# 2. ОПИС ПРОЕКТНИХ РІШЕНЬ

## 2.1. Опис програмних засобів та програмного забезпечення

Python – це мова програмування, яка широко використовується в інтернет-додатках, розробці програмного забезпечення, науці даних і машинному навчанні. Розробники використовують Python, тому що він ефективний, простий у вивченні та працює на різних платформах. Програми мовою Python можна завантажити безкоштовно, вони сумісні з усіма типами систем та підвищують швидкість розробки.

Розробники можуть легко читати та розуміти програми на Python, оскільки мова має базовий синтаксис, схожий на синтаксис англійської.

Python допомагає розробникам бути більш продуктивними, оскільки вони можуть писати програми на Python, використовуючи менше рядків коду, ніж іншими мовами.

Python має велику стандартну бібліотеку, що містить багаторазові коди практично для будь-якого завдання. В результаті розробникам не потрібно писати код із нуля.

PyCharm – інтегроване середовище розробки для мови програмування Python. Надає засоби для аналізу коду, графічний інтерфейс, інструмент для запуску юніт-тестів і підтримує веброзробку на Django. PyCharm розроблена чеською компанією JetBrains на основі IntelliJ IDEA. Також він має:

* статичний аналіз коду, підсвічування синтаксису та помилок;
* навігацію серед проектів та початкового коду: відображення файлової структури проекту, швидкий перехід між файлами, класами та методами;
* рефакторинг: перейменування, витяг методу, введення змінної, введення константи, підняття та опускання методу тощо;
* інструменти для веб-розробки з використанням фреймворку Django;
* вбудований інтерпретатор для Python;
* вбудовані інструменти для юніт-тестування;
* розробку за допомогою Google App Engine;
* підтримку систем контролю версій: загальний користувацький інтерфейс для Mercurial, Git, Subversion, Perforce та CVS з підтримкою списків змін та злиття.

## 2.2. Алгоритм програми

Гра використовує бібліотеку PyGame, на основі якої було стоврено алгоритм програми та методи.

Також є особливості на які треба звернути увагу.

АЛГОРИТМ РОЗТАШУВАННЯ КУЛЬОК НА ІГРОВІЙ ДОШЦІ

Цей алгоритм виконує заповнення масиву кульок array новими кульками та викликає функцію setArrayPos для встановлення їхніх позицій на дошці гри.

Перший зовнішній цикл for проходиться по шарах кульок (BUBBLELAYERS), які розташовані у масиві.

Другий вкладений цикл for проходиться по стовпцях у кожному шарі (len(array[row])), що відповідає кількості стовпців у рядку. В середині циклу for застосовується функція random.shuffle(gameColorList), яка перемішує кольори у списку gameColorList.

Це забезпечує випадковий вибір кольору для нової кульки.Створюється нова кулька з випадково обраним кольором, рядком та стовпцем.

Це виконується за допомогою конструктора Bubble(gameColorList[0], row, column), де gameColorList[0] – перший елемент списку gameColorList, який був перемішаний.Нова кулька newBubble присвоюється відповідному елементу масиву array[row][column].

Після заповнення масиву кульками викликається функція setArrayPos(array), яка задає положення кульок на ігровому полі з урахуванням їх рядка, стовпця та зміщення.

АЛГОРИТМ ВСТАНОВЛЕННЯ КУЛЬОК НА ДОШЦІ

Цей алгоритм відповідає за встановлення позицій кульок у масиві на дошці гри. Функція setArrayPos приймає масив кульок array як параметр.

Перші два вкладених цикли for перебирають рядки та стовпці масиву та встановлюють положення кульки на дошці. Якщо елемент масиву не пустий (BLANK), то встановлюються координати x та y для прямокутника кульки (array[row][column].rect). Координата x обчислюється як (BUBBLEWIDTH \* column) + 5, що відповідає горизонтальному зсуву для кожної кульки. Координата y обчислюється як (BUBBLEWIDTH \* row) + 5, що відповідає вертикальному зсуву для кожної кульки.

Наступний вкладений цикл for проходиться по непарних рядках (починаючи з 1) і зміщує кульки на піврадіуса вправо. Це необхідно для досягнення коректного розташування кульок у рядках, які зміщені на половину кульки відносно інших рядків.

Останній вкладений цикл for проходиться по рядках (починаючи з 1) і зміщує кульки вгору на відповідну відстань BUBBLEYADJUST, що залежить від номера рядка. Це також необхідно для коректного розташування кульок у стовпцях.

