Collision Detection

Separating Axis Theorem

jintaeks@dongseo.ac.kr
May 2020

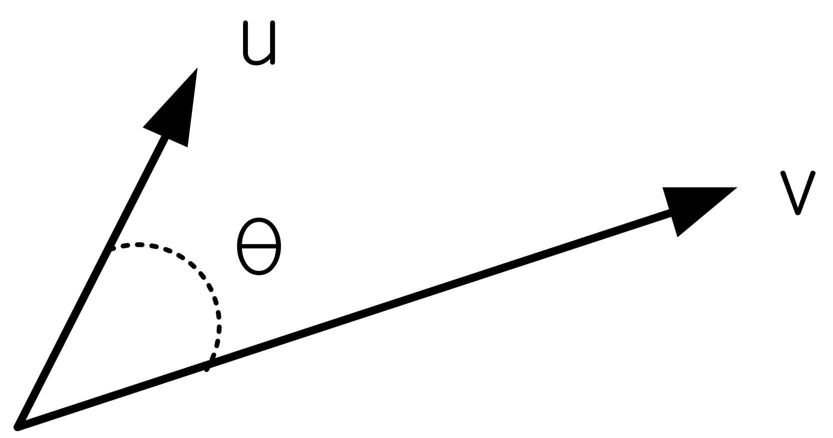
Dongseo University
동서대학교

Inner Product

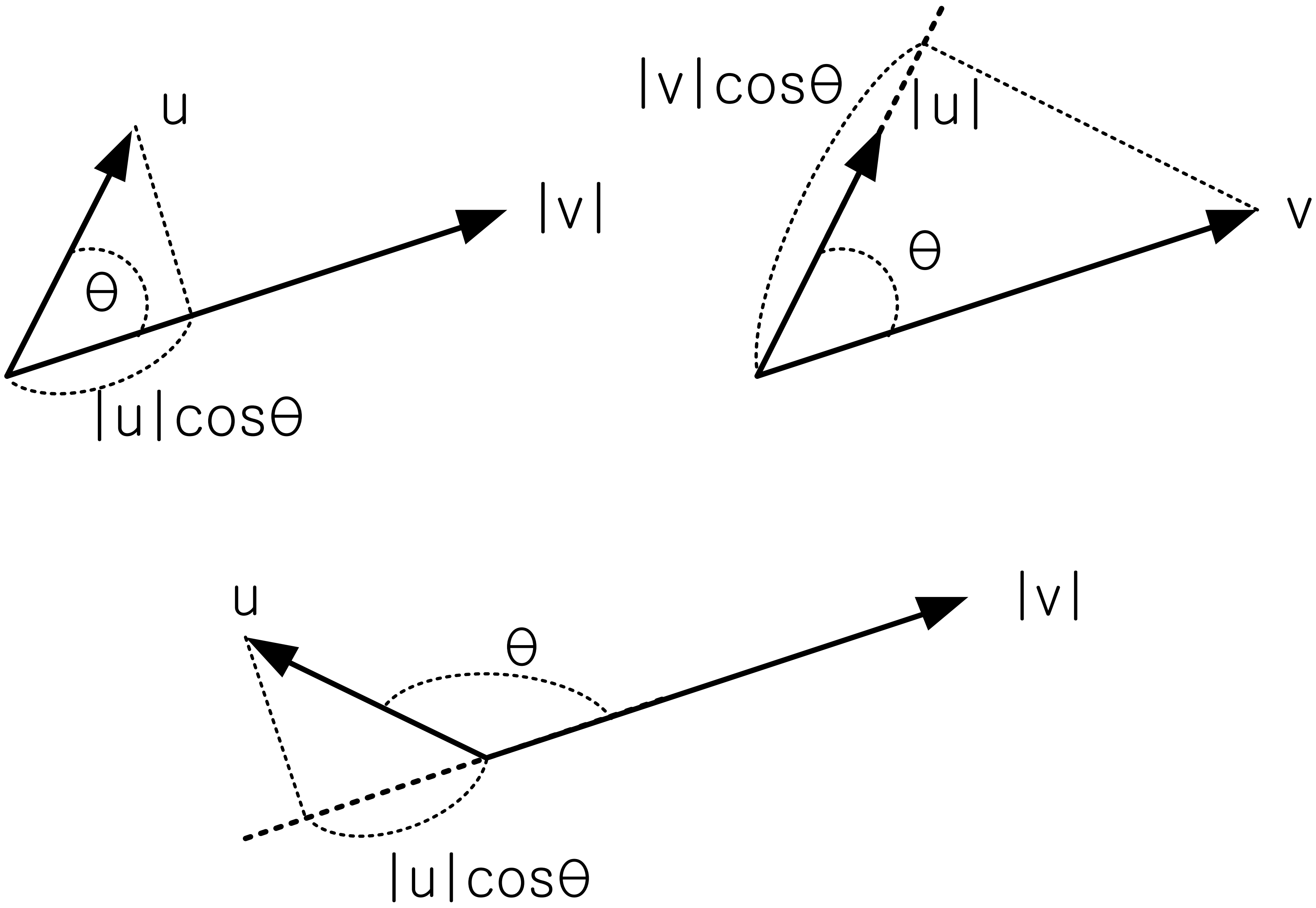
$$u \cdot v = \begin{cases} |u||v|\cos\theta, & \text{if } u \neq 0 \text{ and } v \neq 0 \\ 0, & \text{if } u = 0 \text{ or } v = 0 \end{cases}$$

