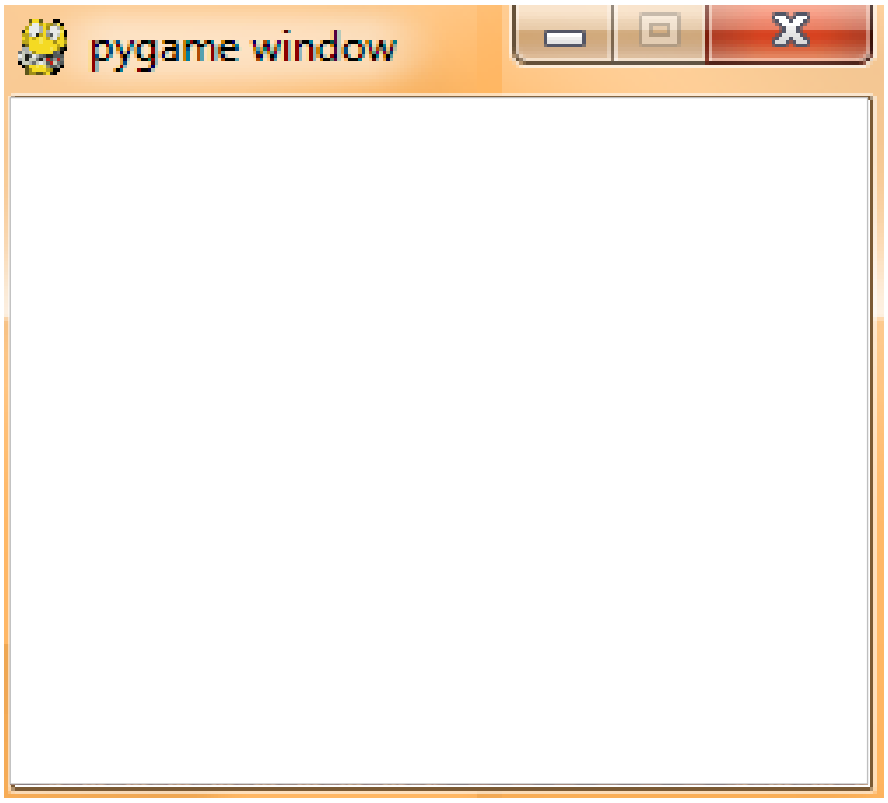
Drawing



Drawing

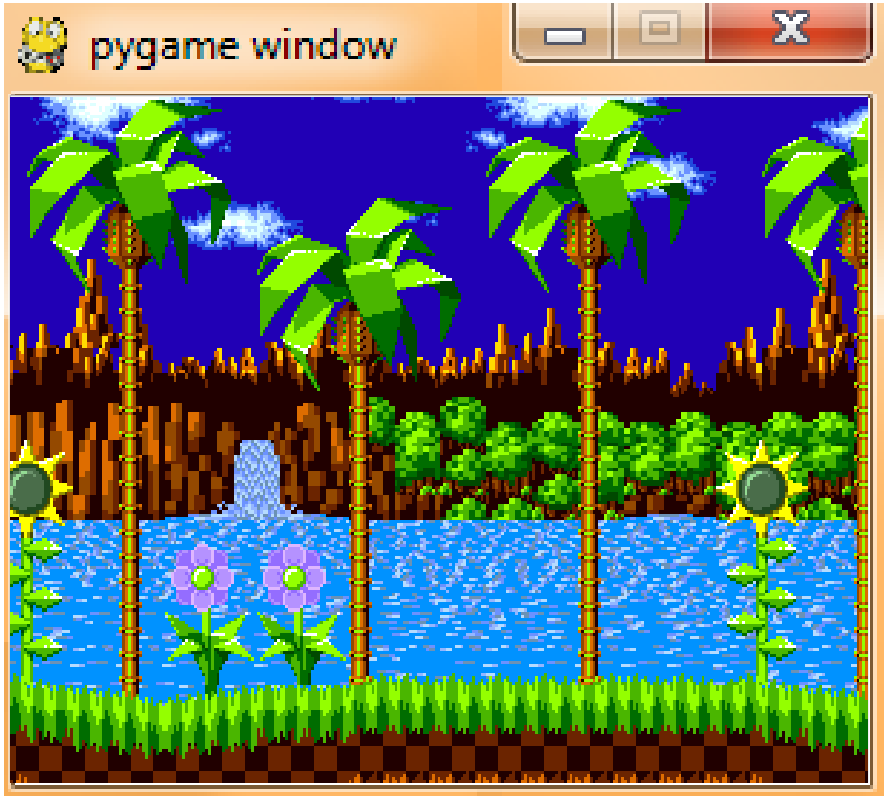


Drawing

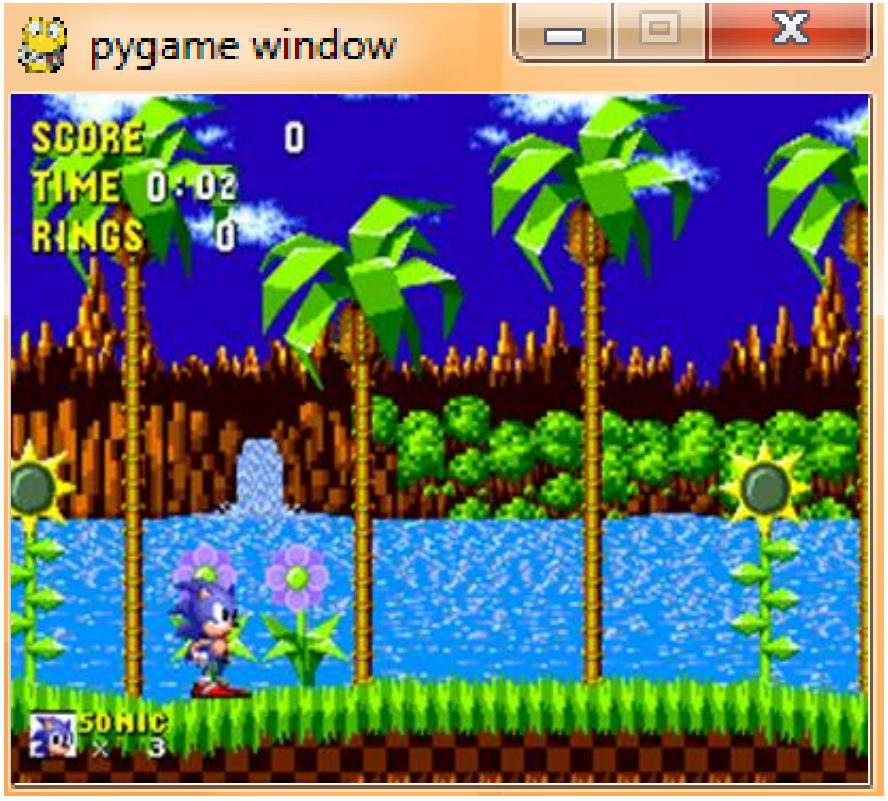


Drawing

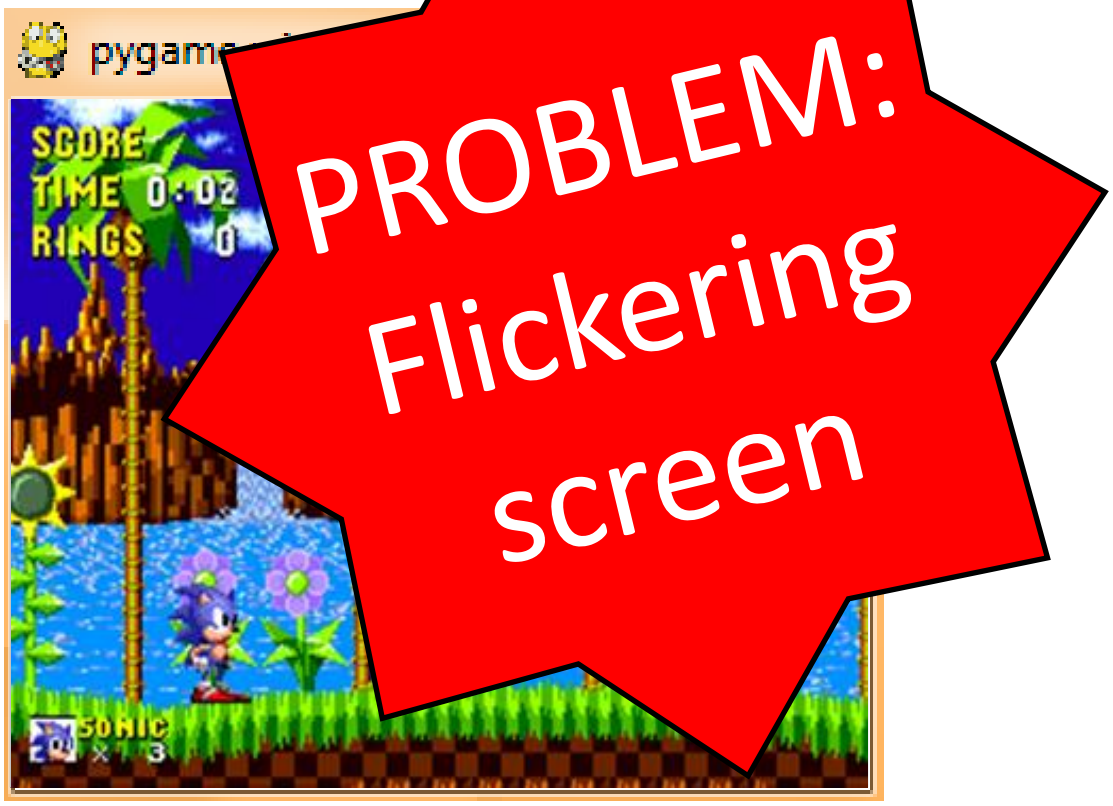
Drawing



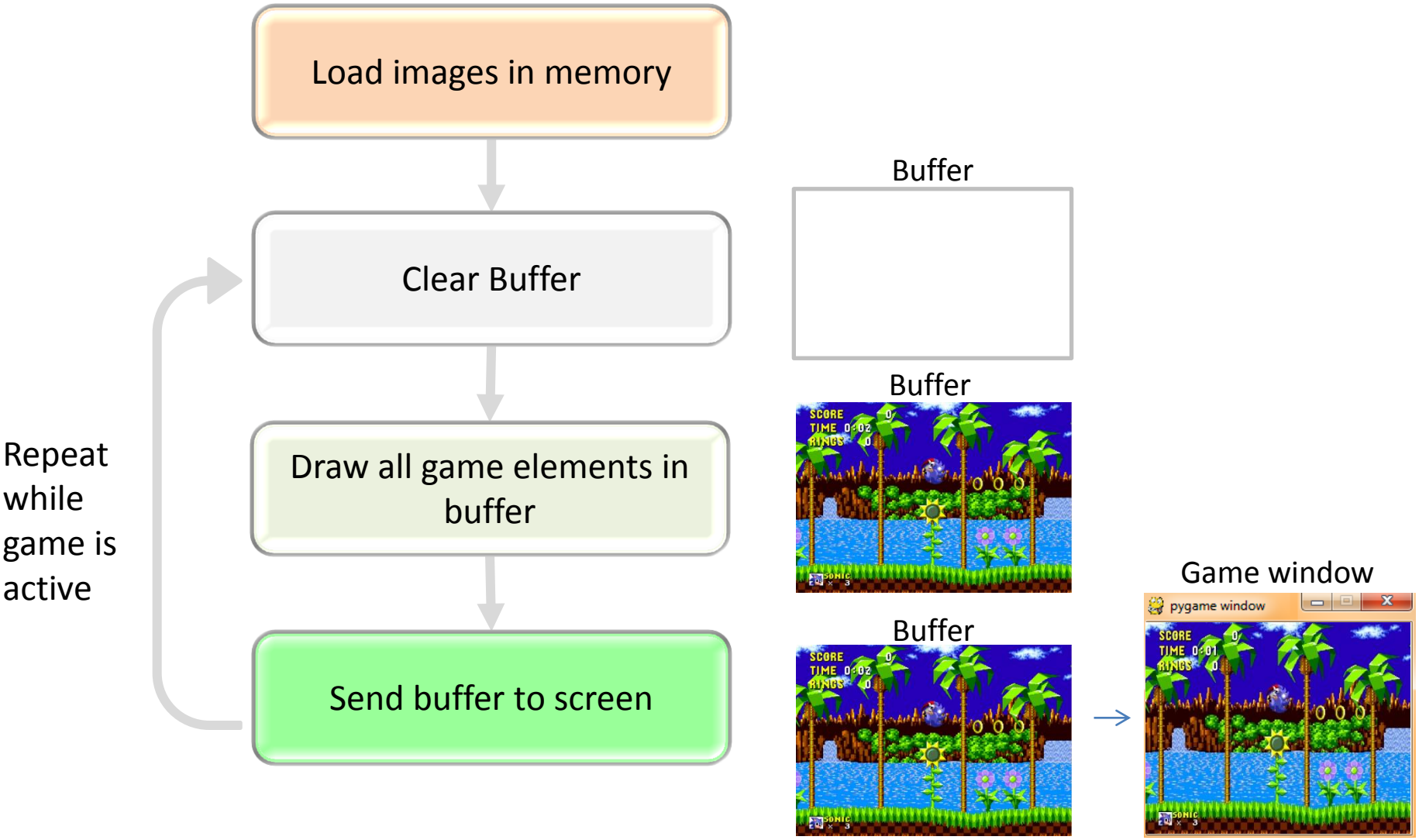
Drawing



Drawing



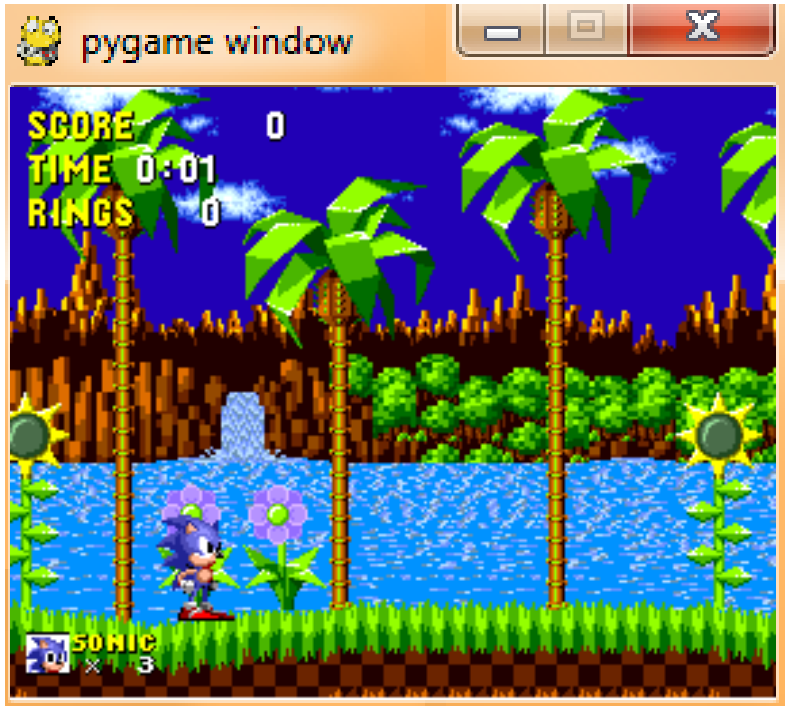
