Sprawozdanie z realizacji projektu ZPK – Gra Pong

---

1. Informacje ogólne

- **Temat projektu**: Gra Pong – wersja komponentowa i niekomponentowa
- **Autorzy**: Serhii Yakubiv
- **Technologie**: C# (.NET), Windows Forms
- **Środowisko pracy**: Visual Studio 2022

---

2. Cel projektu

Celem projektu było stworzenie gry typu retro "Pong" w dwóch wersjach:

- **Wersja komponentowa**: oparta na zasadach programowania komponentowego, z wydzielonymi modułami odpowiadającymi za logikę piłki, paletek, systemu punktacji i konfiguracji.
- **Wersja niekomponentowa**: maksymalnie uproszczona, zrealizowana w jednym pliku C# bez podziału na komponenty.

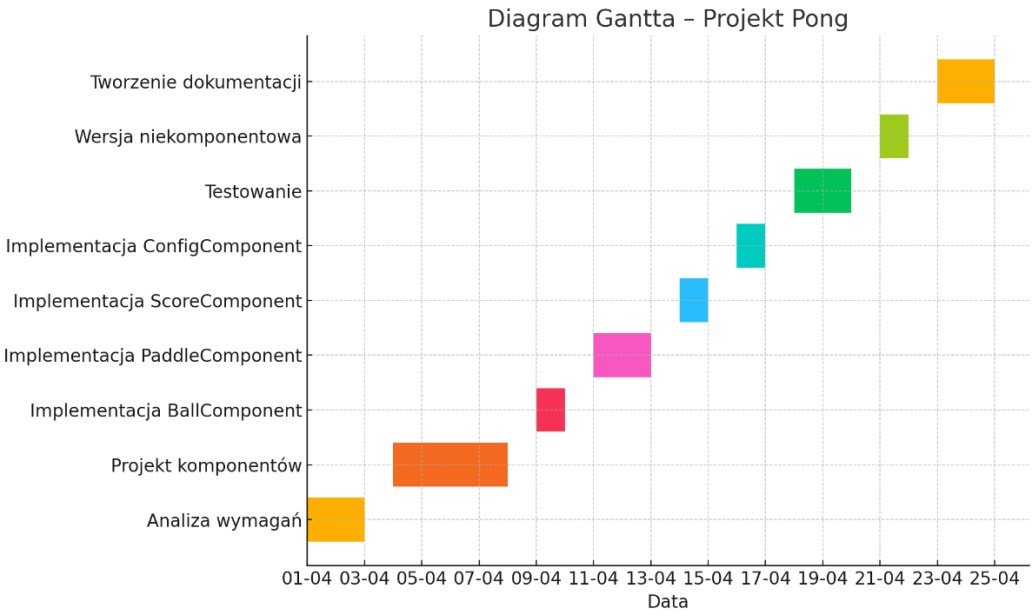
Gra oferuje tryby:

- Gracz vs Gracz
- Gracz vs Komputer (AI)

Celem rozgrywki jest zdobycie ustalonej liczby punktów szybciej niż przeciwnik.

---

3. Diagram Gantta



---

4. Wymagania funkcjonalne

Wymaganie	Opis
Ruch piłki	Piłka porusza się z ustaloną prędkością, odbija się od krawędzi i paletek
Ruch paletek	Paletki sterowane przez gracza lub AI
Kolizje	Obsługa odbić piłki i punktów
System punktacji	Zliczanie punktów, zakończenie gry po osiągnięciu limitu
Konfiguracja	Możliwość ustawienia nazw graczy, trybu gry, prędkości piłki

---

5. Wyodrębnione komponenty

Komponent	Opis
BallComponent	Logika ruchu piłki i odbić
PaddleComponent	Logika paletek: sterowanie, AI
ScoreComponent	System punktacji i zapisu najlepszego wyniku
ConfigComponent	Konfiguracja gry, serializacja do pliku

