Pygame in High School

by Maurizio Boscaini, Accra, Ghana, 8th August 2019

https://github.com/aidosnet/pygame-in-high-school

Table of contents

1/4 - Introduction

Just a short Seminar

Seminare = To Sow

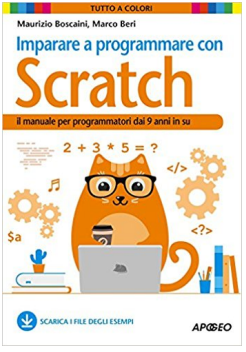
Who am I?

(mainly) I am a High School Computer Science Teacher

My personal and didactical preferences

I am a Book Author 1/2

Manuals to start coding



I am a Book Author 2/2

School Textbooks

I am a human being

with the dream to be connected and to connect People and Coding in Italy

...and in Tanzania and Ghana with the **Africa for Mauri Project**

MSOLWA ST. GASPARE BERTONI SECONDARY SCHOOL

PYCON AFRICA

About this talk

Courses and Talks, Teachers and Students

2009

- Extracurricular Pygame course (teacher: me)
- Talk "Pygame goes to school" at Pycon Italy 3 (speaker: me)

2018

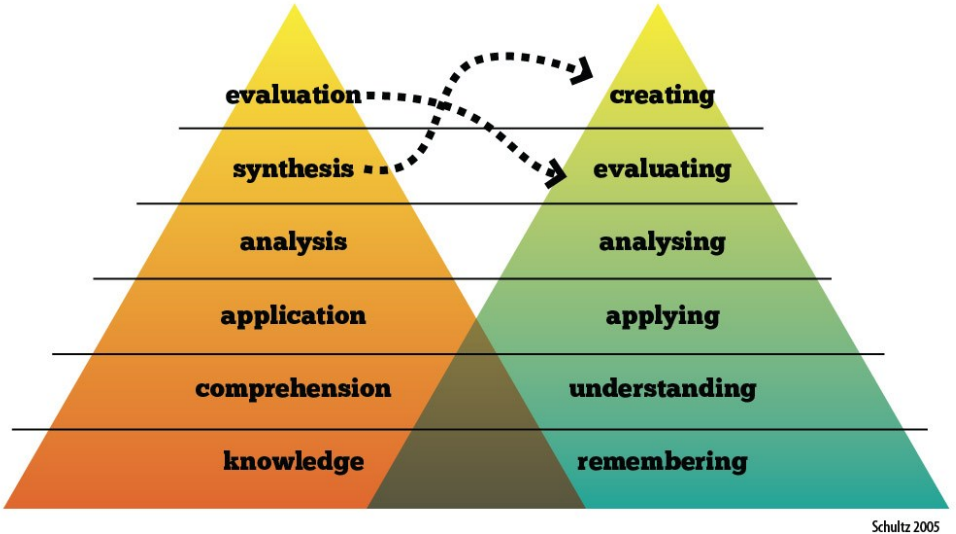
- Extracurricular Pygame course (teacher: my students Emanuele Feola, Alessandro Marchioro and Davide Tonin)
- Talk "Pygame back to school" at Pycon Italy 9 (speaker: Alessandro Marchioro and me)

Objectives

- **reflect** about **coding and computational thinking in education**
- **get to know** the basics of **pygame**
- **take a look** at a learning path through **simple video games**

2/4 - A didactical Storytelling

Bloom Taxonomy on learning objectives



Four fundamental abilities to learn in school

1. Reading
2. Writing
3. Mathematics
4. Computational thinking

Basic concepts and terms

A reference for coding

- Logic
- Algorithms
- Coding or Computer programming
- Computational Thinking
- Information Literacy

Logic

- Thought
- Language
- Mind

Algorithms

- A procedure to solve a problem
- A finite sequence of elementary steps to solve a class of problems
- ...

Coding or Computer programming

> The process of designing, writing, testing, debugging / troubleshooting, and maintaining the source code of computer programs (Wikipedia)

> The process to code our thoughts.

Computational Thinking

> Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can effectively be carried out by an information-processing agent.
>
> (Jeannette Wing, 2006)

Information Literacy

> ... the hyper ability to know when there is a need for information, to be able to identify, locate, evaluate, and effectively use that information for the issue or problem at hand.
>
> (The United States National Forum on Information Literacy in Wikipedia)

How to teach computational thinking?

References, Empirical Methodologies and Theories but No Algorithm

Coding could be a good method for touching and experimenting with **Math** and **Computational Thinking**

(Math and Logic) + Coding = Computational Thinking

3/4 Learning through games with Pygame

What is a video game?

About games and video games

> A game is a series of meaningful choices.
>
> (Sid Meier, designer and producer of the Civilization series)

> The possibilities are infinite!
>
> (Notch, Minecraft Developer)

> A video game is an electronic game that involves interaction with a user interface to generate visual feedback on a two- or three-dimensional video display device such as a TV screen, virtual reality headset or computer monitor.
>
> (Wikipedia)

Video game elements

1. Zero or more **players**
2. One or more **objectives** to achieve
3. A set of **rules** to follow
4. A series of **decisions** to make
5. A textual or graphical **environment** to interact with
6. A bit of **randomness**
7. Some **fun**

What is Pygame?

Pygame is a free and open source cross-platform Python library built on top of the SDL library for making multimedia applications like games.