Drawing strategy: Double buffering



Drawing strategy: Double buffering

Memory Buffer

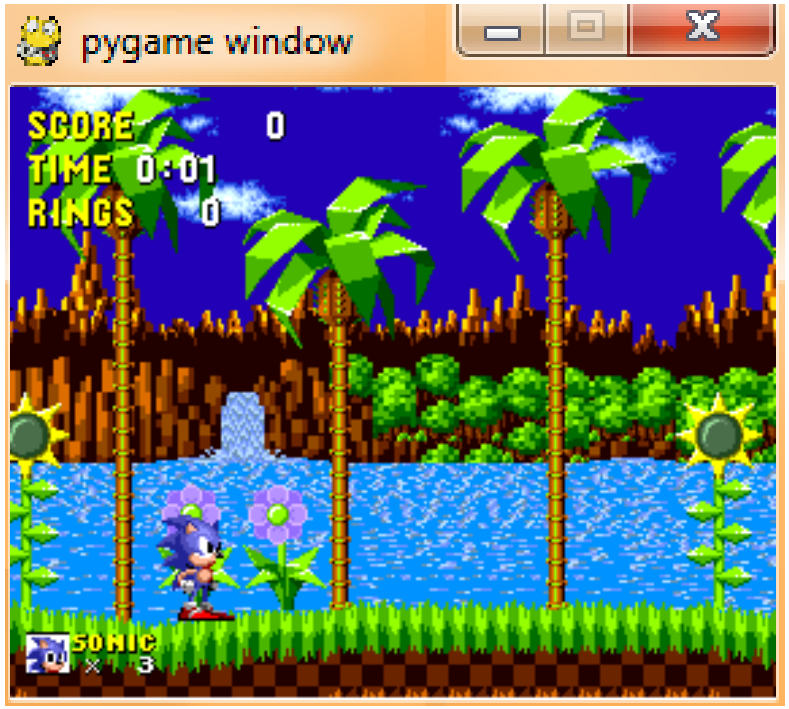
Screen



Drawing strategy: Double buffering

Memory Buffer

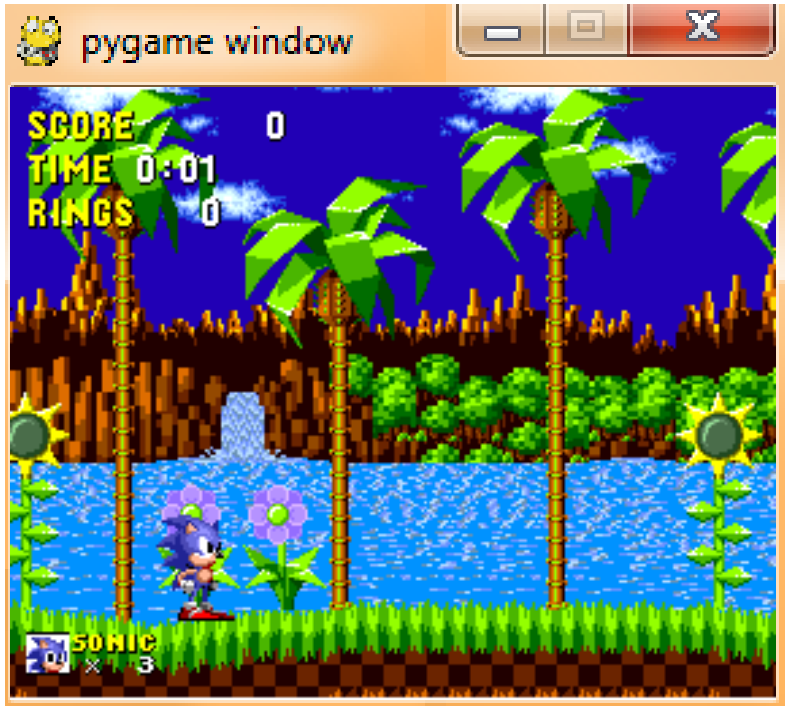
Screen



Drawing strategy: Double buffering

Memory Buffer

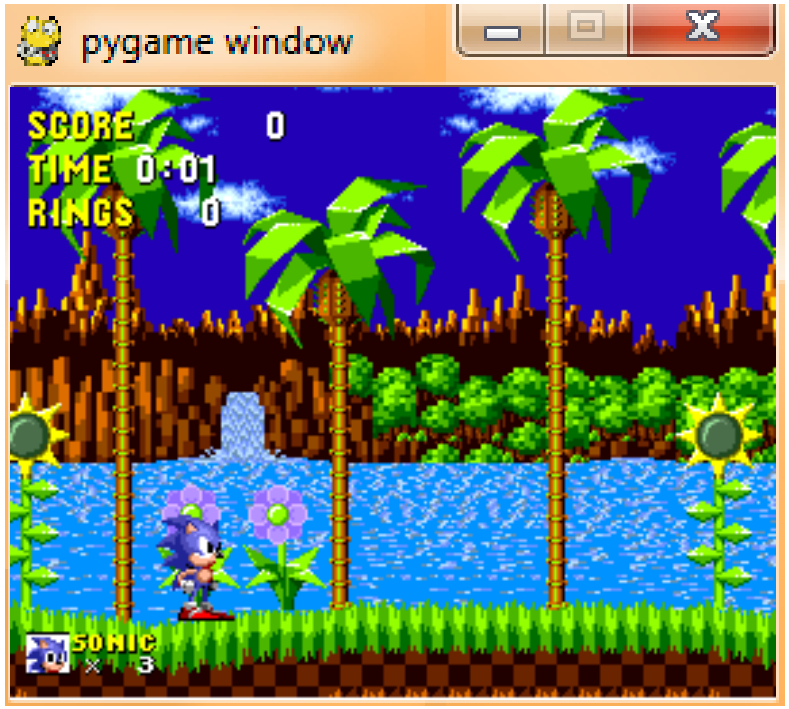
Screen



Drawing strategy: Double buffering

Memory Buffer

Screen



Drawing strategy: Double buffering

Memory Buffer

Screen

Drawing strategy: Double buffering

```
screen = pygame.display.set_mode(
(640,480), pygame.DOUBLEBUF)
```

(Enabled by default)

Drawing: Surface

pygame.surface.Surface

Drawing functions

Load Surface from file:

- `pygame.image.load(str)`

```
>>> player_image = pygame.image.load('link.png')
```

player_image

pygame.surface.Surface

Drawing functions

Load Surface from file:

- `pygame.image.load(str)`

```
>>> player_image = pygame.image.load('link.png')
```

player_image

pygame.surface.Surface

Fill Surface with color:

- `Surface.fill(color)`

RGB

```
>>> screen_img.fill((255, 255, 255))
```

screen_img

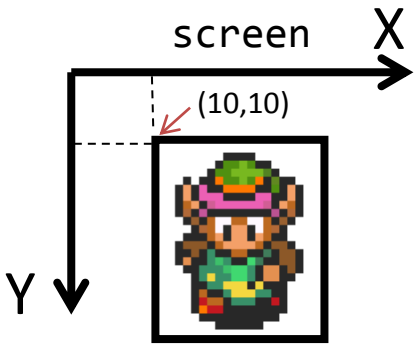
pygame.surface.Surface

(255,255,255)

screen_img

pygame.surface.Surface

Drawing functions



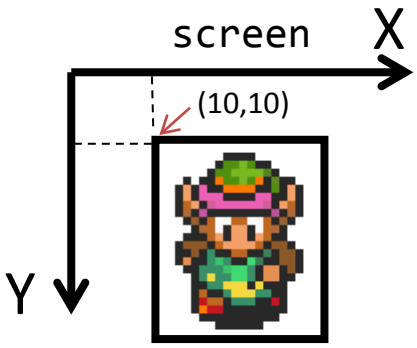
pygame.surface.Surface

Draw inside a Surface:

- `Surface.blit(surface, position)`

```
>>> screen_img.blit(player_image, (10,10))
```

Drawing functions



pygame.surface.Surface

Draw inside a Surface:

- `Surface.blit(surface, position)`

```
>>> screen_img.blit(player_image, (10,10))
```

Update display

- `pygame.display.flip()`

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

Code: Drawing

```
pygame.init()
screen_img = pygame.display.set_mode((640,480))
player_image = pygame.image.load('link.png')
game_running = True
while game_running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            game_running = False
    screen_img.fill((255,255,255))
    screen_img.blit(player_image, (10,10))
    pygame.display.flip()

pygame.quit()
```

Code: Drawing and moving character

```
    pygame.init()
    screen_img = pygame.display.set_mode((640,480))
    player_image = pygame.image.load('link.png')
    game_running = True
+   player_x = 0
    while game_running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                game_running = False
+       player_x += 1
        screen_img.fill((255,255,255))
*       screen_img.blit(player_image, (player_x,10))
        pygame.display.flip()
    pygame.quit()
```

Code: Limit FPS

```
    pygame.init()
    screen_img = pygame.display.set_mode((640,480))
    player_image = pygame.image.load('link.png')
    game_running = True
+   clock = pygame.time.Clock()
    player_x = 0
    while game_running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                game_running = False
        player_x += 1
        screen_img.fill((255,255,255))
        screen_img.blit(player_image, (player_x,10))
        pygame.display.flip()
+       clock.tick(60)
    pygame.quit()
```


INPUT

Reading keyboard

- Reading event queue

  `pygame.event.get()`

```
for event in pygame.event.get():
    if event.type == pygame.KEYDOWN:
        key_pressed = event.key
```

- Check keyboard's current state

  `pygame.key.get_pressed()`

  - Returns dict with pressed keys

Code: Movement with keyboard

```
...
player_x = 0
player_y = 0
while game_running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            game_running = False
    keyboard = pygame.key.get_pressed()
    if keyboard[pygame.K_RIGHT]:
        player_x += 0.1
    if keyboard[pygame.K_LEFT]:
        player_x -= 0.1
    if keyboard[pygame.K_DOWN]:
        player_y += 0.1
    if keyboard[pygame.K_UP]:
        player_y -= 0.1
    screen_img.fill((255,255,255))
    screen_img.blit(player_image, (player_x, player_y))
...
```

Mouse

- pygame.mouse.get_pressed()
- pygame.mouse.get_pos()
- pygame.mouse.set_cursor()
- pygame.mouse.set_visible()

…

Events:

- pygame.MOUSEMOTION
- pygame.MOUSEBUTTONUP
- Pygame.MOUSEBUTTONDOWN

http://www.pygame.org/docs/ref/mouse.html

Architecture?

Architecture

pygame.Rect

- Rectangular coordinates
- Attributes:
  - x
  - y
  - width
  - height
- Methods:
  - move()
  - move_ip()
  - …

+ info: http://www.pygame.org/docs/ref/rect.html

pygame.sprite.Sprite

- Base class for an "actor"
- Attributes:
  - rect <- pygame.Rect
  - image <- pygame.surface.Surface
  - …
- Methods:
  - kill()
  - update()
  - …

+ info: http://www.pygame.org/docs/ref/sprite.html

pygame.sprite.Group

- For managing multiple Sprites
- Methods:
  - add(Sprite)
  - update()
  - draw(Surface)
  - …