---

6. Charakterystyka komponentów

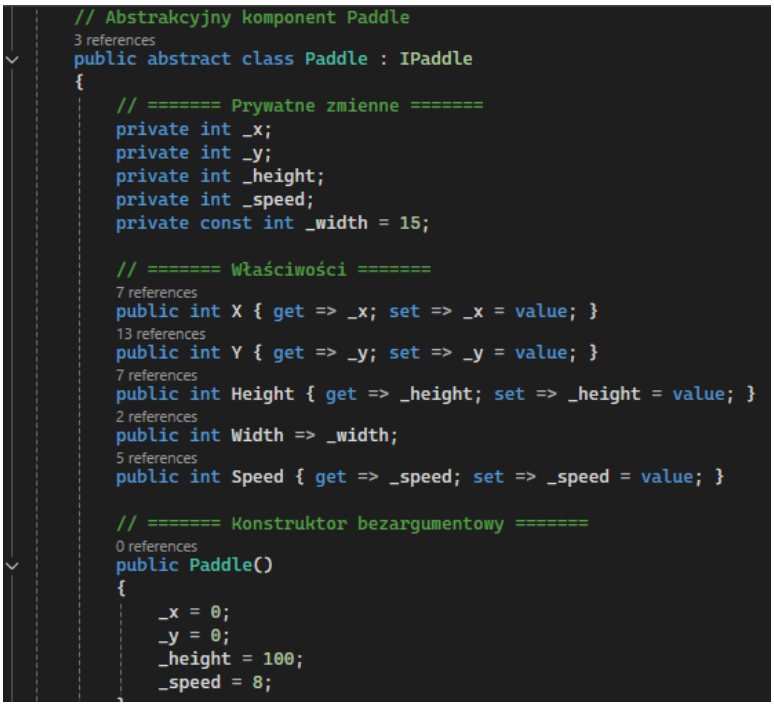
BallComponent

- **Zmiennych prywatnych**: 6
- **Default values**: rozmiar, prędkość, deakceleracja
- **Konstruktor bezargumentowy**: tak
- **Getter/Setter**: tak
- **Serializacja**: tak (JSON)
- **Testy**: BallTests.cs
- **Wywołanie**: ball.Move();, ball.Reset(800, 600);

```
// Komponent: Ball
// Odpowiada za logikę piłki w grze
10 references
public class Ball : IBall
{
    // ======= Prywatne zmienne =======
    private int _x;
    private int _y;
    private int _size;
    private float _speedX;
    private float _speedY;
    private float _deceleration;

    // ======= Wartości domyślne =======
    private const int DefaultSize = 20;
    private const float DefaultSpeed = 8f;
    private const float DefaultDeceleration = 0.5f;

    // ======= Właściwości (gettery/settery) =======
    7 references
    public int X { get => _x; set => _x = value; }
    11 references
    public int Y { get => _y; set => _y = value; }
    12 references
    public int Size { get => _size; set => _size = value; }
    7 references
    public float SpeedX { get => _speedX; set => _speedX = value; }
    5 references
    public float SpeedY { get => _speedY; set => _speedY = value; }
    2 references
    public float Deceleration { get => _deceleration; set => _deceleration = value; }
```