Pygame modules

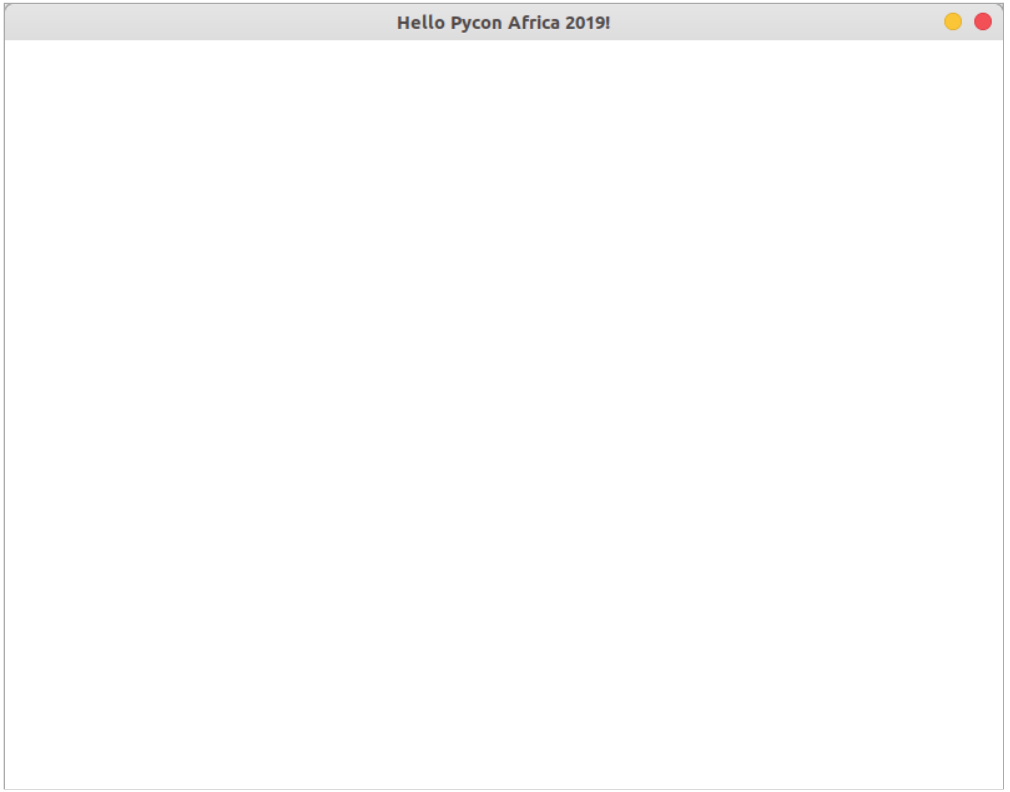
- cdrom
- cursors
- display
- draw
- event
- font
- image
- joystick
- key
- mouse
- movie
- sndarray
- surfarray
- time
- transform

Basic structure of a Pygame program

```
[0] Import general Python modules and Pygame specific modules
[1] Initialize Pygame modules
[2] Load game resources
[3] Setup game objects (display, graphics, sprites,...)
[4] Setup game-specific variables (counters for lives and score,...)

[5] Game Loop
    [6] Events' handling loop
    [7] Game logic (collisions, movements, ...)
    [8] Prepare the game scene on the image buffer
    [9] Draw the game scene on the display area on the screen

[10] Resources release and program exit
```

00 basic_program.py

```
# [0] Import general Python modules and Pygame specific modules
import sys
import pygame

WHITE = (255, 255, 255)

# [1] Initialize Pygame modules
pygame.init()

# [2] Load game resources
# [3] Setup game objects (display, graphics, sprites,...)
screen = pygame.display.set_mode((800, 600))
pygame.display.set_caption("Hello Pycon Africa 2019!")

# [4] Setup game specific variables (counters for lives and score,...)
# [5] Game Loop
while True:
    # [6] Events' handling loop
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit

    # [7] Game logic (collisions, movements, ...)
    # [8] Prepare the game scene on the image buffer
    screen.fill(WHITE)

    # [9] Draw the game scene on the display area on the screen
    pygame.display.update()
```


Hello Pycon Africa 2019!

Pygame modules, classes and objects

- pygame.Surface
- pygame.display
- pygame.draw.*
- pygame.image.*
- pygame.Rect
- pygame.time.Clock
- pygame.locals.*
- pygame.event
- pygame.font.Font
- pygame.sprite.Sprite
- pygame.sprite.Group
- pygame.mixer
- pygame.sprite.*
- ...

pygame.Surface

Class for representing images

Example of surface creation:

```
pygame.Surface( (64, 64) ) → Return a Surface of 64 x 64 pixels
```

```
surface.fill()  # Color a surface with an RGB color.
```

```
surface.blit()  # Draw a surface onto another surface.
```

01 ghana_flag_colors.py

```
# [...] modules import and constants declaration
# [1] Initialize the Pygame modules
pygame.init()
# [2] Load game resources
# [3] Setup game objects (display, graphics, sprites,...)
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption("Colors of the Ghana flag")
height = HEIGHT // 3
red_surface = pygame.Surface((WIDTH, height))
red_surface.fill(RED)
yellow_surface = pygame.Surface((WIDTH, height))
yellow_surface.fill(YELLOW)
green_surface = pygame.Surface((WIDTH, height))
green_surface.fill(GREEN)

# [4] Setup game specific variables (counters for lives and score,...)
# [5] Game Loop
while True:
    # [6] Events' handling loop
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
    # [7] Game logic (collisions, movements, ...)
    # [8] Prepare the game scene on the image buffer
    screen.blit(red_surface, (0, 0))
    screen.blit(yellow_surface, (0, height))
    screen.blit(green_surface, (0, 2 * height))
    # [9] Draw the game scene on the display area on the screen
    pygame.display.update()
```


