PROGRAMACIÓN DE VIDEO JUEGOS

TETRIS

POR:

JUAN RODRIGO CALLO HUAYNA

RODRIGO MIGUEL PALACIOS SALAS

MAURICIO SEBASTIAN VIRRUETA MARQUEZ

¿QUÉ ES TETRIS?

Un video juego implementado en la época de la unión sovietica, fue creado por Alekséi Pázhitnov y fue publicado un 6 de junio de 1984

1 ¿EN QUÉ CONSISTE?

Consiste en que varias piezas geométricas caen para ir rellenando la ventana, el objetivo es formar líneas para evitar que la ventana se llene

2 JUGABILIDAD

Consiste en jugar con las flechas de tu teclado para mover las fichas y cambiarles la direccion

3 LA EVOLUCIÓN

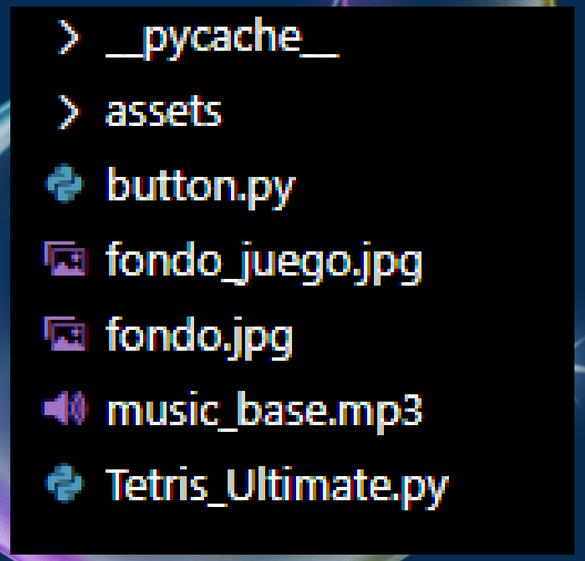
Existen muchas versiones del juego de las cuales las mas resaltadas es tetris 99

4 ¿POR QUÉ LO ESCOJIMOS?

La mayoria de nosotros conocemos y recordamos este juego con nostalgia, por el hecho que fue uno de los primeros y uno de los mas jugados.

¿QUE ES PYGAME?

Pygame es una biblioteca de python que principalmente esta orientada a la programación de video juegos en 2D




```python
import pygame
import random
import sys
from button import Button
colors = [
    (1, 1, 128),            #Azul
    (150, 50, 255),          #Morado
    (255, 128, 1),          #Naranja
    (255, 255, 1),          #Amarillo
    (25, 255, 255),          #Celeste
    (1, 255, 1),            #Verde
    (255, 1, 1),            #Rojo
]
```

```python
class Figure:
    figures = [
    [[1,5,9,13],[4,5,6,7]],                     #I
    [[1,4,5,6],[1,2,5,6],[4,5,6,10],[1,5,8,9]],     #J
    [[1,4,5,6],[1,5,6,9],[4,5,6,9],[1,4,5,9]],      #T
    [[2,4,5,6],[1,5,9,10],[4,5,6,8],[0,1,5,9]],     #L
    [[1,2,5,6]],                                #O
    [[1,2,4,5],[0,4,5,9]],                      #S
    [[0,1,5,6],[1,4,5,8]],                      #Z
    ]
    def __init__(self, x, y):...
    def image(self):                #Forma de la figura...
    def rotate(self):               #Rotacion de la figura...
class Tetris:
    def __init__(self, height, width):...
    def new_figure(self):...
    def intersects(self):...
    def break_lines(self):...
    def go_space(self):...
    def go_down(self):...
    def freeze(self):...
    def go_side(self, mov_x):...
    def rotate(self):...
class Juego:
    def __init__(self):...
    def quit(self):...
    def play(self,mievento):...
    def menu(self):...
def main():...
main()
```

```python
class Button():
    def __init__(self, image, pos, text_input, font, base_color, hovering_color):...

    def update(self, screen):...

    def checkForInput(self, position):...

    def changeColor(self, position):...
```

