Kontrola zámeny ôs X a Y

Pri testovaní hodnôt z Kalmanovho filtra sme odhalili, že dochádza k zámene osí X a Y. Podľa dokumentácie je os X od brány k bráne a os Y je umiestsnená podľa šírky ihriska. Pri strieľaní lopty podľa osi X na základe dokumentácie dochádzalo k zmene na osi Y a nie na osi X, tak ako by malo. Na základe toho sme usúdili že výpočet karteziánskych súradníc v triede *Vector3D.java* je nesprávny. Tento predpoklad sme potvrdili na základe dostupných zdrojov na internete pre prevod sférických súradníc na karteziánske.

Došlo k nasledujúcim zmenám:

- výmena X a Y súradnice pre výpočet karteziánskych súradníc
- oprava spätného prepočtu z karteziánskych súradníc na sférické
- pridanie atribútov pre uhly theta a phi v stupňoch
- výpis vektora s uhlami v stupňoch, namiesto v radiánoch

Pôvodný prevod sférických súradníc na karteziánske *(Vector3D.java):*

```
private void calculateCartesian() {
        x = cos(theta) * sin(phi) * -r;
        x = (int) (x * 1000.0) / 1000.0;
        y = cos(theta) * cos(phi) * r;
        y = (int) (y * 1000.0) / 1000.0;
        z = sin(theta) * r;
        z = (int) (z * 1000.0) / 1000.0;
}
```

Obr. 1: Pôvodná implementácia calculateCartesian()

Nový prevod sférických súradníc na karteziánske *(Vector3D.java*) :

```
private void calculateCartesian() {
    y = cos(theta) * sin(phi) * -r;
    y = (int) (y * 1000.0) / 1000.0;
    x = cos(theta) * cos(phi) * r;
    x = (int) (x * 1000.0) / 1000.0;
    z = sin(theta) * r;
    z = (int) (z * 1000.0) / 1000.0;

}
```

Obr. 2: Nová implementácia calculateCartesian()

Pôvodný prevod karteziánskych súradníc na sférické *(Vector3D.java)*:

```
private void calculateSpherical() {
        r = sqrt(x * x + y * y + z * z);
        theta = asin(z / r);
        if (r == 0.0)
                theta = 0.0;
        phi = atan2(y, x);
        phi -= PI / 2.0d;

        phi = Angles.normalize(phi);
        theta = Angles.normalize(theta);
}
```

Obr. 3: Pôvodná implementácia calculateSpherical()

Nový prevod karteziánskych súradníc na sférické *(Vector3D.java)*:

```
private void calculateSpherical() {
    r = sqrt(x * x + y * y + z * z);
    theta = asin(z / r);
    if (r == 0.0)
        theta = 0.0;
    phi = atan2(x, y);
    phi -= PI / 2.0d;

    phi_deg = toDegrees(phi);
    theta_deg = toDegrees(theta);

    phi = Angles.normalize(phi);
    theta = Angles.normalize(theta);
}
```

Obr. 4: Nová implementácia calculateSpherical()

Vytvorenie vektora pomocou sférických súradníc *(Vector3D.java)*:

```
public static Vector3D spherical(double r, double phi, double theta) {
    Vector3D vector = new Vector3D();
    vector.r = r;
    vector.phi = phi;
    vector.theta = theta;
    vector.theta_deg = toDegrees(theta);
    vector.phi_deg = toDegrees(phi);
    vector.calculateCartesian();
    return vector;
}
```

Obr. 5: Pridanie atribútov pre uhly v stupňoch

Výpis objektu Vector3D so súradnicami v stupňoch *(Vector3D.java)*:

```
@Override
public String toString() {
    return String
            .format("x, y, z: [%.2f, %.2f, %.2f] r, phi, theta: [%.2f, %.2f, %.2f]",
                    x, y, z, r, phi_deg , theta_deg);
}
```

Obr. 6: Formátovanie výpisu sférických súradníc