Colors of the Ghana flag

pygame.display

Module to control the display window and screen

```
pygame.display.set_mode()  # Initialize a window or screen for display.
```

```
pygame.display.set_caption()  # Set the current window caption.
```

Example:

```
window_surface = pygame.display.set_mode( (800, 600) )
pygame.display.set_caption("PyCon Game")
```

pygame.draw.*

Module for drawing shapes

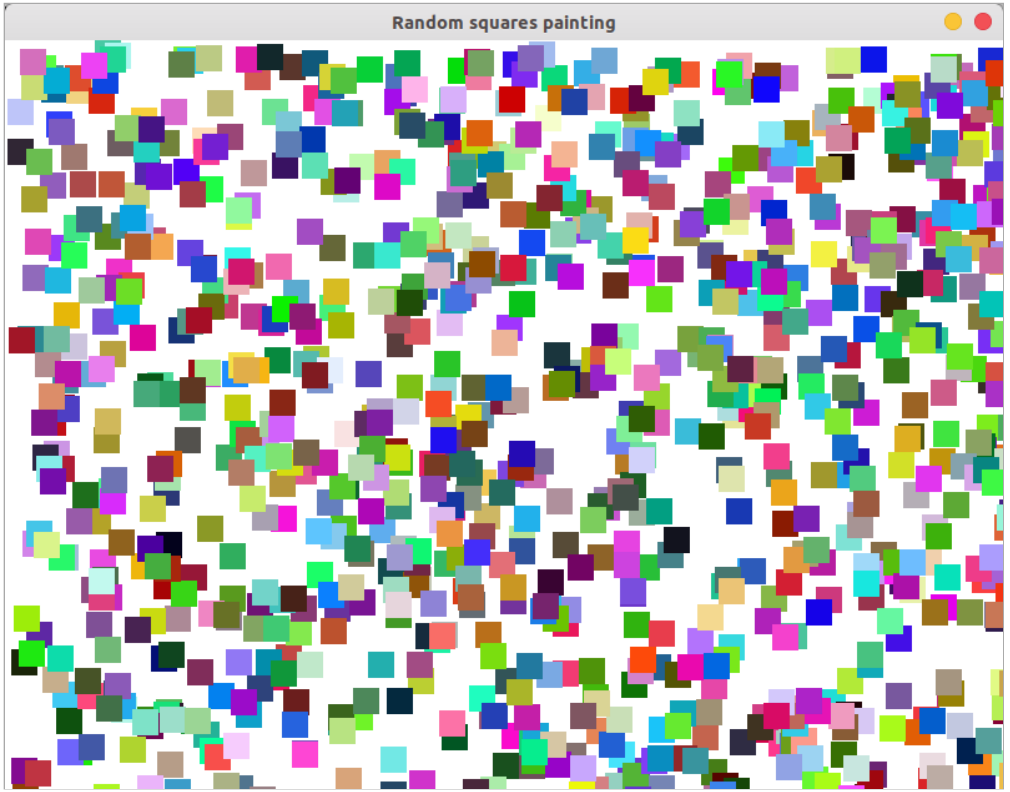
```
pygame.draw.line(Surface, color, start_pos, end_post, width=1)
```

```
pygame.draw.circle(Surface, color, pos, radius, width=0)
```

```
pygame.draw.polygon(Surface, color, pointlist, width=0)
```

02 random_squares.py

```
# [...] modules import and constants declaration
# [1] Initialize the Pygame modules
pygame.init()
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption("Random squares painting")

# [2] Load game resources
# [3] Setup game objects (display, graphics, sprites,...)
screen.fill(WHITE)

for _ in range(N_SQUARES):
    color = randrange(255), randrange(255), randrange(255)
    pos = randrange(WIDTH), randrange(HEIGHT)
    x, y = randrange(WIDTH), randrange(HEIGHT)
    pointlist = [(x, y), (x + SIDE, y), (x + SIDE, y + SIDE), (x, y + SIDE)]
    pygame.draw.polygon(screen, color, pointlist)

# [4] Setup game specific variables (counters for lives and score,...)
# [5] Game Loop
while True:
    # [6] Events' handling loop
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

    # [7] Game logic (collisions, movements, ...)
    # [8] Prepare the game scene on the image buffer
    # [9] Draw the game scene on the display area on the screen
    pygame.display.update()
```


Random squares painting

pygame.image.*

Module for image transfer

```
pygame.image.load('/path/to/file')  # → pygame.Surface
```

```
pygame.image.save(Surface, filename)
```

03 hello_africa.py

```
# [...] modules import
SIZE = WIDTH, HEIGHT = (800, 600)
WHITE = (255, 255, 255)
# [1] Initialize Pygame modules
pygame.init()
# [2] Load game resources
filename = os.path.join('res', 'images', 'logo-pycon-africa.png')
logo = pygame.image.load(filename)

# [3] Setup game objects (display, graphics, sprites,...)
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption("Hello, Pycon Africa 2019!")
x = (WIDTH - logo.get_width()) // 2
y = (HEIGHT - logo.get_height()) // 2

# [4] Setup game specific variables (counters for lives and score,...)
# [5] Game Loop
while True:
    # [6] Events' handling loop
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

    # [7] Game logic (collisions, movements, ...)
    # [8] Prepare the game scene on the image buffer
    screen.fill(WHITE)
    screen.blit(logo, (x, y))
    # [9] Draw the game scene on the display area on the screen
    pygame.display.update()
```


Hello, Pycon Africa 2019!
PYCON AFRICA

pygame.Rect

Object for storing rectangular coordinates.