PaddleComponent

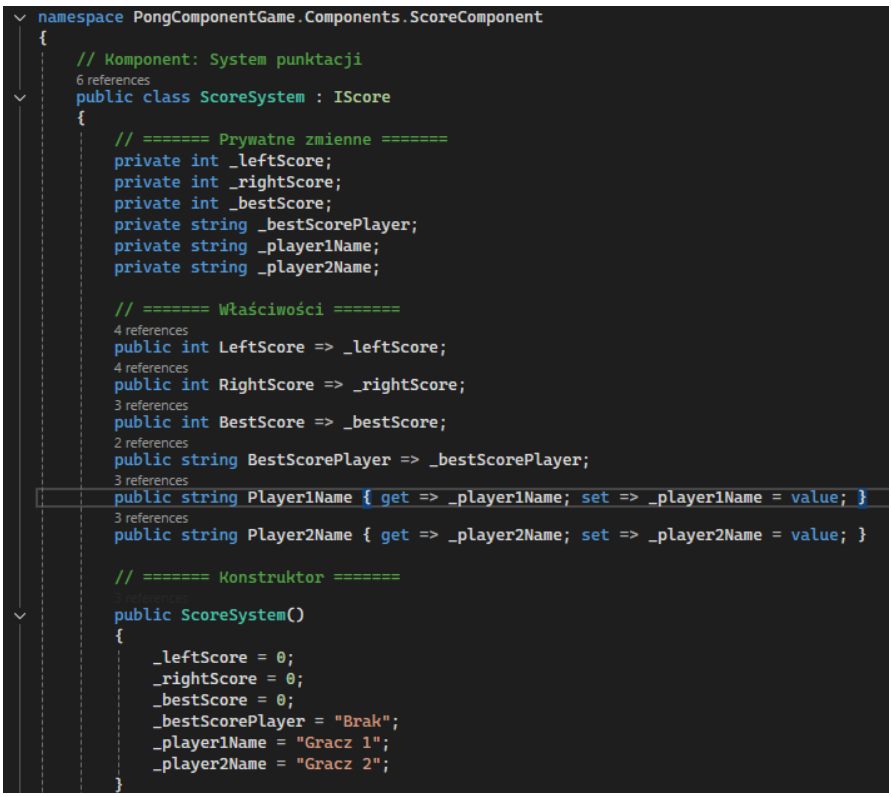
- Paletki dziedziczą po klasie abstrakcyjnej Paddle
- AI i gracz różnią się logiką sterowania
- Zmiennych prywatnych: 5+
- **Testy**: PaddleTests.cs



```
// Abstrakcyjny komponent Paddle
3 references
public abstract class Paddle : IPaddle
{
    // ======= Prywatne zmienne =======
    private int _x;
    private int _y;
    private int _height;
    private int _speed;
    private const int _width = 15;

    // ======= Właściwości =======
    7 references
    public int X { get => _x; set => _x = value; }
    13 references
    public int Y { get => _y; set => _y = value; }
    7 references
    public int Height { get => _height; set => _height = value; }
    2 references
    public int Width => _width;
    5 references
    public int Speed { get => _speed; set => _speed = value; }

    // ======= Konstruktor bezargumentowy =======
    0 references
    public Paddle()
    {
        _x = 0;
        _y = 0;
        _height = 100;
        _speed = 8;
```

ScoreComponent

- Zliczanie punktów, najlepszy wynik
- **Zapis/odczyt JSON**: tak
- **Testy**: ScoreTests.cs



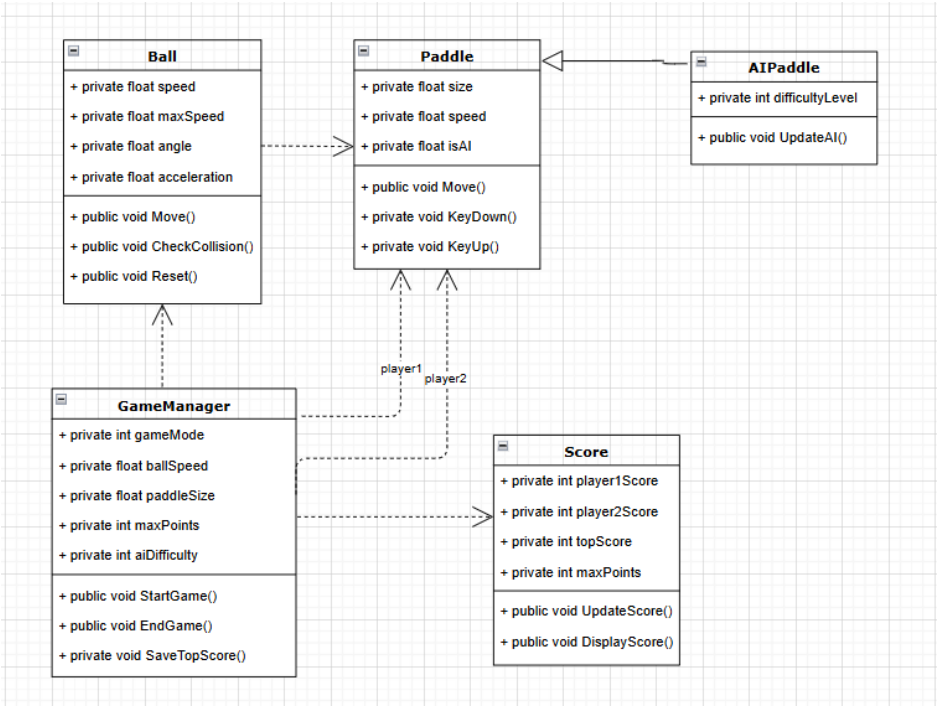
```
namespace PongComponentGame.Components.ScoreComponent
{
    // Komponent: System punktacji
    6 references
    public class ScoreSystem : IScore
    {
        // ======= Prywatne zmienne =======
        private int _leftScore;
        private int _rightScore;
        private int _bestScore;
        private string _bestScorePlayer;
        private string _player1Name;
        private string _player2Name;

        // ======= Właściwości =======
        4 references
        public int LeftScore => _leftScore;
        4 references
        public int RightScore => _rightScore;
        3 references
        public int BestScore => _bestScore;
        2 references
        public string BestScorePlayer => _bestScorePlayer;
        3 references
        public string Player1Name { get => _player1Name; set => _player1Name = value; }
        3 references
        public string Player2Name { get => _player2Name; set => _player2Name = value; }

        // ======= Konstruktor =======

        public ScoreSystem()
        {
            _leftScore = 0;
            _rightScore = 0;
            _bestScore = 0;
            _bestScorePlayer = "Brak";
            _player1Name = "Gracz 1";
            _player2Name = "Gracz 2";
        }
```