$$|u||v|\cos\theta = u_1 v_1 + u_2 v_2 + u_3 v_3$$

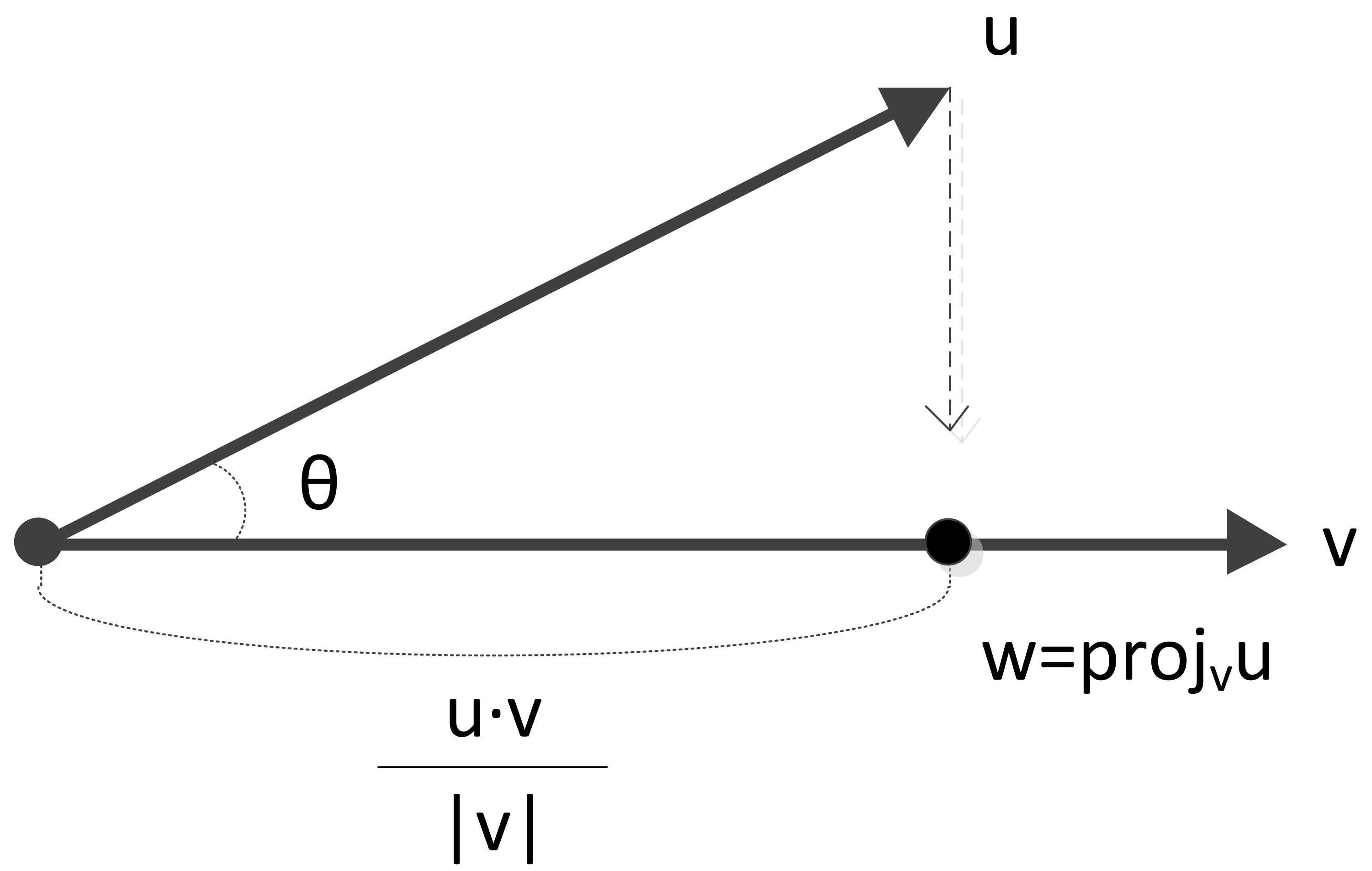
$u \cdot v = |u||v|\cos\theta$



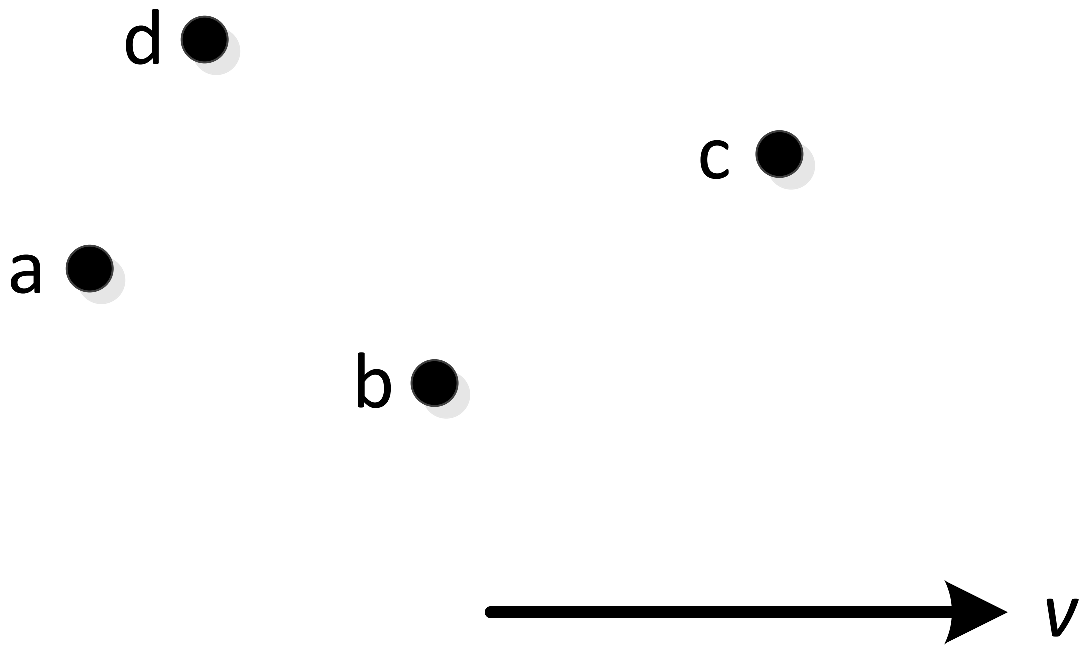
Inner Product Geometric Meaning

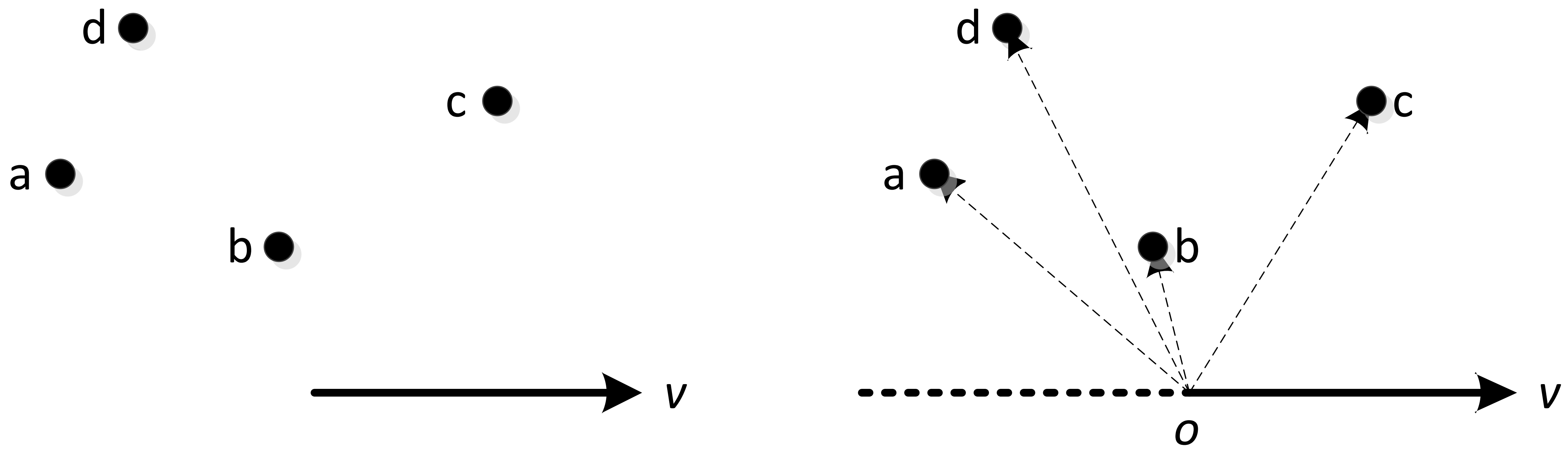


Vector Decomposition



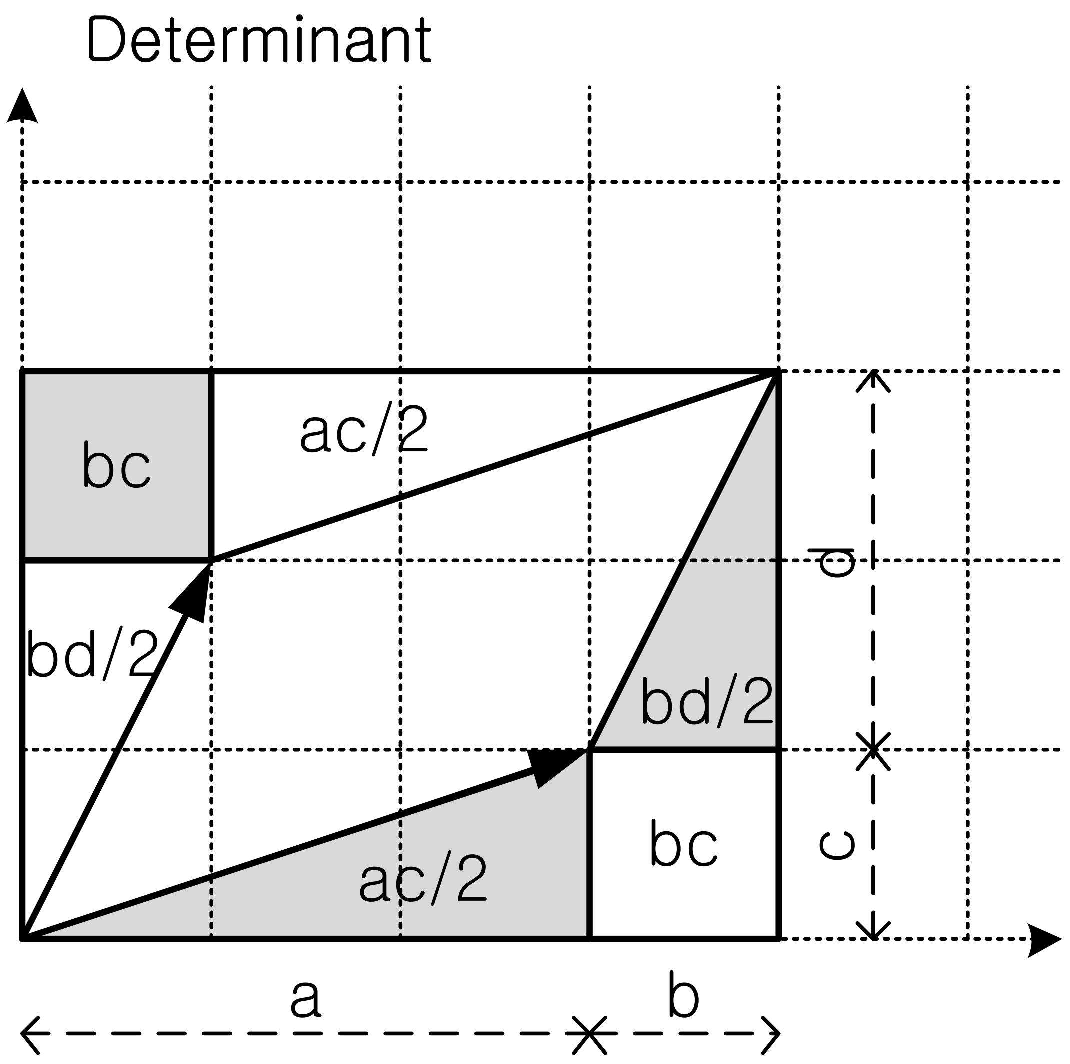
Sort points with respect to a vector




d
c
a
b
v
d
c
a
b
v
o

Determinant

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$



(a+b)(c+d)−ac−bd−2bc=ad−bc

$$
\begin{aligned}
|A| &= \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} \\
&= a\begin{vmatrix} \square & \square & \square \\ \square & e & f \\ \square & h & i \end{vmatrix} - b\begin{vmatrix} \square & \square & \square \\ d & \square & f \\ g & \square & i \end{vmatrix} + c\begin{vmatrix} \square & \square & \square \\ d & e & \square \\ g & h & \square \end{vmatrix} \\
&= a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix} \\
&= aei - afh - bdi + bfg + cdh - ceg
\end{aligned}
$$

$$u = (a, b, c)$$
$$v = (d, e, f)$$

$$u' = (a, b, 0)$$
$$v' = (d, e, 0)$$

$$\begin{vmatrix} a & d \\ b & e \end{vmatrix} = \begin{vmatrix} a & b \\ d & e \end{vmatrix} = ae - bd = z^{ignore}$$

$$\begin{vmatrix} a & b & 0^{ignore} \\ d & e & 0^{ignore} \end{vmatrix} = ae - bd = z^{ignore}$$

augmented determinant

$$\begin{vmatrix} 0^{ignore} & b & c \\ 0^{ignore} & e & f \end{vmatrix} = x^{ignore}$$

$$-\begin{vmatrix} a & 0^{ignore} & c \\ d & 0^{ignore} & f \end{vmatrix} = y^{ignore}$$

$$\begin{vmatrix} a & b & 0^{ignore} \\ d & e & 0^{ignore} \end{vmatrix} = z^{ignore}$$

What does this mean?

$$(x^{ignore}, y^{ignore}, z^{ignore})$$

Pseudo determinant

$$
\begin{aligned}
|A| &= \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} \\
&= a\begin{vmatrix} \square & \square & \square \\ \square & e & f \\ \square & h & i \end{vmatrix} - b\begin{vmatrix} \square & \square & \square \\ d & \square & f \\ g & \square & i \end{vmatrix} + c\begin{vmatrix} \square & \square & \square \\ d & e & \square \\ g & h & \square \end{vmatrix} \\
&= a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix} \\
&= aei - afh - bdi + bfg + cdh - ceg
\end{aligned}
$$

$$
\begin{aligned}
P &= \begin{vmatrix} \overrightarrow{i} & \overrightarrow{j} & \overrightarrow{k} \\ a & b & c \\ d & e & f \end{vmatrix} \\
&= \vec{i}\begin{vmatrix} \square & \square & \square \\ \square & b & c \\ \square & e & f \end{vmatrix} - \vec{j}\begin{vmatrix} \square & \square & \square \\ a & \square & c \\ d & \square & f \end{vmatrix} + \vec{k}\begin{vmatrix} \square & \square & \square \\ a & b & \square \\ d & e & \square \end{vmatrix} \\
&= \vec{i}\begin{vmatrix} b & c \\ e & f \end{vmatrix} - \vec{j}\begin{vmatrix} a & c \\ d & f \end{vmatrix} + \vec{k}\begin{vmatrix} a & b \\ d & e \end{vmatrix} \\
&= (x^{ignore}, y^{ignore}, z^{ignore})
\end{aligned}
$$