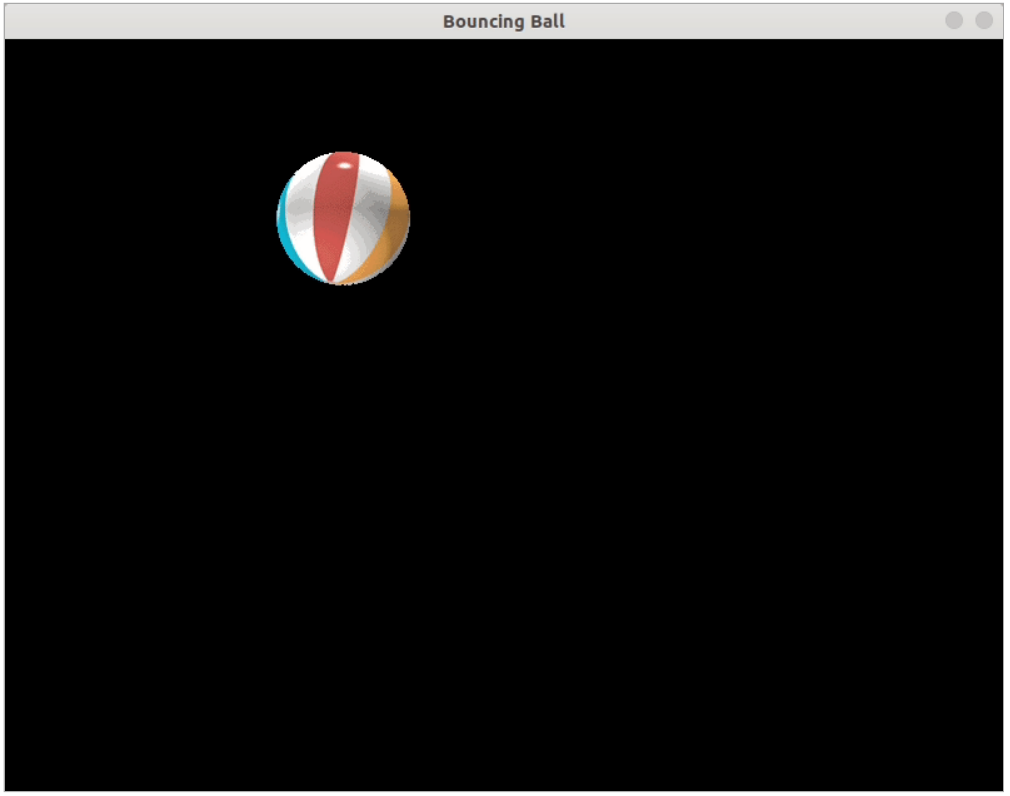
Rect Properties:

- x
- y
- width
- height
- center
- ...

```
surface.get_rect()  # → the rectangular area of the Surface
```

04 bouncing_ball.py

```
# [...] modules import and constants declaration
pygame.init()
filename = os.path.join('res', 'images', 'ball.gif')
ball = pygame.image.load(filename)
screen = pygame.display.set_mode(SIZE)
pygame.display.set_caption("Bouncing Ball")

# [4] Setup game specific variables (counters for lives and score,...)
speed = [2, 2] # Vectorial speed
ball_rect = ball.get_rect() # Rect object for the ball

while True:
    screen.fill(BLACK)
    ball_rect = ball_rect.move(speed)

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

    # [7] Game logic (collisions, movements, ...)
    if ball_rect.left < 0 or ballrect.right > WIDTH:
        speed[0] = -speed[0]
    if ball_rect.top < 0 or ball_rect.bottom > HEIGHT:
        speed[1] = -speed[1]

    screen.blit(ball, ball_rect)
    pygame.display.update()
    pygame.time.delay(5)  # A short delay
```


Bouncing Ball

pygame.time.Clock

Class to create an object to help track time getting a stable **frame rate**.