CLASES IMPLEMENTADAS

clase Button

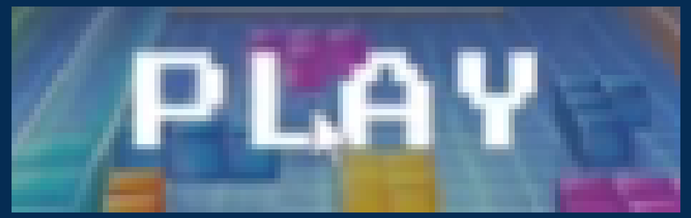
```python
class Button():
    def __init__(self, image, pos, text_input, font, base_color, hovering_color):
        self.image = image
        self.x_pos = pos[0]
        self.y_pos = pos[1]
        self.font = font
        self.base_color, self.hovering_color = base_color, hovering_color
        self.text_input = text_input
        self.text = self.font.render(self.text_input, True, self.base_color)
        if self.image is None:
            self.image = self.text
        self.rect = self.image.get_rect(center=(self.x_pos, self.y_pos))
        self.text_rect = self.text.get_rect(center=(self.x_pos, self.y_pos))

    def update(self, screen):
        if self.image is not None:
            screen.blit(self.image, self.rect)
        screen.blit(self.text, self.text_rect)
```

CLASES IMPLEMENTADAS

clase Button

```
19
20      def checkForInput(self, position):
21          if position[0] in range(self.rect.left, self.rect.right) and
22              return True
23          return False
24
25      def changeColor(self, position):
26          if position[0] in range(self.rect.left, self.rect.right) and
27              self.text = self.font.render(self.text_input, True, self.
28          else:
29              self.text = self.font.render(self.text_input, True, self.
```




```
 and position[1] in range(self.rect.top, self.rect.bottom


 and position[1] in range(self.rect.top, self.rect.bottom
self.hovering_color)

self.base_color)
```

IMPORTS Y COLORES

```
import pygame
import random
import sys
from button import Button
colors = [
    (1, 1, 128),              #Azul
    (150, 50, 255),            #Morado
    (255, 128, 1),            #Naranja
    (255, 255, 1),            #Amarillo
    (25, 255, 255),            #Celeste
    (1, 255, 1),              #Verde
    (255, 1, 1),              #Rojo
]
```

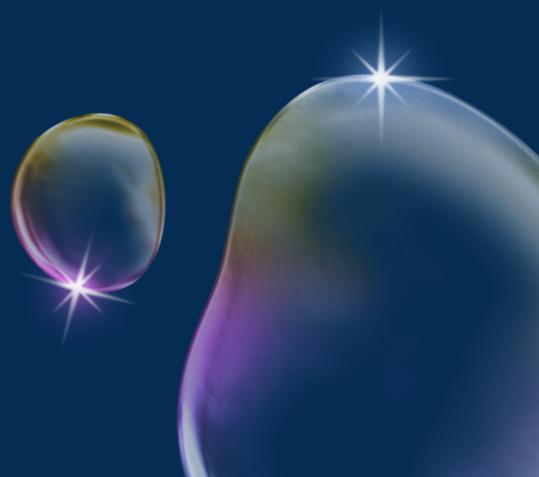
CLASES IMPLEMENTADAS

clase figura

```
class Figure:
    figures = [
    [[1,5,9,13],[4,5,6,7]],                                  #I
    [[1,4,5,6],[1,2,5,6],[4,5,6,10],[1,5,8,9]],              #J
    [[1,4,5,6],[1,5,6,9],[4,5,6,9],[1,4,5,9]],               #T
    [[2,4,5,6],[1,5,9,10],[4,5,6,8],[0,1,5,9]],              #L
    [[1,2,5,6]],                                             #O
    [[1,2,4,5],[0,4,5,9]],                                   #S
    [[0,1,5,6],[1,4,5,8]],                                   #Z
    ]
    def __init__(self, x, y):
        self.x =x                       #Posicion 3 de (0-9)
        self.y = y                       #Posicion 0 de (0-19)
        self.type=random.randint(0, len(self.figures) - 1) #Tipo figura
        self.rotation = 0                #Rotacion inicial
    def image(self):                     #Forma de la figura
        return self.figures[self.type][self.rotation]
    def rotate(self):                    #Rotacion de la figura
        self.rotation = (self.rotation + 1) % len(self.figures[self.type])
```

