**A blue logo with a arrow and a symbol

Description automatically generatedТехнически Университет**

**гр.Варна**

**Факултет: ФИТА**

**Катедра: СИТ**

Курсов проект по дисциплина:

„Графични системи“

Студент:

Теодор Асенов Иванов

ФН:21621650

гр: 1а

**Задание: Проект „Тетрис“**

Създайте програма, представляваща играта „Тетрис“. Програмата трябва да включва различни опции за трудност (оразмеряване на прозореца).

**Разработка**

Проектът "Тетрис" е създаден като модерна интерпретация на класическата игра „Тетрис“, като използва програмния език Python и библиотеката Pygame. Основната цел на играта е подреждането на падащи тетромино фигури така, че да се запълват хоризонтални редове без празни пространства. При запълване на ред, той бива изтрит, като се освобождава място за нови фигури.По този начин потребителят събира точки. Играта приключва, когато нова фигура не може да се побере в игралното поле.

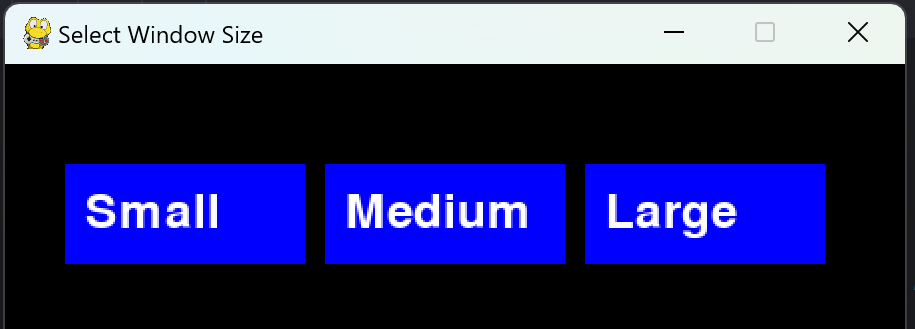
Особеност на разработката е възможността за избор на размер на прозореца на играта, която предлага различни нива на трудност и адаптивност. По-големите размери на прозореца предлагат по-голямо игрално поле и обратно, по-малките размери увеличават трудността.

