UNIVERSITÀ DEGLI STUDI DI CASSINO E DEL LAZIO MERIDIONALE

Facoltà di Ingegneria

Corso di laurea in Ingegneria Informatica e delle Telecomunicazioni

Simulazione della Dinamica di Corpi Rigidi Rettangolari

Relatore
Prof.Alessandro Bria

Candidato
Achille Cannavale
Matr. 0058721

A.A. 2022/2023

Contents

Introduzione

Lo scopo di questa tesi è la progettazione e l'implementazione di un game engine in C++ specializzato nella verifica e nella risoluzione delle collisioni tra corpi rigidi rettangolari.

L'obiettivo è quello di creare un'architettura solida e performante che possa essere utilizzata per lo sviluppo di giochi 2D di differenti generi, come platformer, metroidvania, arcade e gestionali.

Chapter 1

Collision Detection

1 Separating Axis Theorem

First of all collect all vertices of both object into a vectort:

```
std::vector <Vector2f> verticesA = objA->CalculateVertices();
std::vector <Vector2f> verticesB = objB->CalculateVertices();
```

Then calculate each normal of all edge, with this relationship:

$$normal(\bar{ab}) = \left[y\ ,\ -x\right]$$

```
Vector2f CalculateNormal(Vector2f pointA, Vector2f pointB) {
    Vector2f directionVector = pointB - pointA;

    return Vector2f(directionVector.y, -directionVector.x);
}

std::array <Vector2f, 4> CalculateNormals(std::array <Vector2f, 4>& vertices) {
    std::array <Vector2f, 4> normals(vertices.size());

    for (int i = 0; i < vertices.size(); i++) {
        normals[i] = CalculateNormal(vertices[i + 1 % vertices.size()], vertices[i])
    }
    normals[vertices.size() - 1] = CalculateNormal(vertices[0], vertices[vertices.si
    return normals;
}
```

Now, for each normal, we calculate the projection of every vertex on the normal:

$$Projection = vertex \cdot normal$$

And then we calculate the the leastest and the greatest projection for both figures. Finally ‘if (maxA < minB || maxB < minA)‘ there are not collisions. Else we have a collision and we have to define:

$$axisDepth = \min(maxB - minA,\ maxA - minB)$$

```
for (Vector2f normal : normals) {
        float minA = INFINITY;
        float maxA = -INFINITY;
        for (Vector2f vertex : verticesA) {
            float proj = Math::Dot(vertex,normal);
            minA = std::min(minA, proj);
            maxA = std::max(maxA, proj);
        }
        float minB = INFINITY;
        float maxB = -INFINITY;
        for (Vector2f vertex : verticesB) {
            float proj = Math::Dot(vertex, normal);
            minB = std::min(minB, proj);
            maxB = std::max(maxB, proj);
        }
        if (maxA < minB || maxB < minA) {

            result.setAreColliding(false);
            result.setCollidingAxis(Vector2f(0, 0));
            result.setDepth(0);
            return result;
        }
        float axisDepth = (std::min(maxB - minA, maxA - minB));
        if (axisDepth < depth) {
            depth = axisDepth;
            result_normal = normal;
        }
    }
```