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15



clase Tetris

```
class Tetris:
    def __init__(self, height, width):
        self.level = 2
        self.score = 0
        self.state = "start"
        self.field = []
        self.x = 100
        self.y = 50
        self.zoom = 20
        self.figure = None
        self.height = height
        self.width = width
        for i in range(height):
            new_line = []
            for j in range(width):
                new_line.append(-1)
            self.field.append(new_line)
```

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

```
    def new_figure(self):
        self.figure = Figure(3, 0)
    def intersects(self):
        intersection = False
        for i in range(4):
            for j in range(4):
                if i * 4 + j in self.figure.image():
                    if i + self.figure.y > self.height - 1 or
                        intersection = True
        return intersection
```

[2,4,5,6]

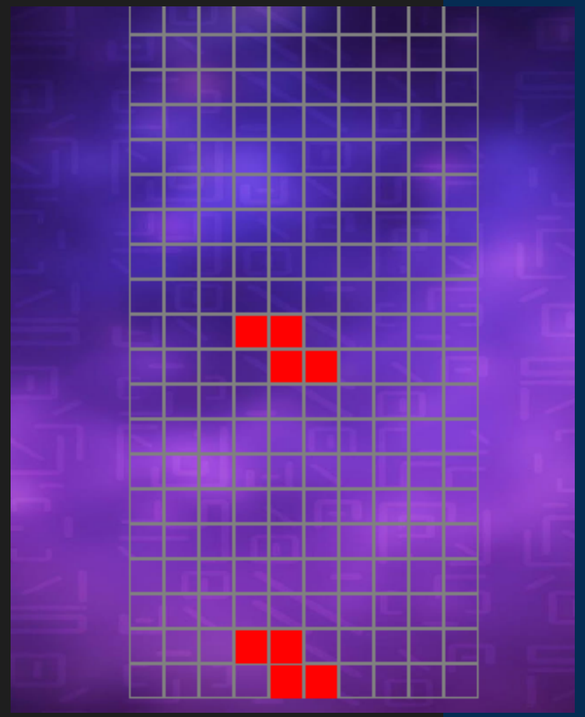
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

```
or j + self.figure.x < 0 or self.field[i + self.figure.y][j + self.figure.x] >= 0:
```

```python
def break_lines(self):
    lines = 0
    for i in range(1, self.height):
        zeros = 0
        for j in range(self.width):
            if self.field[i][j] == -1:
                zeros += 1
        if zeros == 0:
            lines += 1
            for i1 in range(i, 1, -1):
                for j in range(self.width):
                    self.field[i1][j] = self.field[i1 - 1][j]
    self.score += lines ** 2
def go_space(self):
    while not self.intersects():
        self.figure.y += 1
    self.figure.y -= 1
    self.freeze()
def go_down(self):
    self.figure.y += 1
    if self.intersects():
        self.figure.y -= 1
        self.freeze()
```

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	3	4	1	2	3	4	5	6	3

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1



```
def freeze(self):
    for i in range(4):
        for j in range(4):
            if i * 4 + j in self.figure.image():
                self.field[i + self.figure.y][j + self.figure.x] = self.figure.type
    self.break_lines()
    self.new_figure()
    if self.intersects():
        self.state = "gameover"
def go_side(self, mov_x):
    old_x = self.figure.x
    self.figure.x += mov_x
    if self.intersects():
        self.figure.x = old_x
def rotate(self):
    old_rotation = self.figure.rotation
    self.figure.rotate()
    if self.intersects():
        self.figure.rotation = old_rotation
```