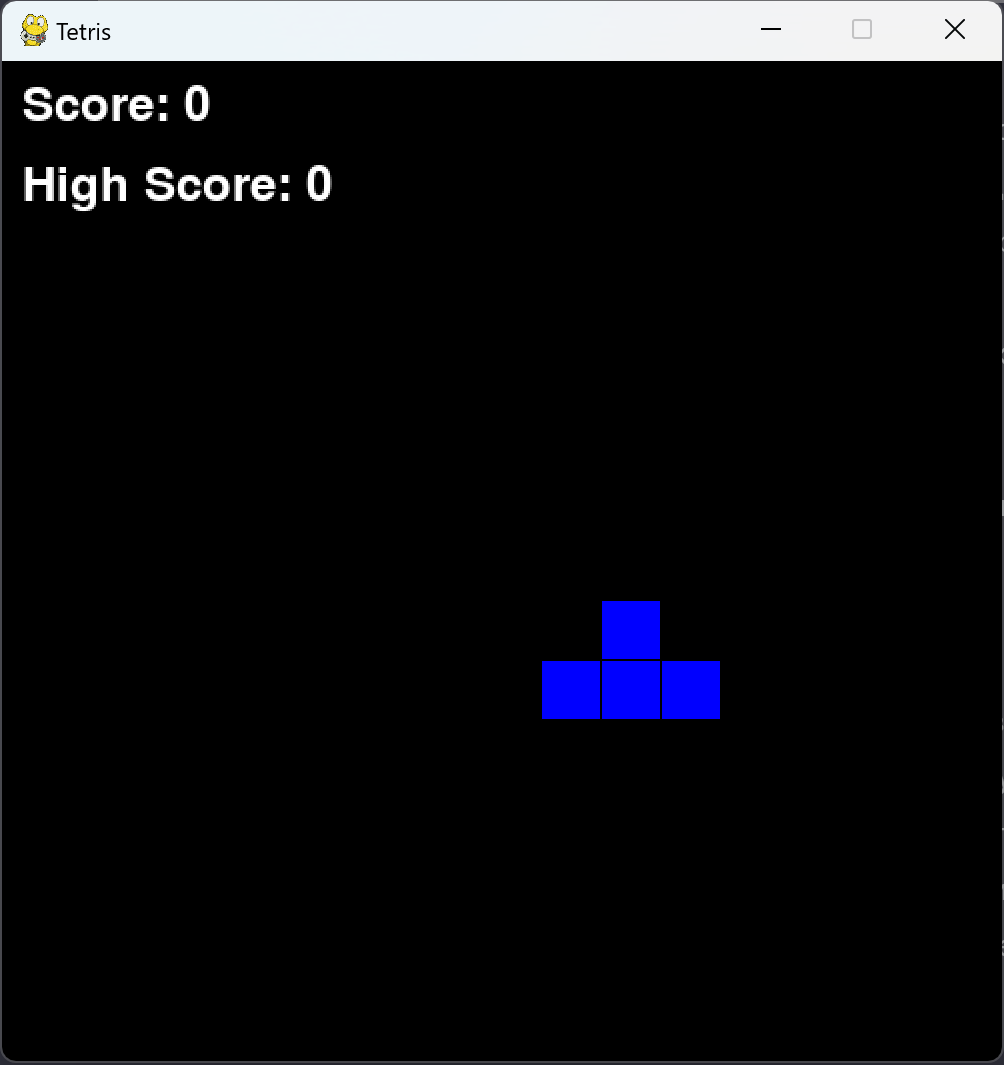
**Модули**

1. **Главно Меню (main\_menu.py):**
   1. Инициализация на **Pygame** и конфигурация на прозореца.
   2. Възможност за избор на размер на прозореца (Малък, Среден, Голям)
2. **Игрална Логика (tetris\_game.py):**
   1. Определение на цветове и форми на тетромино фигурите.
   2. Клас Shape за представяне на индивидуалните фигури.
   3. Клас Tetris за управление на игралното поле, движението на фигурите, роверка за валидни ходове, изтриване на редове и точкуване.
   4. Функции за визуализация на резултати и съобщение за край на играта.
   5. Управление на най-високия резултат чрез файлова система.