+ info: http://www.pygame.org/docs/ref/sprite.html

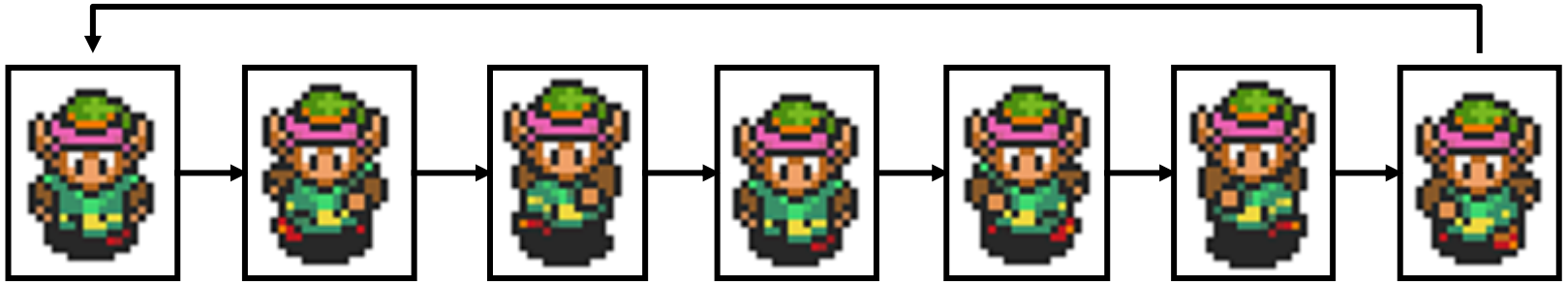
Code: Using classes(1)

```
class Player(pygame.sprite.Sprite):
    image_stand = pygame.image.load('link.png')
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = self.image_stand
        self.rect = self.image.get_rect()

    def update(self):
        keyboard = pygame.key.get_pressed()
        if keyboard[pygame.K_LEFT]:
            self.rect.move_ip(-0.1,0)
        if keyboard[pygame.K_RIGHT]:
            self.rect.move_ip(0.1,0)
        if keyboard[pygame.K_DOWN]:
            self.rect.move_ip(0,-0.1)
        if keyboard[pygame.K_UP]:
            self.rect.move_ip(0,0.1)
...
```

Code: Using classes(2)

```
...
def main():
    pygame.init()
    screen_img = pygame.display.set_mode((640,480))
    player = Player()
    player_group = pygame.sprite.Group()
    player_group.add(player)
    game_running = True
    while game_running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                game_running = False
        screen_img.fill((255,255,255))
        player_group.update()
        player_group.draw(screen_img)
        pygame.display.flip()
```