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	3	-1	-1	-1
-1	-1	-1	-1	3	3	3	-1	-1	-1

clase Juego

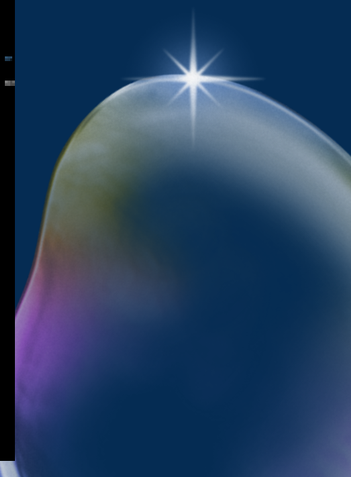
```
class Juego:
    def __init__(self):…
    def quit(self):…
    def play(self,mievento):…
    def menu(self):…
```

```
def __init__(self):
    self.size=(400, 500)
    self.screen = pygame.display.set_mode(self.size)
    pygame.display.set_caption("Tetris")
    self.fondo=pygame.image.load('fondo.jpg')
    self.fondo=pygame.transform.scale(self.fondo,self.size)
    pygame.mixer.music.load("music_base.mp3")
    self.done=False
    self.fps=5
    self.contador=0
    self.pressing_down=False
```

```
class Juego:
    def __init__(self):...
    def quit(self):...
    def play(self,mievento):...
    def menu(self):...
```

```
def quit(self):
    pygame.quit()
    sys.exit()
```

```
def menu(self):
    pygame.mixer.music.play(9)
    while not self.done:
        self.screen.blit(self.fondo,(0,0))
        MENU_MOUSE_POS = pygame.mouse.get_pos()
        PLAY_BUTTON = Button(image=pygame.transform.scale(pygame.image.load("assets/Play Rect.png"),(200,75)
        |   |   |   |   text_input="PLAY", font=pygame.font.Font("assets/font.ttf", 30), base_color="Black",
        QUIT_BUTTON = Button(image=pygame.transform.scale(pygame.image.load("assets/Quit Rect.png"),(200,75)
        |   |   |   |   text_input="QUIT", font=pygame.font.Font("assets/font.ttf", 30), base_color="Black",
        for button in [PLAY_BUTTON,QUIT_BUTTON]:
            button.changeColor(MENU_MOUSE_POS)
            button.update(self.screen)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.quit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if PLAY_BUTTON.checkForInput(MENU_MOUSE_POS):
                    self.play(event)
                if QUIT_BUTTON.checkForInput(MENU_MOUSE_POS):
                    self.quit()
        pygame.display.update()
```