Після встановлення позицій кульок, викликається функція deleteExtraBubbles, яка видаляє зайві кульки з масиву.

Загалом, функція setArrayPos гарантує, що кульки розташовані на дошці гри відповідно до їхніх позицій у масиві, з урахуванням зсувів та коригувань, які забезпечують правильну геометрію гри.

АЛГОРИТМ РОБОТИ MAIN

1. Цей алгоритмпредставляє головну функцію main, яка виконує основний цикл гри.
2. Основна частина коду має таку структуру:
3. Ініціалізуємо бібліотеку Pygame за допомогою pygame.init().
4. Встановлюємо заголовок вікна гри за допомогою pygame.display.set\_caption('Bubble shooter!').
5. Встановлюємо основний шрифт для тексту гри за допомогою MAINFONT = pygame.font.SysFont('Helvetica', TEXTHEIGHT).
6. Створюємо вікно гри та повертаємо поверхню та прямокутник вікна за допомогою DISPLAYSURF, DISPLAYRECT = makeDisplay().
7. Показуємо головне меню гри та отримуємо вибір користувача за допомогою choice = showMenu().
8. Запускаємо гру, якщо користувач обрав "New Game" (відповідно choice == 'newgame').
9. У внутрішньому циклі while True виконується основна логіка гри:

* Викликається функція getBlankBoard(), яка створює нову пусту дошку гри.
* Запускається основний цикл гри за допомогою runGame(). Цей цикл продовжується, поки гра триває.
* Після закінчення гри отримуємо рахунок та статус перемоги або поразки.
* Показуємо екран завершення гри за допомогою endScreen(score, winorlose).
* Показуємо головне меню гри та отримуємо вибір користувача (gameMode = showMenu()).
* Завершуємо гру, якщо користувач обрав "Quit" (відповідно gameMode == 'quit').
* Інакше, якщо користувач обрав "New Game" (відповідно gameMode == 'newgame'), запускаємо нову гру.
* Інакше, якщо choice == 'quit', завершуємо гру за допомогою terminate().

Цей код управляє виконанням гри, показом меню та обробкою вибору користувача, а також запуском та закриття гри.

## 2.3. Опис специфікації функцій

Більш детальніше про функція які були наведенні вище ви зможете дізнатися з таблиці 2.1 і 2.2.

Таблиця 2.1 – Опис специфікації розроблених функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | runGame | Запуск гри |  | Початкова позиція об’єктів | PyGame |
| 2 | deleteExtraBubbles | Видаляє кульки, які виходять за межі вікна | Початкові позиції кульок | Кінцева встановлена позиція об’єктів | PyGame |
| 3 | setArrayPos | Встановлює позиції кульок на дошці | Початкові позиції кульок | Кінцева встановлена позиція об’єктів | PyGame |

Таблиця 2.2 – Опис стандартних функцій

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Заголовний файл |
| 1 | create | Створення об’єктів | PyGame |
| 2 | update | Оновлення створених методів гри | PyGame |
| 3 | deleteList | Функція для рекурсивного видалення кульки | Python |
| 4 | draw | Відображення об’єкта | PyGane |
| 5 | event.key | Перевірка на натиснуту кнопку | libGDX |
| 6 | if | Перевірка умови, повертає true або false | Python |
| 7 | for | Задання циклу | Python |
| 8 | while | Задання циклу з умовою умова | Python |
| 9 | image | Встановлення об’єкту типу картинка | PyGame |
| 10 | direction | Встановлює дію для натиснутої клавіші | PyGame |
| 11 | key | Список допустимих клавіш | PyGame |
| 12 | import | Імпортування бібліотек | Python |

## Тестування

Тестування гри проходило за такими пунктами:

* сумісність з різними ос;
* наявність системних помилок;
* стабільність роботи програми;
* тестування ігрового дизайну;
* тестування ігрового процесу;
* коректне відображення гри.

Під час проведених тестувань, гра показала себе найкращим чином, помилок в роботі не виявлено.

# 

# 3. ІНСТРУКЦІЯ КОРИСТУВАЧА

Програма призначена для розважальних цілей, змаганню з друзями та гарного проведення часу.

Основною вимогою до програмного забезпечення є наявність Python версії 3.10 на комп’ютері.

Мінімальні вимоги до комп’ютера:

* Процесор. 1 гігагерц (ГГц)\* або вище з підтримкою PAE, NX та SSE2
* Оперативна пам’ять: 100 Мб,
* Windows 8 / 8.1 / 10 / 11,
* Пам’ять на жорсткому диску 80 Mб
* Відеоадаптер з 16Мб відеопам’яті.

