Пояснення коду гри Breakout на Pygame

Імпорт бібліотек та ініціалізація

```python
from random import randint
from pygame import *
init()
```

- Імпортуємо функцію `randint` з модуля `random` для генерації випадкових чисел
- Імпортуємо всі класи та функції з бібліотеки `pygame`
- Ініціалізуємо pygame за допомогою функції `init()`

Налаштування вікна та годинника

```python
window = display.set_mode((1000, 800))
clock = time.Clock()
```

- Створюємо вікно розміром 1000x800 пікселів
- Створюємо об'єкт годинника для контролю частоти кадрів (FPS)

Створення платформи

```python
platform = Rect(400, 700, 120, 20)
platform_speed = 10
```

- Створюємо прямокутник для платформи з координатами (400, 700), шириною 120 і висотою 20 пікселів
- Встановлюємо швидкість руху платформи на 10 пікселів за кадр

Клас м'яча (Ball)

python

```python
class Ball:
    def __init__(self, x, y, radius, speed, color):
        self.x = x
        self.y = y
        self.dx = speed
        self.dy = speed
        self.radius = radius
        self.color = color
        self.rect = Rect(self.x, self.y, self.radius * 2, self.radius * 2)
```

- Конструктор класу ініціалізує м'яч з його координатами, швидкістю, радіусом та кольором
- Створюється прямокутник (`self.rect`) для виявлення зіткнень з іншими об'єктами

python

```python
    def reset(self):
        draw.circle(window, self.color, (self.x, self.y), self.radius)
        #draw.rect(window, (255, 255, 0), self.rect, 2)
```

- Метод `reset()` малює м'яч на екрані
- Закоментований рядок дозволяв би відображати прямокутник зіткнень для налагодження

python

```python
    def update(self):
        self.x += self.dx
        self.y += self.dy
        self.rect.x = self.x - self.radius
        self.rect.y = self.y - self.radius

        if self.x >= 1000 - self.radius or self.x <= 0:
            self.dx *= - 1
            self.dx += randint(1, 2)

        if self.y <= 0:
            self.dy *= -1
            self.dy += randint(1, 2)
```

- Метод `update()` оновлює позицію м'яча відповідно до його швидкості
- Оновлює позицію прямокутника зіткнень
- Обробляє відбивання м'яча від лівої, правої та верхньої стін
- При відбиванні трохи збільшує швидкість на випадкове значення (1 або 2)

Функція завантаження рівня

python

```python
def load_level_map(filename):
    bricks = list()
    with open(filename, 'r') as file:
        lines = [line for line in file.readlines()]

    for row_index, line in enumerate(lines):
        for col_index, char in enumerate(line):
            if char == '#':
                x = col_index * 50
                y = row_index * 50
                brick = Rect(x, y, 50, 50)
                bricks.append(brick)

    return bricks
```

- Функція читає текстовий файл, де символ '#' позначає цеглинку
- Перетворює кожен символ '#' на прямокутник розміром 50x50 пікселів
- Повертає список усіх цеглинок рівня

Клас бонусу (Boost)

python

```python
class Boost:
    def __init__(self, x, y, c):
        self.rect = Rect(x, y, 50, 50)
        self.color = c

    def reset(self):
        self.rect.y += 3
        draw.rect(window, self.color, self.rect)
```

- Конструктор створює прямокутник для бонусу з заданими координатами та кольором
- Метод `reset()` малює бонус та зміщує його вниз на 3 пікселі (бонус падає вниз)

Ініціалізація ігрових об'єктів

python

```
balls = list()
balls.append(Ball(200, 200, 10, 8, (255, 255, 255)))
lvl = load_level_map('lvl1.txt')
boosts = list()
```

- Створюємо список для зберігання м'ячів
- Додаємо перший м'яч з координатами (200, 200), радіусом 10, швидкістю 8 та білим кольором
- Завантажуємо рівень з файлу 'lvl1.txt'
- Створюємо пустий список для бонусів

Головний цикл гри

python

```
while True:
    for e in event.get():
        if e.type == QUIT:
            quit()
```

- Цикл продовжується нескінченно (до закриття гри)
- Перевіряємо подію закриття вікна і виходимо з гри, якщо вона відбулася

python

```
    window.fill((0, 0, 0))
    draw.rect(window, (0, 255, 255), platform, border_radius=15)
```

- Заповнюємо вікно чорним кольором
- Малюємо бірюзову платформу з заокругленими краями (радіус 15)

python

```
    for brick in lvl:
        draw.rect(window, (255, 0, 0), brick)
        draw.rect(window, (0, 0, 0), [brick.x, brick.y, brick.w, brick.h], 2)
```

- Малюємо всі цеглинки червоним кольором
- Додаємо чорну рамку шириною 2 пікселі до кожної цеглинки

python

```
    for boost in boosts:
        boost.reset()
        if boost.rect.colliderect(platform):
            balls.append(Ball(boost.rect.x, boost.rect.y, 10, 8, (255, 255, 255)))
            boosts.remove(boost)
```

- Оновлюємо та відображаємо всі бонуси
- Перевіряємо зіткнення бонусів з платформою
- Якщо бонус зібраний, додаємо новий м'яч та видаляємо бонус

python

```
    for ball in balls:
        ball.reset()
        ball.update()
        if ball.rect.colliderect(platform):
            ball.dy *= -1
```

- Відображаємо та оновлюємо всі м'ячі
- Перевіряємо зіткнення м'яча з платформою і змінюємо напрямок руху (відбиваємо вгору)

python

```
        colliding_indexes = ball.rect.collidelistall(lvl)
        if colliding_indexes:
            ball.dy *= -1
            if not randint(0, 10):
                boosts.append(Boost(ball.rect.x, ball.rect.y, (0, 255, 0)))
            for i in colliding_indexes:
                lvl.pop(i)
```

- Перевіряємо зіткнення м'яча з цеглинками
- При зіткненні відбиваємо м'яч
- З ймовірністю 1/11 створюємо зелений бонус на місці зруйнованої цеглинки
- Видаляємо всі цеглинки, з якими зіткнувся м'яч

python

```
        if ball.y >= window.get_height():
            balls.remove(ball)
```

- Якщо м'яч виходить за нижню межу екрану, видаляємо його

python

```
keys = key.get_pressed()
if keys[K_d]:
    platform.x += platform_speed
if keys[K_a]:
    platform.x -= platform_speed
```

- Отримуємо стан клавіш
- Якщо натиснута клавіша D, рухаємо платформу вправо
- Якщо натиснута клавіша A, рухаємо платформу вліво

python

```
if not balls:
    balls.append(Ball(400, 200, 10, 8, (255, 255, 255)))
```

- Якщо всі м'ячі втрачені, створюємо новий м'яч

python

```
display.update()
clock.tick(60)
```

- Оновлюємо екран
- Обмежуємо частоту кадрів до 60 FPS