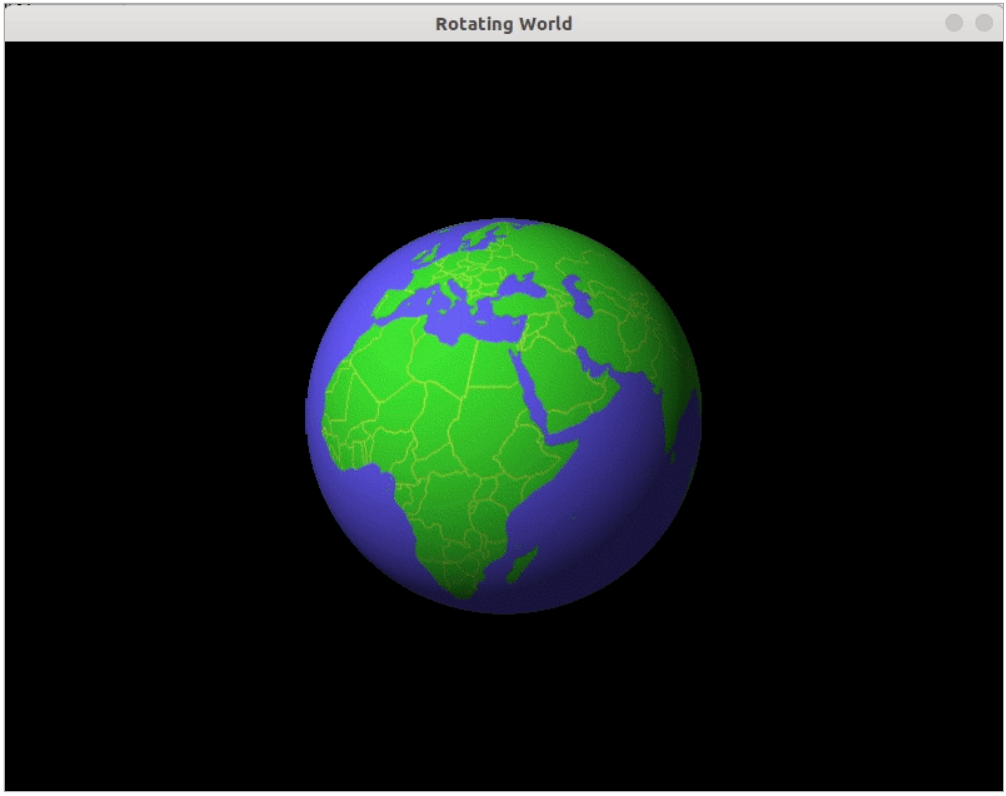
Common values of **FPS** (Frame Per Second): 24, 30, 60, 120, 144.

```
clock.tick(fps) # In the gameloop sets the maximum iteration frequency
```

```
clock.get_fps() # → the current fps value
```

05 image_animation.py

```
# [...] modules import
def main(name, n_images, fps=60, bg_color=BLACK, sep='-', zero_leading=False):
    pygame.init()
    # [2] Load game resources
    images = []
    for i in range(n_images):
        num = '{:02d}'.format(i) if zero_leading else str(i)
        filename = os.path.join('res', 'images', name, name + sep + num +'.png')
        image = pygame.image.load(filename)
        images.append(image)

    screen = pygame.display.set_mode(SIZE)
    pygame.display.set_caption("Rotating World")
    clock = pygame.time.Clock()

    x = (WIDTH - images[0].get_rect().width) // 2
    y = (HEIGHT - images[0].get_rect().height) // 2
    index = 0

    while True:
        # [...] Events management
        image = images[index]
        index = (index + 1) % n_images
        screen.fill(bg_color)
        screen.blit(image, (x, y))
        pygame.display.update()
        clock.tick(fps)

if __name__ == "__main__":
    main("Rotating_globe", 299, fps=20)
```


Rotating World

from pygame.locals import *

Module with a set of commonly used *constants* and functions from PyGame that have proven useful to put into your program's global namespace.

- QUIT
- FULLSCREEN
- ...
- K_UP
- ...
- K_a
- ...

pygame.event

Module for interacting with events and queues

```
# Event loop and control of the quit event
for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
```

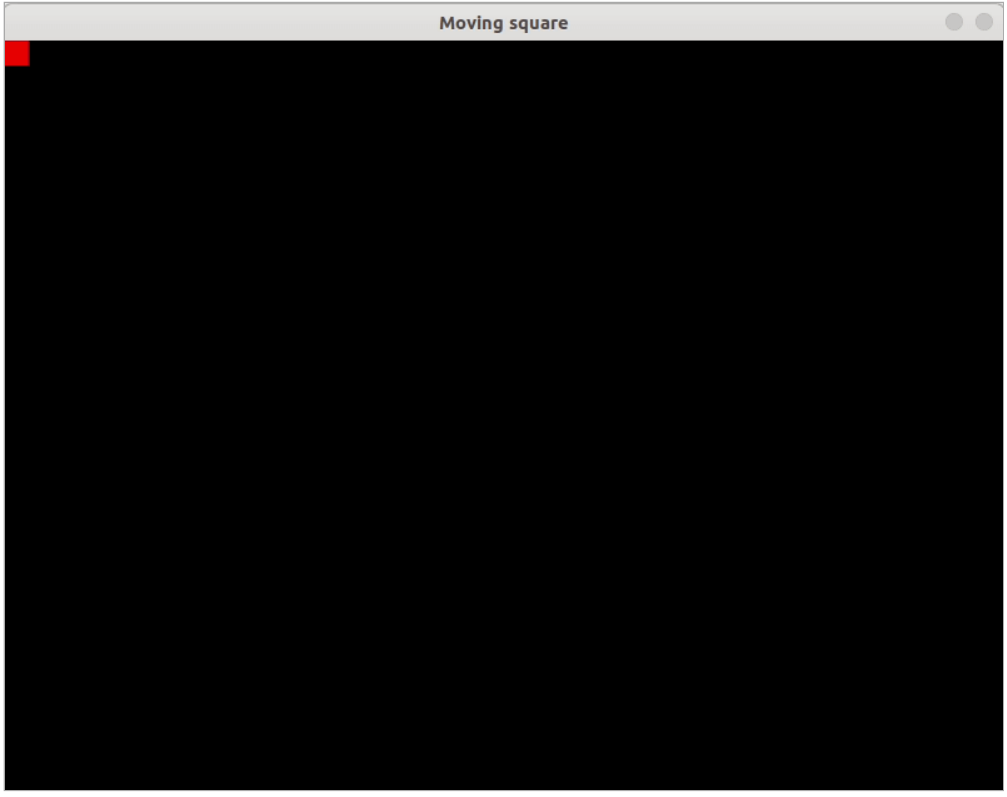
Keyboard control

```
    # Keyboard control (1st mode)
    keys = pygame.key.get_pressed()  # Get the state of all keyboard buttons

    if keys[K_UP]:
        pass
```

```
    # Keyboard control (2nd mode)
    if event.type == KEYDOWN:
        if event.key == K_ESCAPE:  # event.key has the ASCII value of the key
            pygame.quit()
        if event.key == K_a:
            pass
```