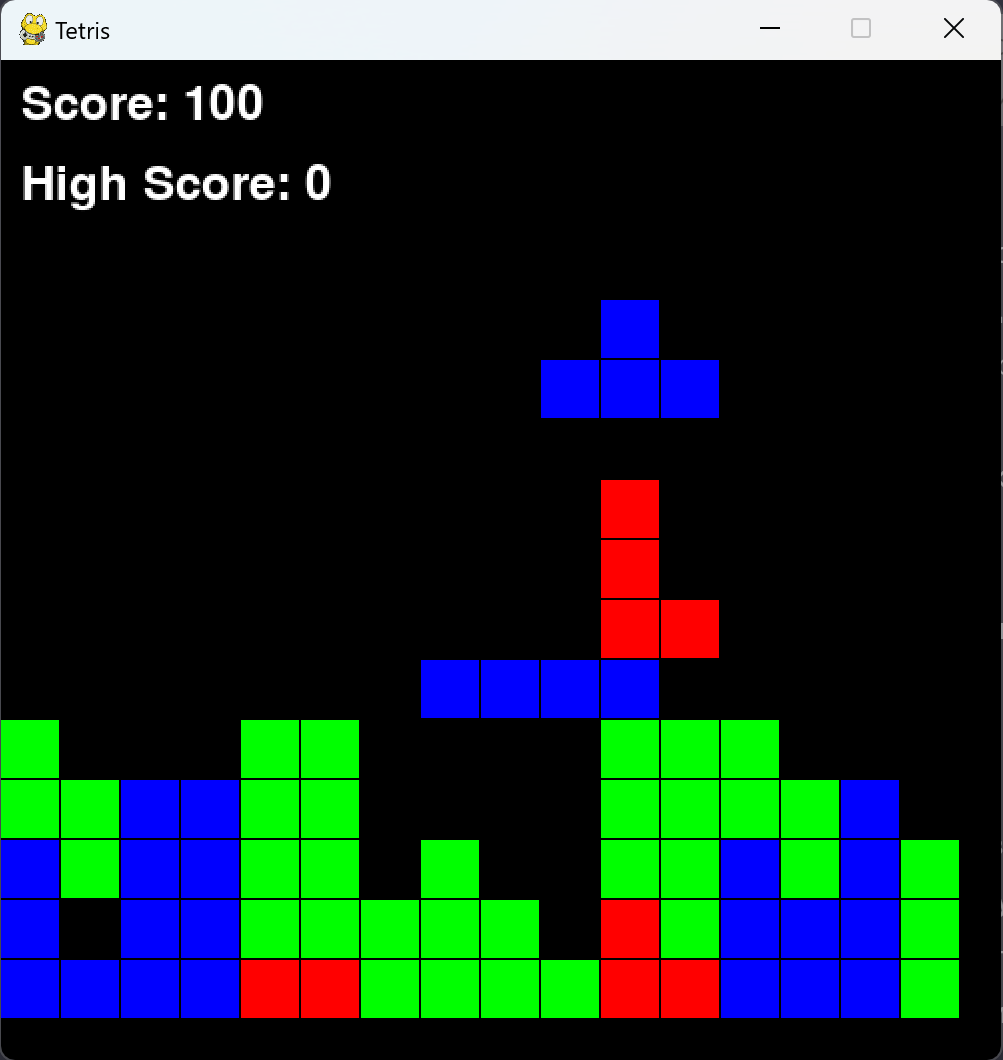
**Резултати**

Показване на начален прозорец с различни опции за трудност при стартиране на играта:

**фиг. 1: Главен прозорец**

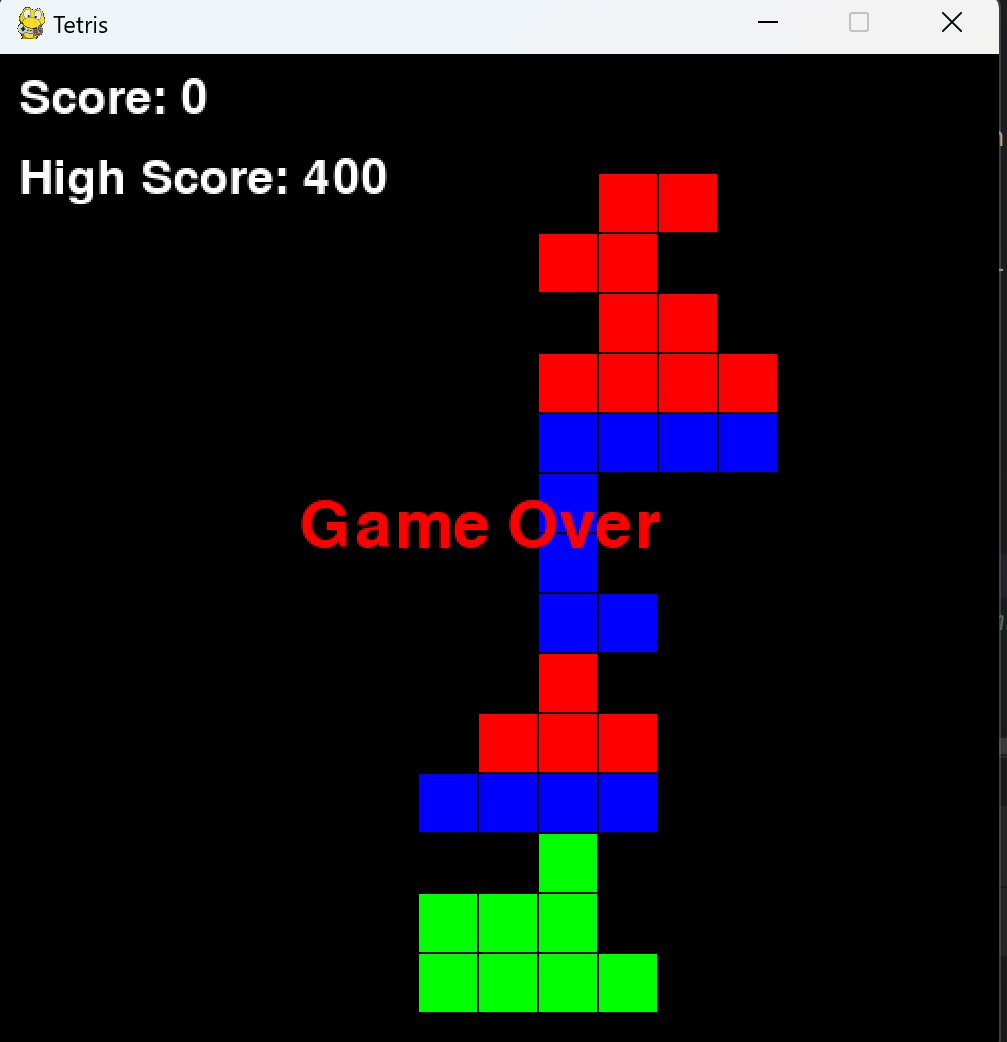
****Реализация на игралното поле при започване на играта