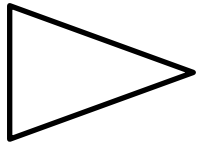
Animations

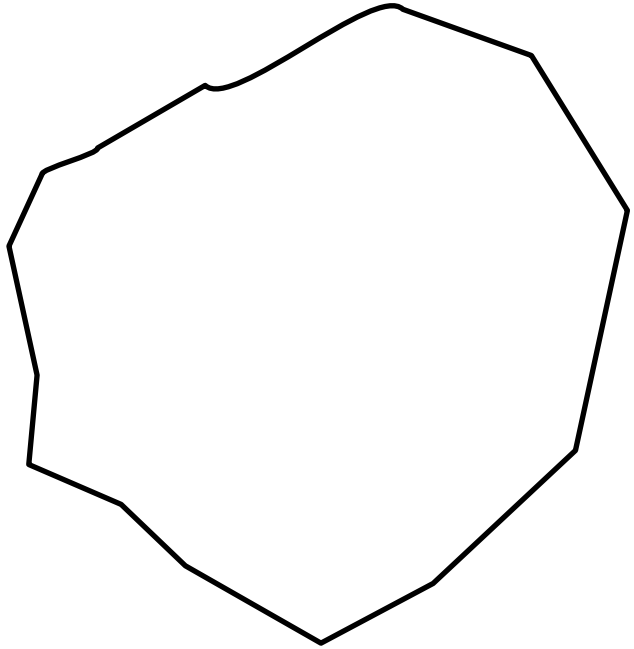


- Create and load a list of images
- Replace Sprite.image in each update

COLLISIONS

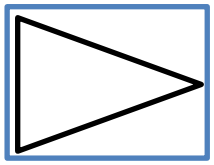
Collisions

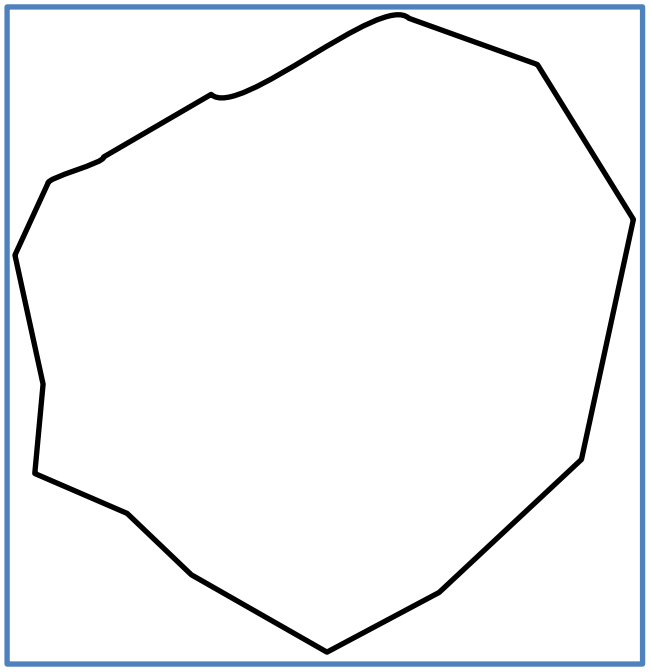




Approach: Rectangles

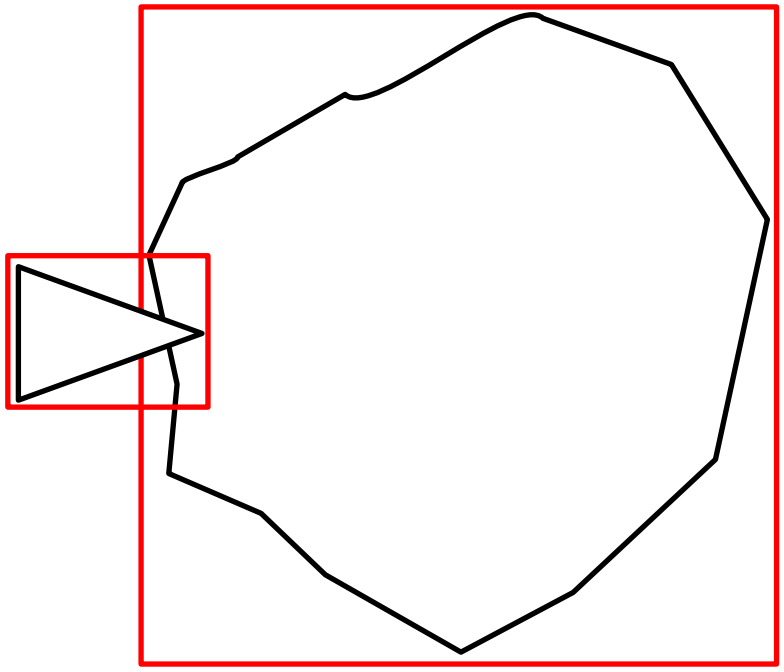
pygame.sprite.collide_rect(Sprite, Sprite)





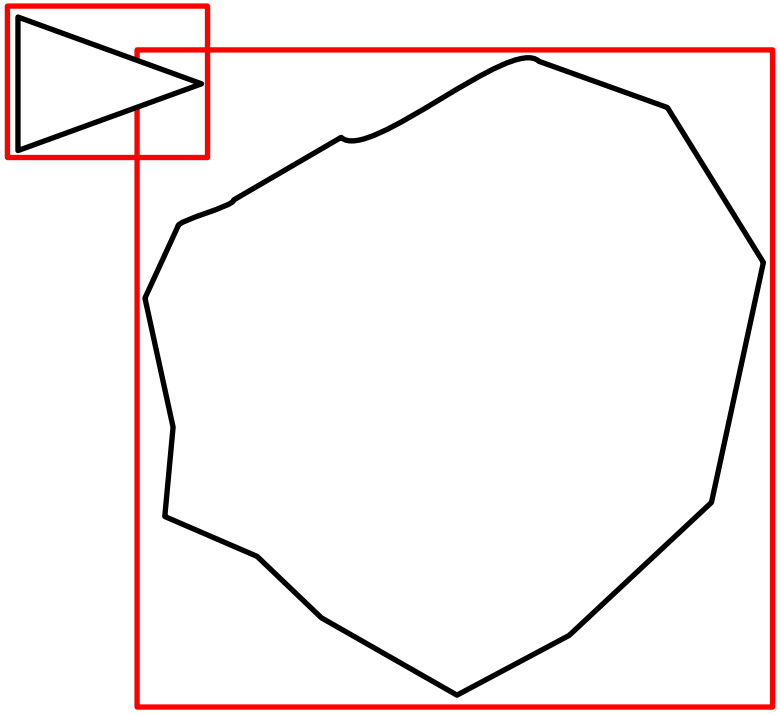
Approach: Rectangles

pygame.sprite.collide_rect(Sprite, Sprite)



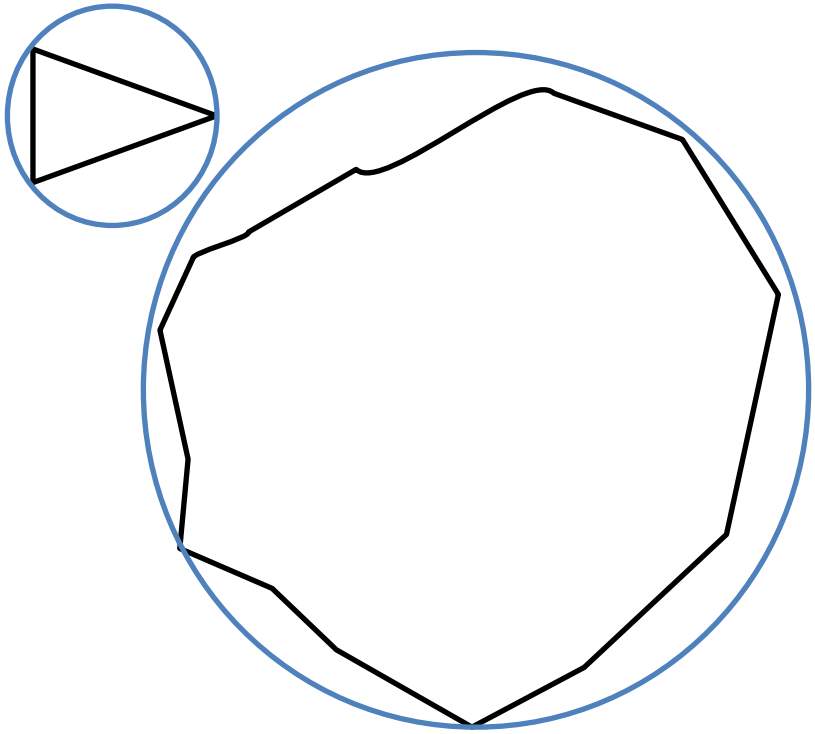
Approach: Rectangles

pygame.sprite.collide_rect(Sprite, Sprite)



