* **Klasy …Enum (np BallEnum)**

Wszystkie robią to samo i nie wydają się potrzebne.

Kod w rodzaju:

public IEnumerator<Position> GetEnumerator()

{

return new PositionEnum(positions);

}

Można zamienić na:

public IEnumerator<Position> GetEnumerator()

{

return positions.GetEnumerator();

}

* **Interpolacja stringów**

Zamiast

Logger.Instance.Write(string.Format("Ball {0}: {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}",…

Można zrobić czytelniej:

Logger.Instance.Write($"Ball {(reverse ? "reverse" : "")}: {PosX}, {PosY}, {Width}, {Height}, {OffsetX}, {OffsetY}, {DirectionX}, {DirectionY}, {Iteration}, {Degree}");

* **Duplikacja w ctor Ball**

Degree = randomGenerator.GenerateDegree();

Można użyć metody CalculateNewDegree(), żeby nie duplikować kodu

* **Implementowanie ICollection**

IBallManager i inne klasy niepotrzebnie implementują ICollection bo wykorzystujesz chyba tylko Add i Count, a reszta kodu jest zbędna.

Można zrobić własny interfejs np. coś w stylu:

public interface IGameElementCollection<T> : IEnumerable<T>

{

int Count { get; }

void Add(T item);

}

i powyrzucać niepotrzebne metody Contains, CopyTo itd.

* **Zbędne rzutowanie na IEnumerable**

IEnumerator IEnumerable.GetEnumerator()

{

return (IEnumerator)GetEnumerator();

}

* **Properties w IElement**

public interface IElement

{

int PosX { get; set; }

int PosY { get; set; }

int Width { get; set; }

int Height { get; set; }

}

Zamiast 4 osobnych properties można by użyć struct.

Coś w stylu:

public struct Vector2

{

int X { get; set; }

int Y { get; set; }

}

public struct ElementBounds

{

Vector2 Min { get; set; } // X, Y

Vector2 Max { get; } // X + Width, Y + Height

Vector2 Size { get; set; } // Width, Height

}

W kilku miejscach jest kod w stylu first.PosX + first.Width. Zamiast tego można by używać property tego struct.

Może zamiast implementowania IElement zrobić dziedziczenie po klasie abstrakcyjnej?

public abstract Element

{

ElementBounds Bounds { get; set; }

}

Vector2 można by tez użyć w innych miejscach, gdzie jest PosX, PosY, DirectionX, DirectionY itp.