06 moving_square.py

```
# [...] modules import and initial settings
clock = pygame.time.Clock()
pygame.key.set_repeat(10, 10)
player = pygame.Rect((0, 0, 20, 20))
speed = 10
running = True

while running:
    for event in pygame.event.get():
        if event.type == QUIT:
            running = False
        # Move up/down/left/right by checking for pressed keys
        # and moving the player rect in-place
        keys = pygame.key.get_pressed()

        if keys[K_UP]:
            player.move_ip(0, -speed)  # Moves the rectangle, in place
        if keys[K_DOWN]:
            player.move_ip(0, speed)
        if keys[K_RIGHT]:
            player.move_ip(speed, 0)
        if keys[K_LEFT]:
            player.move_ip(-speed, 0)

    # Ensure the player rect does not go out of screen
    player.clamp_ip(screen.get_rect())
    screen.fill(BLACK)
    pygame.draw.rect(screen, RED, player)
    pygame.display.flip()
    clock.tick(FPS)
```


Moving square

Formula to calculate the movement per second of an object

```
distance = FPS * value
```

Example:

```
FPS = 60
speed = 10
distance = 10 px/frame * 60 frame/s = 600 px/s
```

A geography game

With a taste of functional programming

07 guess_the_flag.py

A list comprehension to load the data into a list.

```
nations = [nation.strip() for nation in open(filename)]
```

nations.txt

```
Afghanistan
Albania
Algeria
Andorra
Angola
Antigua_and_Barbuda
Saudi_Arabia
Argentina
Armenia
Australia
Austria
Azerbaijan
the_Bahamas
Bahrain
Bangladesh
Barbados
Belgium
[...]
```

Guess the flag

Question 2 / 5

Score: 1 / 5

[1] Uganda

Correct answer!
Press SPACE to continue

[2] Angola

[3] Estonia

A visual memory game

pygame.font.Font

Create a new Font object from a file.

Loading

```
filename = os.path.join('res', 'fonts', 'Boogaloo-Regular.ttf')
font = pygame.font.Font(filename, 40)
```

Rendering

```
font.render("...", True, BLACK) # Draw a text with antialias and return a surface
```

08 guess_the_animal

A simple example of **Deterministic finite automaton** (DFS).

Mouse control

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        running = False
    if event.type == MOUSEBUTTONDOWN:
        if state == 'guess the animal':
            pos = pygame.mouse.get_pos()
            choice = check_answer(pos, animals)

            if choice:
                state = "check answer"
```

Guess the most frequent animal!

Timer: 2.00 Score: 0/1

You Missed!

Press SPACE to continue ESC to quit

Into the OOP universe

09 random_cells.py: class Cell

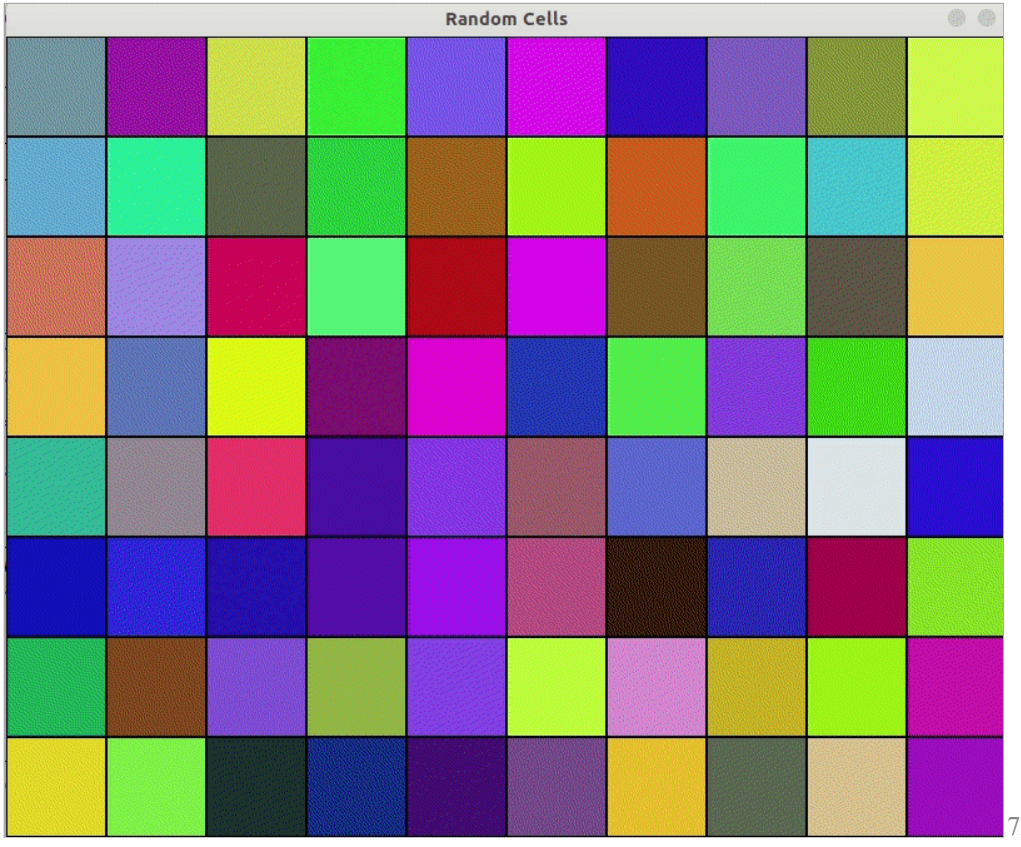
```
class Cell:
    """Represents a cell on a grid"""
    def __init__(self, row, col, border=1):
        self.row = row
        self.col = col
        self.border = border
        self.color = (randint(0, 255), randint(0, 255), randint(0, 255))