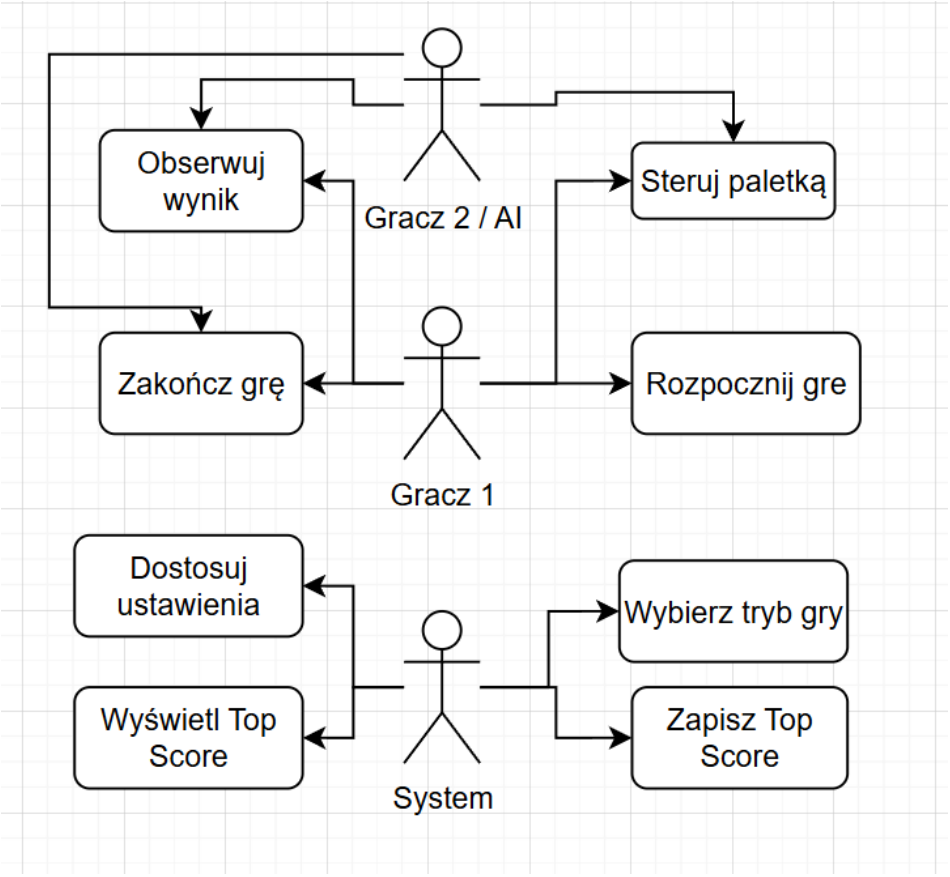
ConfigComponent

- Przechowuje ustawienia gry
- Możliwość zmiany przez formularz
- **Testy**: ConfigTests.cs

```
// Komponent: Konfiguracja gry
11 references
public class GameConfig : IConfigurable
{
    // ======= Prywatne zmienne =======
    private int _windowWidth;
    private int _windowHeight;
    private int _ballSpeed;
    private int _pointsToWin;
    private bool _isAgainstAI;
    private string _player1Name;
    private string _player2Name;
    private Keys _player1UpKey;
    private Keys _player1DownKey;
    private Keys _player2UpKey;
    private Keys _player2DownKey;

    // ======= Wartości domyślne =======

    public GameConfig()
    {
        _windowWidth = 800;
        _windowHeight = 600;
        _ballSpeed = 8;
        _pointsToWin = 10;
        _isAgainstAI = true;
        _player1Name = "Gracz 1";
        _player2Name = "Gracz 2";
        _player1UpKey = Keys.W;
        _player1DownKey = Keys.S;
        _player2UpKey = Keys.Up;
        _player2DownKey = Keys.Down;
    }

    // ======= Właściwości =======

    public int WindowWidth { get => _windowWidth; set => _windowWidth = value; }

    public int WindowHeight { get => _windowHeight; set => _windowHeight = value; }
    8 references
    public int BallSpeed { get => _ballSpeed; set => _ballSpeed = value; }

    public int PointsToWin { get => _pointsToWin; set => _pointsToWin = value; }
    7 references
    public bool IsAgainstAI { get => _isAgainstAI; set => _isAgainstAI = value; }
```

7. Diagramy klas i przypadków użycia





8. Wersja niekomponentowa (C#)

- Implementacja w jednym pliku: SimplePong.cs
- Brak podziału na klasy
- Wszystkie zmienne i logika (piłka, paletki, punktacja) zdefiniowane lokalnie
- Obsługa gry przez Form1, Timer i zdarzenia klawiatury

Zalety:

- Prostota
- Szybka implementacja

