Raytracing: krok po kroku

cz. 3 - płaszczyzny

I. Co/Dlaczego?

W tym artykule zajmiemy się implementacją kolejnej figury z której będziemy budować świat przedstawiony - płaszczyzny.

II. Płaszczyzna: teoria?

Płaszczyznę można wyobrazić sobie jako płaskie pole rozciągające się w nieskończoność (na przykład podłogę nieskończenie wielkiej sali).
W ujęciu geometrii analitycznej, płaszczyzna to zbiór punktów spełniających równanie

$$Ax + By + Cz + D = 0$$

(Przy czym A, B, C nie mogą być jednocześnie równe 0).
Jest to tzw. równanie ogólne płaszczyzny. Wektor [A, B, C] to wektor prostopadły do płaszczyzny (normalna).
Dla naszych potrzeb, zamiast używać bezpośrednio równania ogólnego wygodniej jest reprezentować płaszczyznę jako normalną tej płaszczyzny oraz punkt przez który ta płaszczyzna przechodzi.

III. Implementacja

Wiedząc jakich parametrów potrzebujemy, łatwo napisać szkielet klasy reprezentującej płaszczyznę. Trzeba również pamiętać że jako renderowany obiekt musi ona dziedziczyć z abstrakcyjnej klasy GeometricObject. Gotowy kod:

```
class Plane : GeometricObject
{
    /// <summary>Punkt przez który płaszczyzna przechodzi</summary>
    Vector3 point;

    /// <summary>Normalna do płaszczyzny</summary>
    Vector3 normal;

    public Plane(Vector3 point, Vector3 normal, Color color)
    {
        this.point = point;
        this.normal = normal;
        base.Color = color;
    }

    public override bool HitTest(Ray ray, ref double distance)
    {
        throw new System.NotImplementedException();
    }
}
```

Pozostaje zaimplementować HitTest.
W tym celu podstawiamy równanie prostej (1) do równania płaszczyzny (2) otrzymując (3), które po rozwiązaniu dla t daje (4) (gdzie `a` to punkt na płaszczyźnie a `n` to normalna prostej)

$$(1): p = o + d * t$$

$$(2): (p - a) \cdot n = 0$$

$$(3): (o + d * t - a) \cdot n = 0$$

$$(4): t = (a - o) \cdot n / (d \cdot n)$$

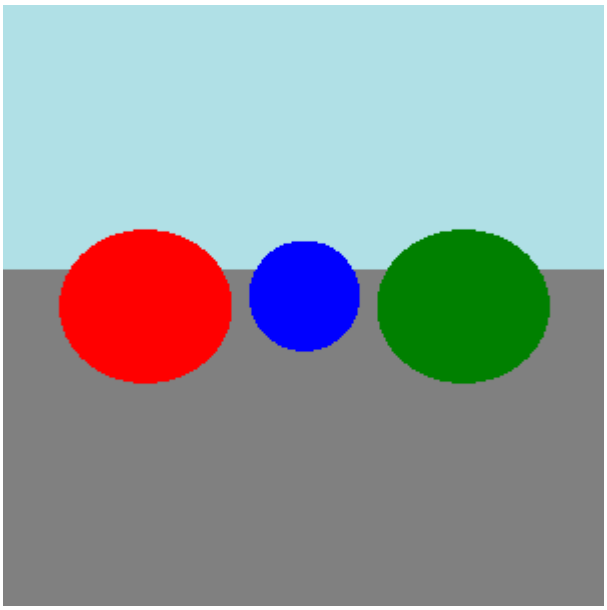
```
public override bool HitTest(Ray ray, ref double distance)
{
    double t = (point - ray.Origin).Dot(normal) / ray.Direction.Dot(normal);

    if (t > Ray.Epsilon)
    {
        distance = t;
        return true;
    }

    return false;
}
```

Co się stanie jeśli promień jest równoległy do płaszczyzny? Wtedy ray.Direction.Dot(normal) == 0, a dzielenie niezerowej wartoście przez zero, zgodnie ze standardem IEEE zwróci nieskończoność. Nie jest to problemem ponieważ trafienie zostanie uznane za 'nieskończenie dalekie' i nie będzie brane pod uwagę.

IV. ...Koniec?

Tak, tylko tyle jeśli chodzi o płaszczyznę. Możemy teraz dodać nowy obiekt do naszego świata i podziwiać (przy pewnej wyobraźni) jego piękno.

```
world.Add(new Plane(new Vector3(0, -2, 0), // punkt płaszczyzny
    new Vector3(0, 1, 0), // normalna płaszczyzny
    Color.Gray)); // kolor
```



Niniejszym skończyliśmy przygotowania - w następnej części rozpoczniemy pracę nad naprawdę ciekawymi rzeczami (i doprowadzimy do tego że w końcu nie trzeba się będzie domyślać co przedstawia obraz)...