З після запуску програми перед користувачем з’являється меню, де потрібно натискаючи кнопки N, щоб почати гру, чи Q, щоб вийти з неї (Рисунок 3.1).

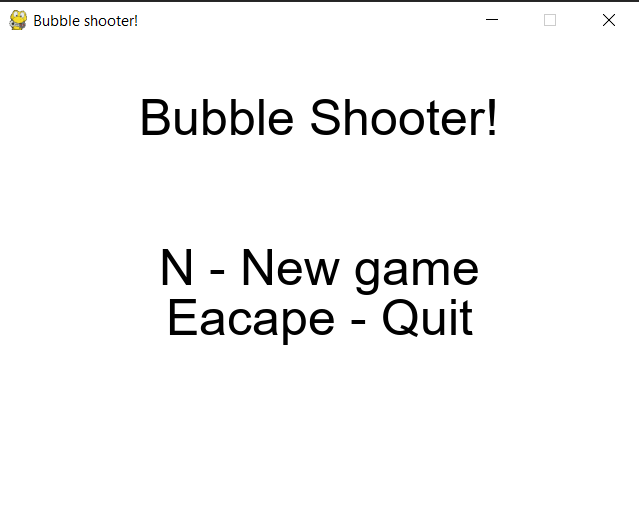


Рисунок 3.1 – Меню гри

Після початку гри відкривається ігрове поле, на якому розміщені кульки, стрілка-вказівник, показчик очок, та показчик наступного кольору кульки, яка буде вистріляна наступною (Рисунок 3.2).

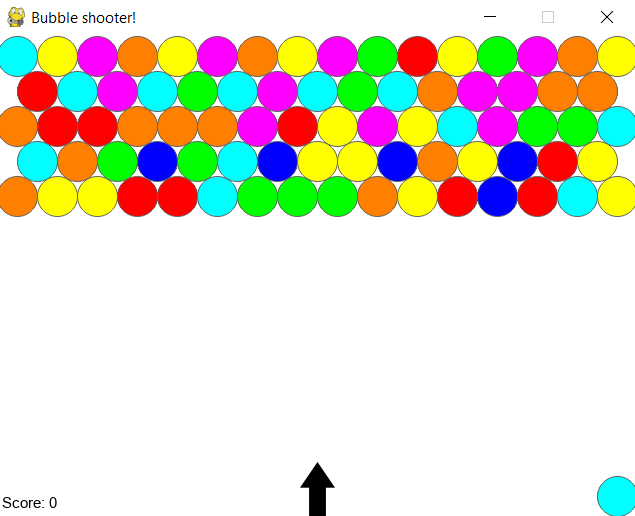


Рисунок 3.2 – Ігрове поле

Користувач може керувати наведенням кульки за допомогою клавіш зі стрілками ліворуч та праворуч відповідно, щоб вистрілити кулькою потрібно натиснути лавішу «Пробіл»

Користувач може програти, якщо вистріляні ним кульки дістануть до нижньої лінії вікна, після чого з’явиться вікно програшу, де буде показано набрану кількість очок, також користувачу буде запропоновано почати знову, або вийти з гри (Рисунок 3.3).

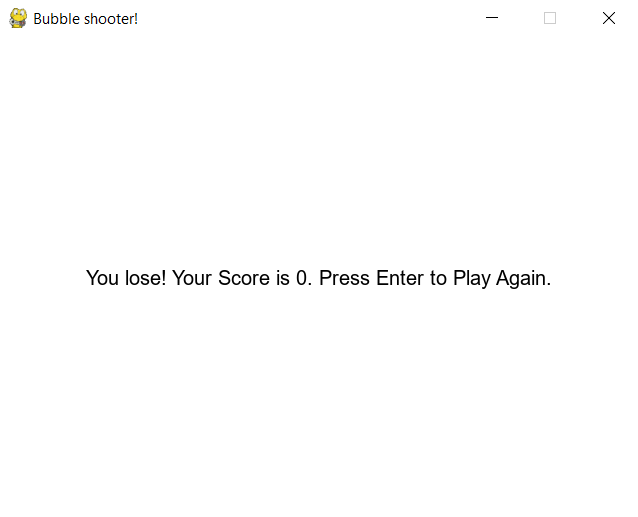


Рисунок 3.3 – Вікно програшу

Після того, як користувач знищить всі кульки на ігровій дошці, йому зобразиться аналогічне як і до програшу вікно з зазначенням того, що користувач вийграв з кількістю очок, та буде запропоновано зіграти ще.

# ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

import copy

import math

import random

import sys

import pygame

import pygame.gfxdraw

from pygame.locals import \*

# Загальні константи

FPS = 120

WINDOWWIDTH = 640

WINDOWHEIGHT = 480

TEXTHEIGHT = 20

BUBBLERADIUS = 20

BUBBLEWIDTH = BUBBLERADIUS \* 2

BUBBLELAYERS = 5

BUBBLEYADJUST = 5

STARTX = WINDOWWIDTH / 2

STARTY = WINDOWHEIGHT - 27

ARRAYWIDTH = 26

ARRAYHEIGHT = 24

# Константи меню

MENUTEXTHEIGHT = 50

MENUOPTION1 = 'N - New game'

MENUOPTION2 = 'Eacape - Quit'

RIGHT = 'right'

LEFT = 'left'

BLANK = '.'

# Кольори

GRAY = (100, 100, 100)

NAVYBLUE = (60, 60, 100)

WHITE = (255, 255, 255)

RED = (255, 0, 0)

GREEN = (0, 255, 0)

BLUE = (0, 0, 255)

YELLOW = (255, 255, 0)

ORANGE = (255, 128, 0)

PURPLE = (255, 0, 255)

CYAN = (0, 255, 255)

BLACK = (0, 0, 0)

COMBLUE = (233, 232, 255)

BGCOLOR = WHITE

COLORLIST = [RED, GREEN, BLUE, YELLOW, ORANGE, PURPLE, CYAN]

class Bubble(pygame.sprite.Sprite):

def \_\_init\_\_(self, color, row=0, column=0):

# Ініціалізація об'єкта кульки

pygame.sprite.Sprite.\_\_init\_\_(self)

self.rect = pygame.Rect(0, 0, 30, 30)

self.rect.centerx = STARTX

self.rect.centery = STARTY

self.speed = 10

self.color = color

self.radius = BUBBLERADIUS

self.angle = 0

self.row = row

self.column = column

def update(self):

# Оновлення позиції кульки на кожній ітерації гри

global x, y

if self.angle == 90:

x = 0

y = self.speed \* -1

elif self.angle < 90:

x = self.xcalculate(self.angle)

y = self.ycalculate(self.angle)

elif self.angle > 90:

x = self.xcalculate(180 - self.angle) \* -1

y = self.ycalculate(180 - self.angle)

self.rect.x += x

self.rect.y += y

def draw(self):

# Малювання кульки на екрані

pygame.gfxdraw.filled\_circle(DISPLAYSURF, self.rect.centerx, self.rect.centery, self.radius, self.color)

pygame.gfxdraw.aacircle(DISPLAYSURF, self.rect.centerx, self.rect.centery, self.radius, GRAY)

def xcalculate(self, angle):

# Обчислення руху по осі X залежно від кута

radians = math.radians(angle)

xmove = math.cos(radians) \* self.speed

return xmove

def ycalculate(self, angle):

# Обчислення руху по осі Y залежно від кута

radians = math.radians(angle)

ymove = math.sin(radians) \* self.speed \* -1

return ymove

class Arrow(pygame.sprite.Sprite):

def \_\_init\_\_(self):

# Ініціалізація об'єкта стрілки

pygame.sprite.Sprite.\_\_init\_\_(self)

self.angle = 90

arrowImage = pygame.image.load('Arrow.png')

arrowImage.convert\_alpha()

arrowRect = arrowImage.get\_rect()

self.image = arrowImage

self.transformImage = self.image

self.rect = arrowRect

self.rect.centerx = STARTX

self.rect.centery = STARTY

def update(self, direction):

# Оновлення кута стрілки відповідно до напрямку руху

if direction == LEFT and self.angle < 170:

self.angle += 2

elif direction == RIGHT and self.angle > 10:

self.angle -= 2

self.transformImage = pygame.transform.rotate(self.image, self.angle)

self.rect = self.transformImage.get\_rect()

self.rect.centerx = STARTX

self.rect.centery = STARTY

def draw(self):

# Малювання стрілки на екрані

DISPLAYSURF.blit(self.transformImage, self.rect)

class Score(object):

def \_\_init\_\_(self):

# Ініціалізація об'єкта для рахунку очків

self.total = 0

self.font = pygame.font.SysFont('Helvetica', 15)

self.render = self.font.render('Score: ' + str(self.total), True, BLACK, WHITE)

self.rect = self.render.get\_rect()

self.rect.left = 5

self.rect.bottom = WINDOWHEIGHT - 5

def update(self, deleteList):