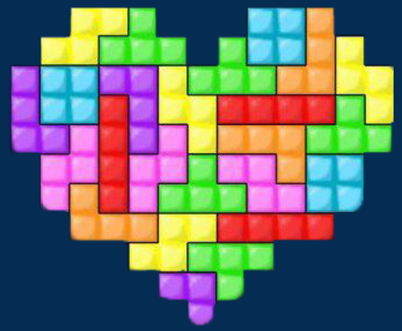
```
class Juego:
    def __init__(self):...
    def quit(self):...
    def play(self,mievento):...
    def menu(self):...
```

```
def play(self,mievento):
    clock=pygame.time.Clock()
    self.new_fondo=pygame.image.load('fondo_juego.jpg')
    self.new_fondo=pygame.transform.scale(self.new_fondo,self.size)
    game=Tetris(20,10)
```

```
while True:
    self.screen.blit(self.new_fondo,(0,0))
    if game.figure is None:
        game.new_figure()
    self.contador=+1
    if self.contador>1000000:
        self.contador=0
    if self.contador % (self.fps // game.level // 2) == 0 or self.pressing_down:
        if game.state == "start":
            game.go_down()
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.quit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                game.rotate()
            if event.key == pygame.K_DOWN:
                self.pressing_down = True
            if event.key == pygame.K_LEFT:
                game.go_side(-1)
            if event.key == pygame.K_RIGHT:
                game.go_side(1)
            if event.key == pygame.K_SPACE:
                game.go_space()
            if event.key == pygame.K_ESCAPE:
                game.__init__(20, 10)
```

```
        if mievento.type == pygame.KEYUP:
            if mievento.key == pygame.K_DOWN:
                self.pressing_down = False
        for i in range(game.height):#gradilla
            for j in range(game.width):
                pygame.draw.rect(self.screen, (128,128,128), [game.x + game.zoom * j, game.y + game.zoom * i
                if game.field[i][j] >= 0:
                    pygame.draw.rect(self.screen, colors[game.field[i][j]],
                                     [game.x + game.zoom * j + 1, game.y + game.zoom * i + 1, game.zoom - 2,
        if game.figure is not None:#pintar pantalla
            for i in range(4):
                for j in range(4):
                    p = i * 4 + j
                    if p in game.figure.image():
                        pygame.draw.rect(self.screen, colors[game.figure.type],
                                         [game.x + game.zoom * (j + game.figure.x) + 1,
                                         game.y + game.zoom * (i + game.figure.y) + 1,
                                         game.zoom - 2, game.zoom - 2])
        font = pygame.font.SysFont('Arial', 25, True, False)
        font1 = pygame.font.SysFont('Arial', 65, True, False)
        text = font.render("Score: " + str(game.score), True, (1,1,250))
        text_game_over = font1.render("Game Over", True, (125, 125, 125))
        text_game_over1 = font1.render("Press ESC", True, (0, 0, 0))
        self.screen.blit(text, [5, 5])
        if game.state == "gameover":
            self.screen.blit(text_game_over, [20, 200])
            self.screen.blit(text_game_over1, [25, 265])
        pygame.display.update()
        clock.tick(self.fps)
```

```
def main():
    pygame.init()
    miJuego=Juego()
    miJuego.menu()
    pygame.quit()
main()
```



PARTES TRABAJADAS POR LOS ALUMNOS

Mauricio Sebastian Virrueta Marquez

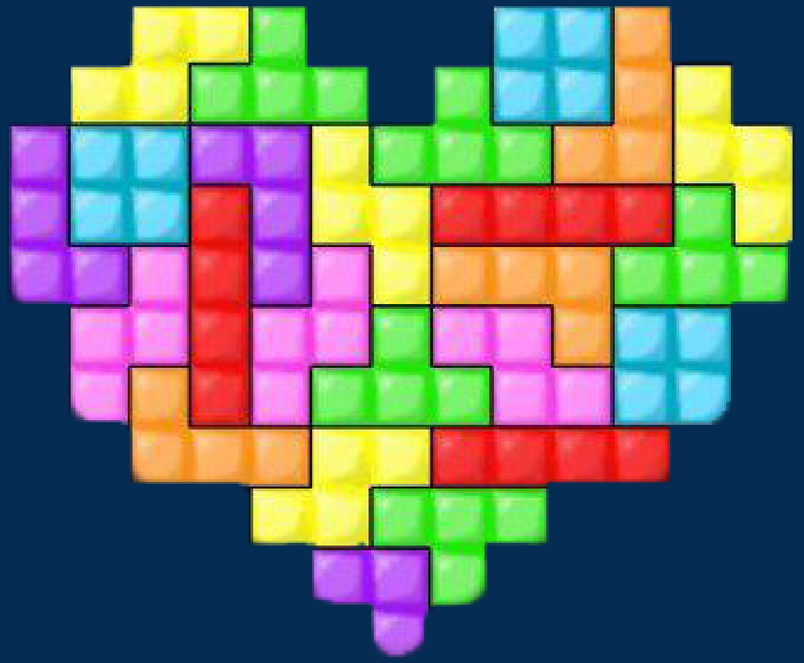
ESTRUCTURA BÁSICA

Juan Rodrigo Callo Huayna

DISEÑO DE CLASES

Rodrigo Miguel Palacios Salas

PINTADO , MUSICA E INTERFAZ DEL JUEGO



GRACIAS