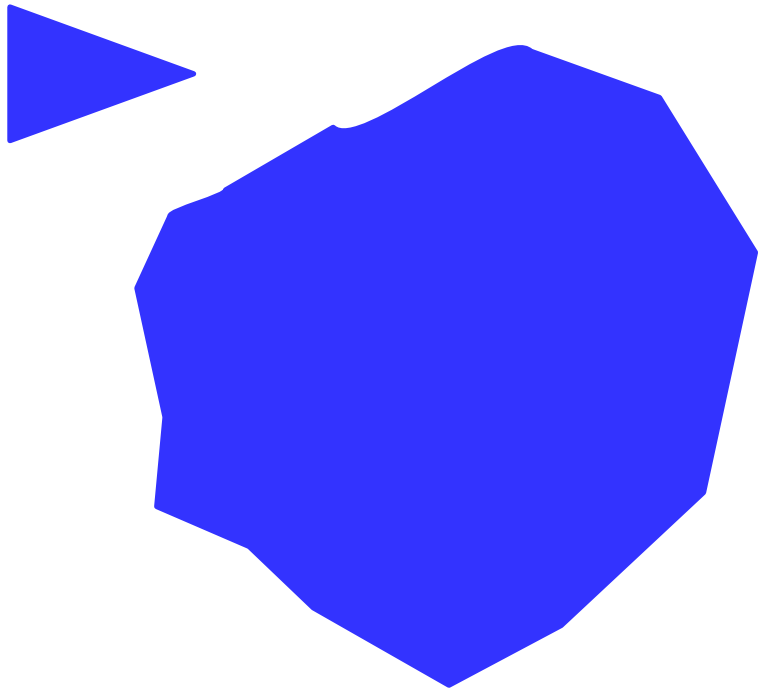
Approach: Circles

pygame.sprite.collide_circle(Sprite, Sprite)



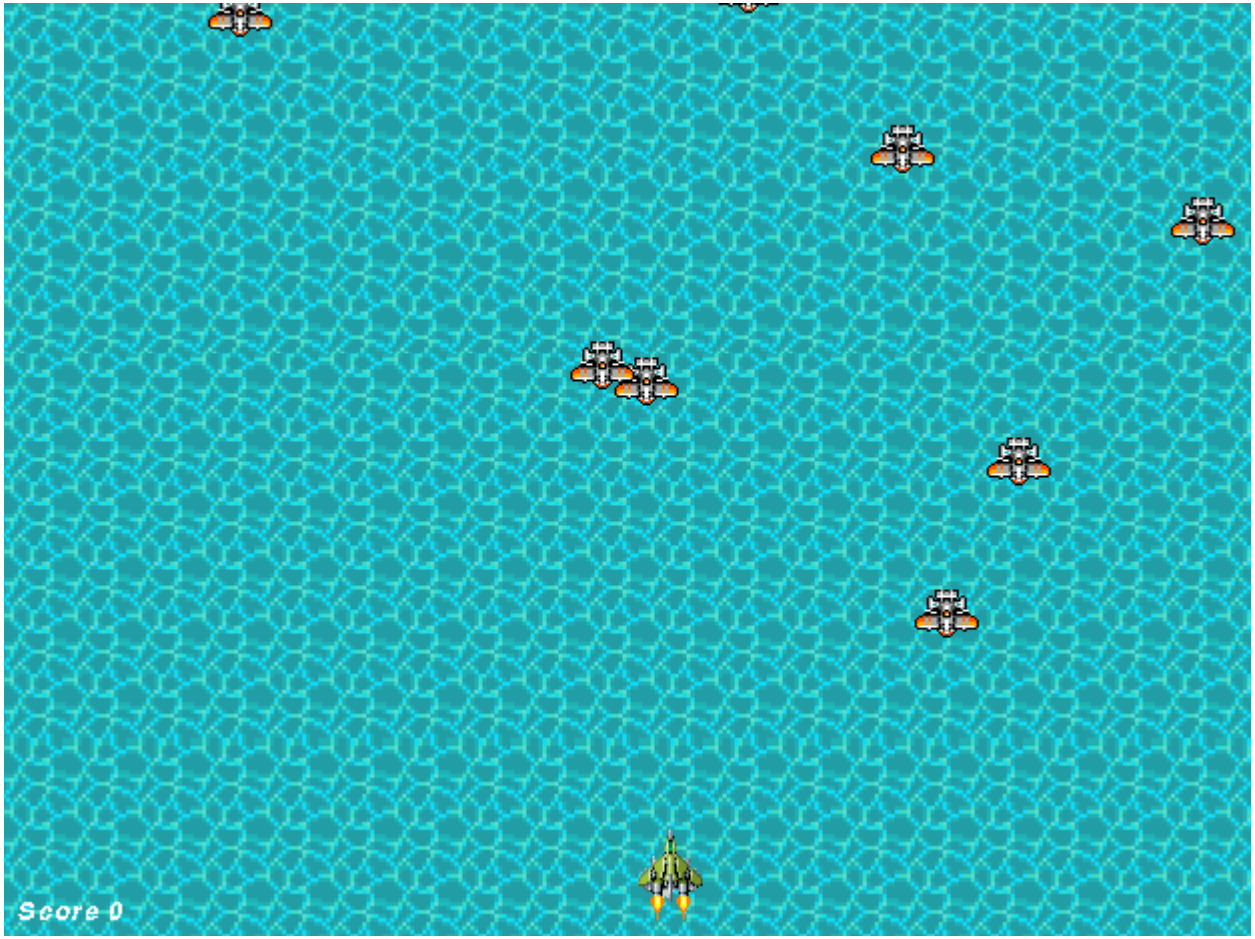
Approach: Mask

pygame.sprite.collide_mask(Sprite, Sprite)



sprite.mask = pygame.mask.from_surface(sprite.image)

Example: PyFighter




Questions?