# Оновлення рахунку очків

self.total += ((len(deleteList)) \* 10)

self.render = self.font.render('Score: ' + str(self.total), True, BLACK, WHITE)

def draw(self):

# Малювання рахунку очків на екрані

DISPLAYSURF.blit(self.render, self.rect)

def getBlankBoard():

# Створення пустої дошки

board: list[list[None]] = []

for X in range(WINDOWWIDTH):

board.append([None] \* WINDOWHEIGHT)

return board

def showMenu():

# Відображення головного меню гри

global DISPLAYSURF

font = pygame.font.SysFont('Helvetica', MENUTEXTHEIGHT)

titleText = font.render('Bubble Shooter!', True, BLACK)

titleRect = titleText.get\_rect()

titleRect.centerx = DISPLAYSURF.get\_rect().centerx

titleRect.top = 50

option1Text = font.render(MENUOPTION1, True, BLACK)

option1Rect = option1Text.get\_rect()

option1Rect.centerx = DISPLAYSURF.get\_rect().centerx

option1Rect.top = 200

option2Text = font.render(MENUOPTION2, True, BLACK)

option2Rect = option2Text.get\_rect()

option2Rect.centerx = DISPLAYSURF.get\_rect().centerx

option2Rect.top = 250

while True:

DISPLAYSURF.fill(WHITE)

DISPLAYSURF.blit(titleText, titleRect)

DISPLAYSURF.blit(option1Text, option1Rect)

DISPLAYSURF.blit(option2Text, option2Rect)

pygame.display.update()

for event in pygame.event.get():

if event.type == QUIT:

terminate()

elif event.type == KEYDOWN:

if event.key == K\_n:

return 'newgame'

elif event.key == K\_ESCAPE:

return 'quit'

def makeBlankBoard():

# Створює пусту дошку з кульками

array = []

for row in range(ARRAYHEIGHT):

column = []

for i in range(ARRAYWIDTH):

column.append(BLANK)

array.append(column)

return array

def setBubbles(array, gameColorList):

# Розташовує кульки на дошці з кольорами зі списку gameColorList

for row in range(BUBBLELAYERS):

for column in range(len(array[row])):

random.shuffle(gameColorList)

newBubble = Bubble(gameColorList[0], row, column)

array[row][column] = newBubble

setArrayPos(array)

def setArrayPos(array):

# Встановлює позиції кульки на дошці

for row in range(ARRAYHEIGHT):

for column in range(len(array[row])):

if array[row][column] != BLANK:

array[row][column].rect.x = (BUBBLEWIDTH \* column) + 5

array[row][column].rect.y = (BUBBLEWIDTH \* row) + 5

for row in range(1, ARRAYHEIGHT, 2):

for column in range(len(array[row])):

if array[row][column] != BLANK:

array[row][column].rect.x += BUBBLERADIUS

for row in range(1, ARRAYHEIGHT):

for column in range(len(array[row])):

if array[row][column] != BLANK:

array[row][column].rect.y -= (BUBBLEYADJUST \* row)

deleteExtraBubbles(array)

def deleteExtraBubbles(array):

# Видаляє кульки, які виходять за межі вікна

for row in range(ARRAYHEIGHT):

for column in range(len(array[row])):

if array[row][column] != BLANK:

if array[row][column].rect.right > WINDOWWIDTH:

array[row][column] = BLANK

def updateColorList(bubbleArray):

# Оновлює список кольорів на основі кульок на дошці

newColorList = []

for row in range(len(bubbleArray)):

for column in range(len(bubbleArray[0])):

if bubbleArray[row][column] != BLANK:

newColorList.append(bubbleArray[row][column].color)

colorSet = set(newColorList)

if len(colorSet) < 1:

colorList = [WHITE]

return colorList

else:

return list(colorSet)

def checkForFloaters(bubbleArray):

# Перевіряє, чи є кульки, які "висять" у повітрі (не мають кульок-сусідів)

bubbleList = [column for column in range(len(bubbleArray[0]))

if bubbleArray[0][column] != BLANK]

newBubbleList = []

for i in range(len(bubbleList)):

if i == 0:

newBubbleList.append(bubbleList[i])

elif bubbleList[i] > bubbleList[i - 1] + 1:

newBubbleList.append(bubbleList[i])

copyOfBoard = copy.deepcopy(bubbleArray)

for row in range(len(bubbleArray)):

for column in range(len(bubbleArray[0])):

bubbleArray[row][column] = BLANK

for column in newBubbleList:

popFloaters(bubbleArray, copyOfBoard, column)

def popFloaters(bubbleArray, copyOfBoard, column, row=0):