Wady:

- Brak modularności
- Trudna rozbudowa

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace SimplePong
{
    2 references
    public class Form1 : Form
    {
        Timer timer = new Timer();
        Rectangle ball;
        Rectangle paddle1, paddle2;
        int ballSpeedX = 5, ballSpeedY = 3;
        int paddleSpeed = 8;
        int paddle1Score = 0, paddle2Score = 0;
        bool up1, down1, up2, down2;

        1 reference
        public Form1()
        {
            this.Text = "Simple Pong";
            this.ClientSize = new Size(800, 600);
            this.DoubleBuffered = true;
            this.BackColor = Color.Black;

            ball = new Rectangle(390, 290, 20, 20);
            paddle1 = new Rectangle(20, 250, 15, 100);
            paddle2 = new Rectangle(765, 250, 15, 100);

            timer.Interval = 16;
            timer.Tick += GameLoop;
            timer.Start();

            this.KeyDown += OnKeyDown;
            this.KeyUp += OnKeyUp;
        }

        1 reference
        private void GameLoop(object sender, EventArgs e)
        {
```

---

9. Porównanie wersji komponentowej i niekomponentowej

Cecha	Komponentowa	Niekomponentowa
Modularność	✅	❌
Czytelność kodu	✅	❌
Rozszerzalność	✅	❌
Liczba plików	wiele	1
Testy jednostkowe	✅	❌

---

10. Testowanie

- `Debug.Assert` w testach komponentów
- `RunAll()` w testach Ball, Paddle, Score, Config

---

11. Wnioski końcowe

- Projekt spełnia wszystkie podstawowe wymagania ZPK
- Gra jest w pełni funkcjonalna
- Dzięki modularnej strukturze można łatwo rozwijać projekt
- Wersja uproszczona (SimplePong.cs) pokazuje kontrast i potwierdza zalety komponentów

---

12. Zestawienie użytych narzędzi

- Visual Studio 2022
- .NET 6 / Windows Forms
- JSON (serializacja)

---

13. Źródła / materiały pomocnicze

- Dokumentacja Microsoft C#
- Przykłady gier Pong online