**фиг. 2: Игрално поле**

****Трупане на фигури, запълване на редове, събиране на точки

**фиг 3: Поведение на фигурите и точкуването в игралното поле**

Край на играта, когато полето се претрупа вертикално

**фиг. 4: Край на играта**

**Приложение**

**1. Сорс код**

**1.1 menu.py**

**from tetris\_game import run\_game**

**import pygame**

**import sys**

**import os**

**WINDOW\_SIZES = {**

**"Small": "400x400",**

**"Medium": "500x500",**

**"Large": "600x600"**

**}**

**def resource\_path(relative\_path):**

**try:**

**base\_path = sys.\_MEIPASS**

**except Exception:**

**base\_path = os.path.abspath(".")**

**return os.path.join(base\_path, relative\_path)**

**def draw\_button(screen, text, position, size):**

**font = pygame.font.Font(None, 36)**

**text\_render = font.render(text, True, (255, 255, 255))**

**x, y, width, height = position**

**pygame.draw.rect(screen, (0, 0, 255), position)**

**screen.blit(text\_render, (x + 10, y + height / 4))**

**def setup\_screen():**

**pygame.init()**

**screen = pygame.display.set\_mode((450, 150))**

**pygame.display.set\_caption("Select Window Size")**

**buttons = {**

**"Small": ((30, 50, 120, 50), WINDOW\_SIZES["Small"]),**

**"Medium": ((160, 50, 120, 50), WINDOW\_SIZES["Medium"]),**

**"Large": ((290, 50, 120, 50), WINDOW\_SIZES["Large"])**

**}**

**running = True**

**while running:**

**screen.fill((0, 0, 0))**

**for event in pygame.event.get():**

**if event.type == pygame.QUIT:**

**pygame.quit()**

**sys.exit()**

**elif event.type == pygame.MOUSEBUTTONDOWN:**

**x, y = event.pos**

**for text, size in buttons.items():**

**pos = buttons[text][0]**

**if pos[0] <= x <= pos[0] + pos[2] and pos[1] <= y <= pos[1] + pos[3]:**

**pygame.quit()**

**width, height = map(int, WINDOW\_SIZES[text].split('x'))**

**run\_game(width, height)**

**return**

**for text, (pos, size) in buttons.items():**

**draw\_button(screen, text, pos, size)**

**pygame.display.flip()**

**pygame.quit()**

**if \_\_name\_\_ == "\_\_main\_\_":**

**setup\_screen()**

**1.2 tetris\_game.py**

import random

import sys

import pygame

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

RED = (255, 0, 0)

BLUE = (0, 0, 255)

GREEN = (0, 255, 0)

COLORS = [RED, BLUE, GREEN]

SHAPES = [

[

['.....',

'.....',

'.....',

'.OOOO',

'.....'],

['.....',

'..O..',

'..O..',

'..O..',

'..O..']