# Рекурсивно видаляє "висячі" кульки

if (row < 0 or row > (len(bubbleArray) - 1)

or column < 0 or column > (len(bubbleArray[0]) - 1)):

return

elif copyOfBoard[row][column] == BLANK:

return

elif bubbleArray[row][column] == copyOfBoard[row][column]:

return

bubbleArray[row][column] = copyOfBoard[row][column]

if row == 0:

popFloaters(bubbleArray, copyOfBoard, column + 1, row)

popFloaters(bubbleArray, copyOfBoard, column - 1, row)

popFloaters(bubbleArray, copyOfBoard, column, row + 1)

popFloaters(bubbleArray, copyOfBoard, column - 1, row + 1)

elif row % 2 == 0:

popFloaters(bubbleArray, copyOfBoard, column + 1, row)

popFloaters(bubbleArray, copyOfBoard, column - 1, row)

popFloaters(bubbleArray, copyOfBoard, column, row + 1)

popFloaters(bubbleArray, copyOfBoard, column - 1, row + 1)

popFloaters(bubbleArray, copyOfBoard, column, row - 1)

popFloaters(bubbleArray, copyOfBoard, column - 1, row - 1)

else:

popFloaters(bubbleArray, copyOfBoard, column + 1, row)

popFloaters(bubbleArray, copyOfBoard, column - 1, row)

popFloaters(bubbleArray, copyOfBoard, column, row + 1)

popFloaters(bubbleArray, copyOfBoard, column + 1, row + 1)

popFloaters(bubbleArray, copyOfBoard, column, row - 1)

popFloaters(bubbleArray, copyOfBoard, column + 1, row - 1)

def stopBubble(bubbleArray, newBubble, launchBubble, score):

# Зупиняє кульку, перевіряє зіткнення з іншими кульками та видаляє групи кульок

global newRow, newColumn

deleteList = []

for row in range(len(bubbleArray)):

for column in range(len(bubbleArray[row])):

if bubbleArray[row][column] != BLANK and newBubble is not None:

if (pygame.sprite.collide\_rect(newBubble, bubbleArray[row][column])) or newBubble.rect.top < 0:

if newBubble.rect.top < 0:

newRow, newColumn = addBubbleToTop(bubbleArray, newBubble)

elif newBubble.rect.centery >= bubbleArray[row][column].rect.centery:

if newBubble.rect.centerx >= bubbleArray[row][column].rect.centerx:

if row == 0 or row % 2 == 0:

newRow = row + 1

newColumn = column

if bubbleArray[newRow][newColumn] != BLANK:

newRow = newRow - 1

bubbleArray[newRow][newColumn] = copy.copy(newBubble)

bubbleArray[newRow][newColumn].row = newRow

bubbleArray[newRow][newColumn].column = newColumn

else:

newRow = row + 1

newColumn = column + 1

if bubbleArray[newRow][newColumn] != BLANK:

newRow = newRow - 1

bubbleArray[newRow][newColumn] = copy.copy(newBubble)

bubbleArray[newRow][newColumn].row = newRow

bubbleArray[newRow][newColumn].column = newColumn

elif newBubble.rect.centerx < bubbleArray[row][column].rect.centerx:

if row == 0 or row % 2 == 0:

newRow = row + 1

newColumn = column - 1

if newColumn < 0:

newColumn = 0

if bubbleArray[newRow][newColumn] != BLANK:

newRow = newRow - 1

bubbleArray[newRow][newColumn] = copy.copy(newBubble)

bubbleArray[newRow][newColumn].row = newRow

bubbleArray[newRow][newColumn].column = newColumn

else:

newRow = row + 1

newColumn = column

if bubbleArray[newRow][newColumn] != BLANK:

newRow = newRow - 1

bubbleArray[newRow][newColumn] = copy.copy(newBubble)

bubbleArray[newRow][newColumn].row = newRow

bubbleArray[newRow][newColumn].column = newColumn

elif newBubble.rect.centery < bubbleArray[row][column].rect.centery:

if newBubble.rect.centerx >= bubbleArray[row][column].rect.centerx:

if row == 0 or row % 2 == 0:

newRow = row - 1

newColumn = column

if bubbleArray[newRow][newColumn] != BLANK:

newRow = newRow + 1

bubbleArray[newRow][newColumn] = copy.copy(newBubble)

bubbleArray[newRow][newColumn].row = newRow

bubbleArray[newRow][newColumn].column = newColumn

else:

newRow = row - 1

newColumn = column + 1

if bubbleArray[newRow][newColumn] != BLANK:

newRow = newRow + 1

bubbleArray[newRow][newColumn] = copy.copy(newBubble)

bubbleArray[newRow][newColumn].row = newRow

bubbleArray[newRow][newColumn].column = newColumn

elif newBubble.rect.centerx <= bubbleArray[row][column].rect.centerx:

if row == 0 or row % 2 == 0:

newRow = row - 1

newColumn = column - 1

if bubbleArray[newRow][newColumn] != BLANK:

newRow = newRow + 1

bubbleArray[newRow][newColumn] = copy.copy(newBubble)

bubbleArray[newRow][newColumn].row = newRow

bubbleArray[newRow][newColumn].column = newColumn

else:

newRow = row - 1

newColumn = column

if bubbleArray[newRow][newColumn] != BLANK:

newRow = newRow + 1

bubbleArray[newRow][newColumn] = copy.copy(newBubble)

bubbleArray[newRow][newColumn].row = newRow

bubbleArray[newRow][newColumn].column = newColumn

popBubbles(bubbleArray, newRow, newColumn, newBubble.color, deleteList)

if len(deleteList) >= 3:

for pos in deleteList:

row = pos[0]

column = pos[1]

bubbleArray[row][column] = BLANK

checkForFloaters(bubbleArray)

score.update(deleteList)

launchBubble = False

newBubble = None

return launchBubble, newBubble, score

def addBubbleToTop(bubbleArray, bubble):

# Додає кульку на верхню позицію в масиві кульок

posx = bubble.rect.centerx

leftSidex = posx - BUBBLERADIUS

columnDivision = math.modf(float(leftSidex) / float(BUBBLEWIDTH))

column = int(columnDivision[1])

if columnDivision[0] < 0.5:

bubbleArray[0][column] = copy.copy(bubble)

else:

column += 1

bubbleArray[0][column] = copy.copy(bubble)

row = 0

return row, column

def popBubbles(bubbleArray, row, column, color, deleteList):

# Рекурсивно видаляє групи кульок одного кольору

if row < 0 or column < 0 or row > (len(bubbleArray) - 1) or column > (len(bubbleArray[0]) - 1):

return

elif bubbleArray[row][column] == BLANK:

return

elif bubbleArray[row][column].color != color:

return

for bubble in deleteList:

if bubbleArray[bubble[0]][bubble[1]] == bubbleArray[row][column]:

return

deleteList.append((row, column))

if row == 0:

popBubbles(bubbleArray, row, column - 1, color, deleteList)

popBubbles(bubbleArray, row, column + 1, color, deleteList)

popBubbles(bubbleArray, row + 1, column, color, deleteList)

popBubbles(bubbleArray, row + 1, column - 1, color, deleteList)

elif row % 2 == 0:

popBubbles(bubbleArray, row + 1, column, color, deleteList)

popBubbles(bubbleArray, row + 1, column - 1, color, deleteList)

popBubbles(bubbleArray, row - 1, column, color, deleteList)

popBubbles(bubbleArray, row - 1, column - 1, color, deleteList)

popBubbles(bubbleArray, row, column + 1, color, deleteList)

popBubbles(bubbleArray, row, column - 1, color, deleteList)

else:

popBubbles(bubbleArray, row - 1, column, color, deleteList)

popBubbles(bubbleArray, row - 1, column + 1, color, deleteList)

popBubbles(bubbleArray, row + 1, column, color, deleteList)

popBubbles(bubbleArray, row + 1, column + 1, color, deleteList)

popBubbles(bubbleArray, row, column + 1, color, deleteList)

popBubbles(bubbleArray, row, column - 1, color, deleteList)

def drawBubbleArray(array):

# Малює всі кульки в масиві

for row in range(ARRAYHEIGHT):

for column in range(len(array[row])):

if array[row][column] != BLANK:

array[row][column].draw()

def makeDisplay():

# Створює вікно для відображення гри

displaysurf = pygame.display.set\_mode((WINDOWWIDTH, WINDOWHEIGHT))

displayrect = displaysurf.get\_rect()

displaysurf.fill(BGCOLOR)

displaysurf.convert()

pygame.display.update()

return displaysurf, displayrect

def terminate():