    def draw(self, surface):
        """Draw the cell on the given surface"""
        pygame.draw.rect(surface, self.color, (
            self.col * SIDE + self.border,
            self.row * SIDE + self.border,
            SIDE - self.border*2, SIDE - self.border*2
            )
        )
```

09 random_cells.py: class Grid

```
class Grid:
    """Represents a grid of cells"""
    def __init__(self, rows, cols):
        self.rows = rows
        self.cols = cols
        self.reset()

    def reset(self):
        """Create a new grid of cells"""
        self.grid = [
          [Cell(row, col) for col in range(self.cols)] for row in range(self.rows)
        ]

    def draw(self, surface):
        """Draw the grid on the given surface"""
        for row in range(self.rows):
            for col in range(self.cols):
                cell = self.grid[row][col]
                cell.draw(surface)
```

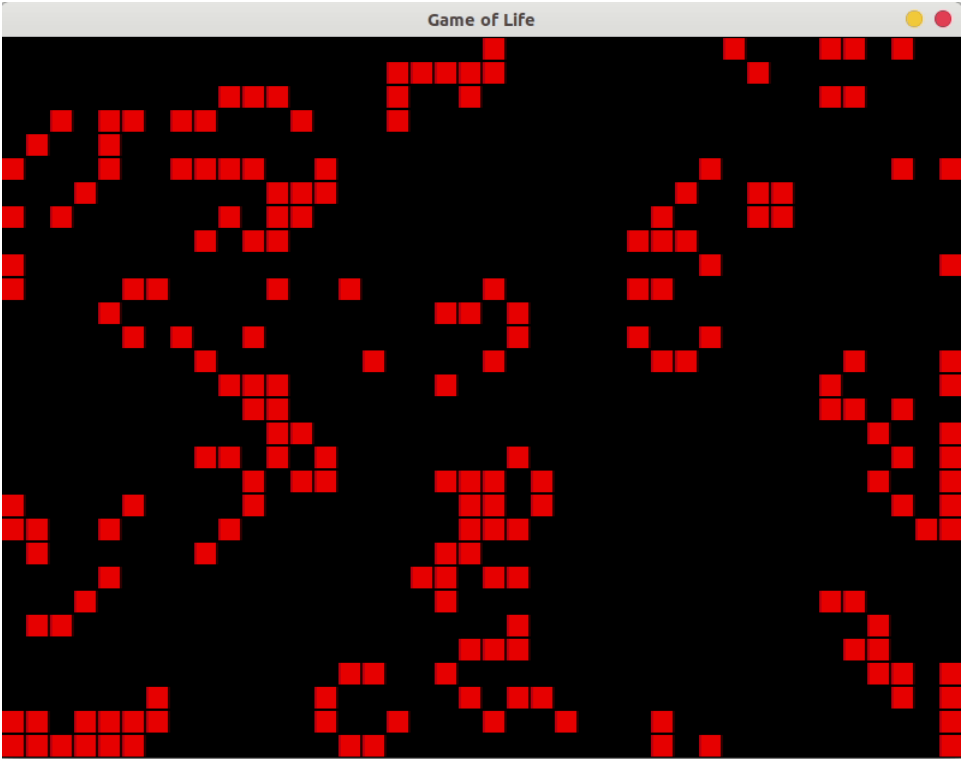
09 random_cells.py: main block

```
# [...] modules import and initial settings
pygame.init()
screen = pygame.display.set_mode(SIZE, HWACCEL|HWSURFACE|DOUBLEBUF)
pygame.display.set_caption('Random Cells')
clock = pygame.time.Clock()
_rows = HEIGHT // SIDE
_cols = WIDTH // SIDE
grid = Grid(_rows, _cols)
running = True

while running:
    for event in pygame.event.get():
        if event.type == QUIT:
            running = False
        if event.type == KEYDOWN:
            if event.key == K_ESCAPE:
                running = False
            if event.key == K_SPACE:
                grid.reset()

    grid.reset()
    grid.draw(screen)
    pygame.display.update()
    clock.tick(FPS)

pygame.quit()
```


Random Cells

A Zero-Player Game

10 game_of_life.py - Conway's Game of Life



pygame.sprite.Sprite

The base class for visible game objects.

- Derived classes will want to override the **Sprite.update**() method and assign **Sprite.image** and **Sprite.rect** attributes.
- The initializer can accept any number of **Group** instances that the Sprite will become a member of.

pygame.sprite.Group

Container class for Sprites.