],

[

['.....',

'.....',

'..OO.',

'..OO.',

'.....'],

],

[

['.....',

'.....',

'..O..',

'.OOO.',

'.....'],

['.....',

'..O..',

'.OO..',

'..O..',

'.....'],

['.....',

'.....',

'.OOO.',

'..O..',

'.....'],

['.....',

'..O..',

'..OO.',

'..O..',

'.....']

],

[

['.....',

'.....',

'..OO.',

'.OO..',

'.....'],

['.....',

'.....',

'.OO..',

'..OO.',

'.....'],

['.....',

'.O...',

'.OO..',

'..O..',

'.....'],

['.....',

'..O..',

'.OO..',

'.O...',

'.....']

],

[

['.....',

'..O..',

'..O.',

'..OO.',

'.....'],

['.....',

'...O.',

'.OOO.',

'.....',

'.....'],

['.....',

'.OO..',

'..O..',

'..O..',

'.....'],

['.....',

'.....',

'.OOO.',

'.O...',

'.....']

],

]

class Shape:

def \_\_init\_\_(self, x, y, shape):

self.color = random.choice(COLORS)

self.shape = shape

self.x = x

self.y = y

self.rotation = 0

class Tetris:

def \_\_init\_\_(self, width, height):

self.width = width

self.height = height

self.grid = [[0 for \_ in range(width)] for \_ in range(height)]

self.current\_piece = self.new\_piece()

self.game\_over = False

self.score = 0

def new\_piece(self):

shape = random.choice(SHAPES)

return Shape(self.width // 2, 0, shape)

def valid\_move(self, piece, x, y, rotation):

for i, row in enumerate(piece.shape[(piece.rotation + rotation) % len(piece.shape)]):

for j, cell in enumerate(row):

if cell == 'O':

if not (0 <= piece.x + j + x < self.width) or \

not (0 <= piece.y + i + y < self.height) or \

(self.grid[piece.y + i + y][piece.x + j + x] != 0):

return False

return True

def clear\_lines(self):

lines\_cleared = 0

for i, row in enumerate(self.grid):

if all(cell != 0 for cell in row):

lines\_cleared += 1

del self.grid[i]

self.grid.insert(0, [0 for \_ in range(self.width)])

return lines\_cleared

def lock\_piece(self, piece):

for i, row in enumerate(piece.shape[piece.rotation % len(piece.shape)]):

for j, cell in enumerate(row):

if cell == 'O':

self.grid[piece.y + i][piece.x + j] = piece.color

lines\_cleared = self.clear\_lines()

self.score += lines\_cleared \* 100

self.current\_piece = self.new\_piece()

if not self.valid\_move(self.current\_piece, 0, 0, 0):

self.game\_over = True

return lines\_cleared

def update(self):

if not self.game\_over:

if self.valid\_move(self.current\_piece, 0, 1, 0):

self.current\_piece.y += 1

else:

self.lock\_piece(self.current\_piece)

def draw(self, screen):

for y, row in enumerate(self.grid):

for x, cell in enumerate(row):

if cell:

pygame.draw.rect(screen, cell, (x \* SQUARE\_SIZE, y \* SQUARE\_SIZE, SQUARE\_SIZE - 1, SQUARE\_SIZE - 1))

if self.current\_piece:

for i, row in enumerate(

self.current\_piece.shape[self.current\_piece.rotation % len(self.current\_piece.shape)]):

for j, cell in enumerate(row):

if cell == 'O':

pygame.draw.rect(screen, self.current\_piece.color, (

(self.current\_piece.x + j) \* SQUARE\_SIZE, (self.current\_piece.y + i) \* SQUARE\_SIZE,

SQUARE\_SIZE - 1,

SQUARE\_SIZE - 1))

def draw\_scores(screen, score, high\_score, x, y):

font = pygame.font.Font(None, 36)

score\_text = font.render(f"Score: {score}", True, WHITE)

high\_score\_text = font.render(f"High Score: {high\_score}", True, WHITE)

screen.blit(score\_text, (x, y))

screen.blit(high\_score\_text, (x, y + 40))

def draw\_game\_over(screen, x, y):

font = pygame.font.Font(None, 48)

text = font.render("Game Over", True, RED)

screen.blit(text, (x, y))

def read\_high\_score(file\_path):

try:

with open(file\_path, 'r') as file:

high\_score = file.read()

return int(high\_score) if high\_score else 0

except FileNotFoundError:

return 0

def save\_high\_score(file\_path, score):

with open(file\_path, 'w') as file:

file.write(str(score))

def run\_game(width,height):

pygame.init()

pygame.font.init()

if width and height:

WIDTH = width

HEIGHT = height

else:

WIDTH, HEIGHT = 400, 400

global GRID\_WIDTH, SQUARE\_SIZE

if WIDTH <= 400:

GRID\_WIDTH = 12

elif WIDTH <= 500:

GRID\_WIDTH = 16

else:

GRID\_WIDTH = 20

SQUARE\_SIZE = WIDTH // GRID\_WIDTH

GRID\_HEIGHT = HEIGHT // SQUARE\_SIZE

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption('Tetris')

clock = pygame.time.Clock()

game = Tetris(GRID\_WIDTH, GRID\_HEIGHT)

fall\_time = 0

fall\_speed = 80

high\_score\_file = 'high\_score.txt'

high\_score = read\_high\_score(high\_score\_file)

while True:

screen.fill(BLACK)

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

if game.valid\_move(game.current\_piece, -1, 0, 0):

game.current\_piece.x -= 1

if event.key == pygame.K\_RIGHT:

if game.valid\_move(game.current\_piece, 1, 0, 0):

game.current\_piece.x += 1

if event.key == pygame.K\_DOWN:

if game.valid\_move(game.current\_piece, 0, 1, 0):

game.current\_piece.y += 1

if event.key == pygame.K\_UP:

if game.valid\_move(game.current\_piece, 0, 0, 1):

game.current\_piece.rotation += 1

if event.key == pygame.K\_SPACE:

while game.valid\_move(game.current\_piece, 0, 1, 0):

game.current\_piece.y += 1

game.lock\_piece(game.current\_piece)

delta\_time = clock.get\_rawtime()

fall\_time += delta\_time

if fall\_time >= fall\_speed:

game.update()

fall\_time = 0

draw\_scores(screen, game.score, high\_score, 10, 10)

game.draw(screen)

if game.game\_over:

draw\_game\_over(screen, WIDTH // 2 - 100, HEIGHT // 2 - 30)

for event in pygame.event.get():

if event.type == pygame.KEYDOWN:

game = Tetris(WIDTH // SQUARE\_SIZE, HEIGHT // SQUARE\_SIZE)

game.game\_over = False

continue

if game.score > high\_score:

high\_score = game.score

save\_high\_score(high\_score\_file, high\_score)

if event.type == pygame.KEYDOWN:

game = Tetris(WIDTH // SQUARE\_SIZE, HEIGHT // SQUARE\_SIZE)

pygame.display.flip()

clock.tick(60)

2. Материали и източници

**Python**: Език за програмиране от високо ниво. Използва се за разработването на играта Тетрис, като предоставя необходимата инфраструктура и функционалности. Включени са някои модули от стандартната му библиотека като **random** и **sys**

**Pygame**: Библиотека, съдадена за **Python**, специализирана в създаването на игри и мултимедийни приложения. Използвана е за графичното представяне на играта, обработка на събитията от клавиатурата и управлението на игровия цикъл.

**PyInstaller**: Инструмент за компилиране на **Python** скриптове в изпълними .exe файлове. Използва се за създаване на стандартни Windows приложения от Python код, като обединява **Python** интерпретатора, всички необходими библиотеки и кода в един изпълним файл.

<https://www.techwithtim.net/tutorials/game-development-with-python/tetris-pygame/tutorial-1> - Връзка към ръководство и обяснения за разработката на играта Тетрис, използвайки Python и Pygame. Ресурсът е използван за насоки при създаването на играта, включително работата с компютърна графика, обработка на събитията и логиката на играта.

<https://stackoverflow.com/questions/54210392/how-can-i-convert-pygame-to-exe> - Връзка към страница от форум, с указания за компилирането на **Pygame** проект в изпълним (.exe) файл.