# Завершує програму і виходить з неї

pygame.quit()

sys.exit()

def coverNextBubble():

# Показує наступну кульку, якою гравець стрілятиме

whiteRect = pygame.Rect(0, 0, BUBBLEWIDTH, BUBBLEWIDTH)

whiteRect.bottom = WINDOWHEIGHT

whiteRect.right = WINDOWWIDTH

pygame.draw.rect(DISPLAYSURF, BGCOLOR, whiteRect)

def endScreen(score, winorlose):

# Відображає екран завершення гри з результатом

endFont = pygame.font.SysFont('Helvetica', 20)

endMessage1 = endFont.render('You ' + winorlose + '! Your Score is ' + str(score) + '. Press Enter to Play Again.',

True, BLACK, BGCOLOR)

endMessage1Rect = endMessage1.get\_rect()

endMessage1Rect.center = DISPLAYRECT.center

DISPLAYSURF.fill(BGCOLOR)

DISPLAYSURF.blit(endMessage1, endMessage1Rect)

pygame.display.update()

while True:

for event in pygame.event.get():

if event.type == QUIT:

terminate()

elif event.type == KEYUP:

if event.key == K\_RETURN:

return

elif event.key == K\_ESCAPE:

terminate()

def runGame():

# Ініціалізація змінних для гри

gameColorList = copy.deepcopy(COLORLIST)

direction = None

launchBubble = False

newBubble = None

arrow = Arrow()

bubbleArray = makeBlankBoard()

setBubbles(bubbleArray, gameColorList)

nextBubble = Bubble(gameColorList[0])

nextBubble.rect.right = WINDOWWIDTH - 5

nextBubble.rect.bottom = WINDOWHEIGHT - 5

score = Score()

while True:

DISPLAYSURF.fill(BGCOLOR)

for event in pygame.event.get():

if event.type == QUIT:

terminate()

elif event.type == KEYDOWN:

if event.key == K\_LEFT:

direction = LEFT

elif event.key == K\_RIGHT:

direction = RIGHT

elif event.type == KEYUP:

direction = None

if event.key == K\_SPACE:

launchBubble = True

elif event.key == K\_ESCAPE:

terminate()

if launchBubble:

if newBubble is None:

newBubble = Bubble(nextBubble.color)

newBubble.angle = arrow.angle

newBubble.update()

newBubble.draw()

if newBubble.rect.right >= WINDOWWIDTH - 5:

newBubble.angle = 180 - newBubble.angle

elif newBubble.rect.left <= 5:

newBubble.angle = 180 - newBubble.angle

launchBubble, newBubble, score = stopBubble(bubbleArray, newBubble, launchBubble, score)

finalBubbleList = []

for row in range(len(bubbleArray)):

for column in range(len(bubbleArray[0])):

if bubbleArray[row][column] != BLANK:

finalBubbleList.append(bubbleArray[row][column])

if bubbleArray[row][column].rect.bottom > (WINDOWHEIGHT - arrow.rect.height - 10):

return score.total, 'lose'

if len(finalBubbleList) < 1:

return score.total, 'win'

gameColorList = updateColorList(bubbleArray)

random.shuffle(gameColorList)

if not launchBubble:

nextBubble = Bubble(gameColorList[0])

nextBubble.rect.right = WINDOWWIDTH - 5

nextBubble.rect.bottom = WINDOWHEIGHT - 5

nextBubble.draw()

if launchBubble:

coverNextBubble()

arrow.update(direction)

arrow.draw()

setArrayPos(bubbleArray)

drawBubbleArray(bubbleArray)

score.draw()

pygame.display.update()

FPSCLOCK.tick(FPS)

def main():

global FPSCLOCK, DISPLAYSURF, DISPLAYRECT, MAINFONT

pygame.init()

FPSCLOCK = pygame.time.Clock()

pygame.display.set\_caption('Bubble shooter!')

MAINFONT = pygame.font.SysFont('Helvetica', TEXTHEIGHT)

DISPLAYSURF, DISPLAYRECT = makeDisplay()

choice = showMenu()

# Запускаємо гру, якщо користувач вибирає "New Game"

while choice == 'newgame':

getBlankBoard()

while True:

score, winorlose = runGame()

endScreen(score, winorlose)

gameMode = showMenu()

if gameMode == 'quit':

terminate()

elif gameMode == 'newgame':

runGame()

elif choice == 'quit':

terminate()

if \_\_name\_\_ == '\_\_main\_\_':

main()