```
draw(self, surface)  # Draw all sprites onto the surface
has(self, *sprites)  # Ask if the group has a sprite or sprites
```

Checking for exceptions

We should teach this topic...

```
def load_image(filename):
    '''Load and return an image from a file'''
    filename = os.path.join('res', 'images', filename)
    try:
        image = pygame.image.load(filename)
    except pygame.error as message:
        print("Cannot load image: " + filename)
        raise SystemExit(message)
    return image
```

pygame.mixer

Module for loading and playing sounds

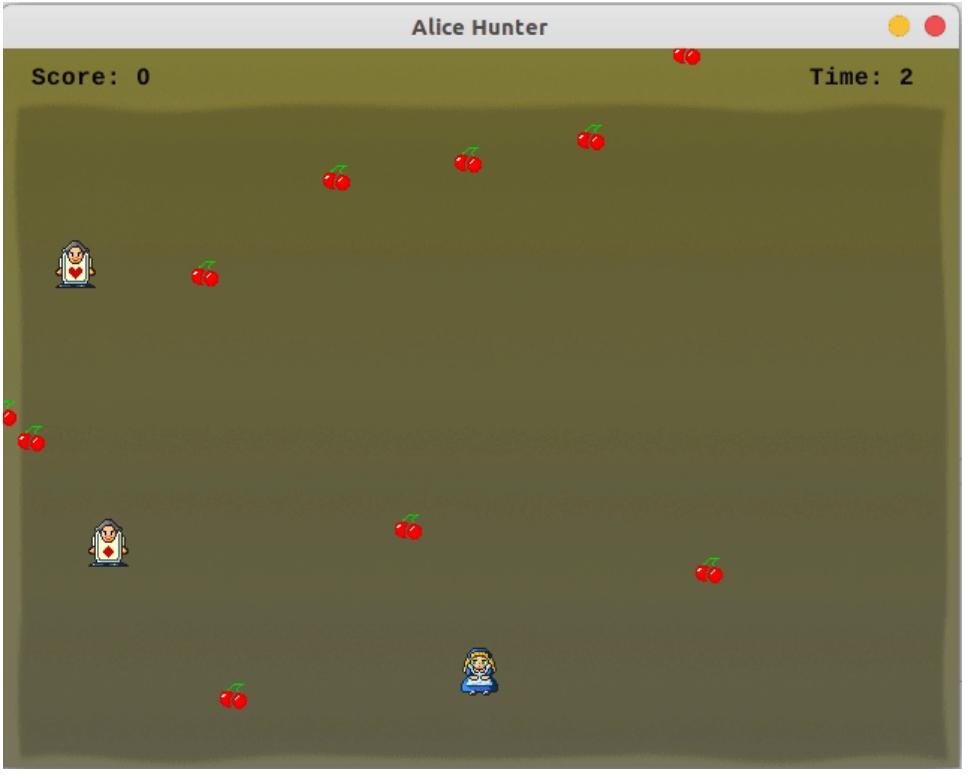
```
if pygame.mixer:
    music = os.path.join('res', 'sounds', 'background.wav')
    pygame.mixer.music.load(music)
    pygame.mixer.music.play(-1) # Background music
```

pygame.sprite.* and sprite collisions

```
# Create sprite groups
all = pygame.sprite.Group()
cherries = pygame.sprite.Group()
enemies = pygame.sprite.Group()
players = pygame.sprite.Group()

# Assign sprites to groups
Alice.containers = all, players
Enemy.containers = all, enemies
Cherry.containers = all, cherries
```

```
# Collisions detection
if pygame.sprite.groupcollide(enemies, players, 0, 1):
    return  # Game over
elif pygame.sprite.spritecollide(player, cherries, 1):
    score += 1
```

11 Alice



Good teaching is a quarter **preparation** and three-quarters **theater**

(Galileo Galilei)

The role of Pygame in high school

Objectives and Skills

- Coding principles
- Curiosity
- Community
- Fun
- Challenges

and more...

Other Pygame modules

- joystick
- transform
- gfxdraw
- music
- custom events
- ...

Other Python Game libraries

- https://wiki.python.org/moin/PythonGameLibraries

- pyglet written in pure Python
- Pygame Zero "...for creating games without boilerplate"

4/4 Conclusion

> The main task of education is to help to face what is hard and difficult so that our students can pass to the next level.

> Learn to code is hard.

> Teach to code is difficult.

> Pygame is a good option to face this challenge.

Future develpment

- Pygame in the curriculum
- Pygame school contest
- Surveys and tests on the learning outcomes

Questions?

Thanks

Credits: these slides are made with:

- remarkjs converts *markdown* text into HTML5 slides
- embedmd embeds files or fractions of files into Markdown files

References

- PyGame va a Scuola, Maurizio Boscaini, Pycon Italia 3, Florence - April 2009
- PyGame torna a Scuola, Alessandro Marchiori (and Maurizio Boscaini), Pycon Italia 9, Florence - April 2018
- Ricomincio-da-bloom, Gianfranco Marini, 2015
- https://github.com/marcioz98/PyGameGoesBackToSchool
- https://en.wikipedia.org/wiki/Coding
- https://en.wikiquote.org/wiki/Sid_Meier