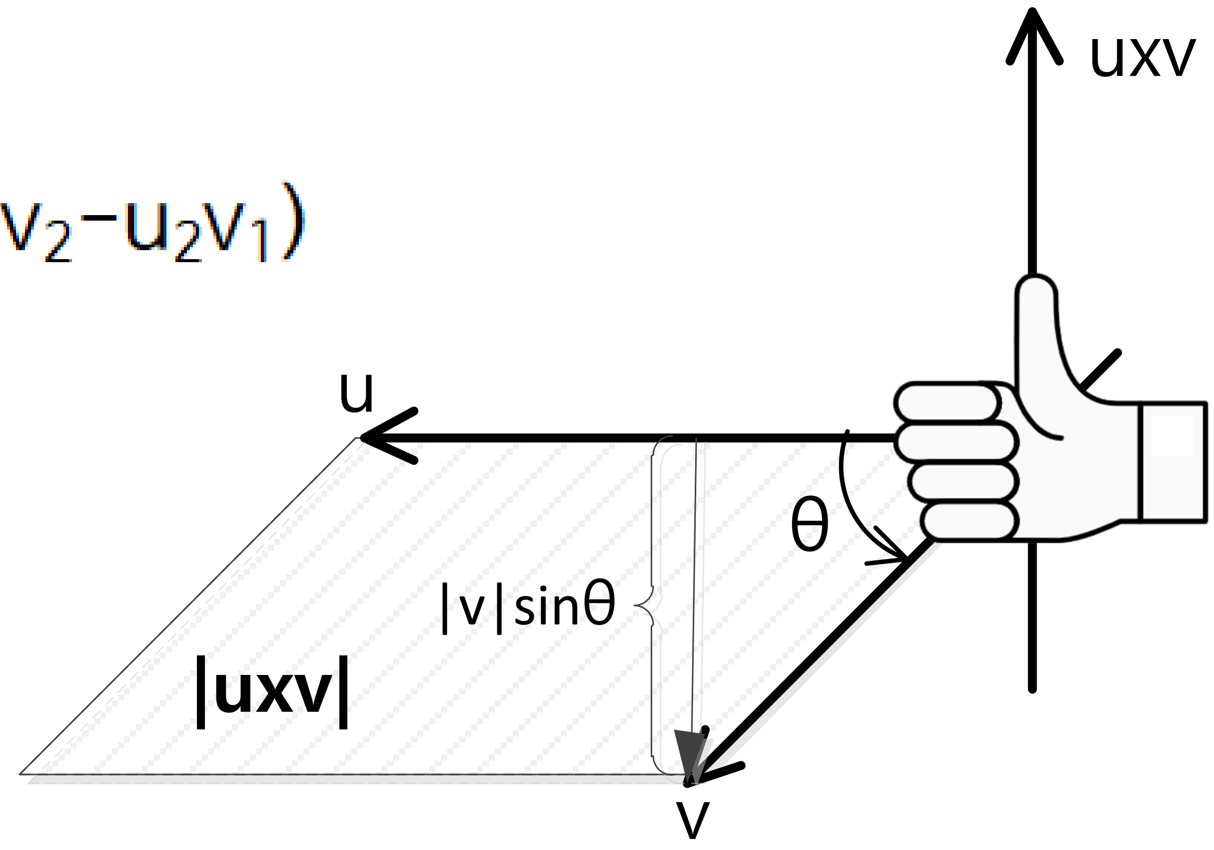
$$
\begin{aligned}
P &= \begin{vmatrix} \overrightarrow{i} & \overrightarrow{j} & \overrightarrow{k} \\ a & b & 0 \\ d & e & 0 \end{vmatrix} \\
&= \vec{i} \begin{vmatrix} \square & \square & \square \\ \square & b & 0 \\ \square & e & 0 \end{vmatrix} - \vec{j} \begin{vmatrix} \square & \square & \square \\ a & \square & 0 \\ d & \square & 0 \end{vmatrix} + \vec{k} \begin{vmatrix} \square & \square & \square \\ a & b & \square \\ d & e & \square \end{vmatrix} \\
&= \vec{i} \begin{vmatrix} b & 0 \\ e & 0 \end{vmatrix} - \vec{j} \begin{vmatrix} a & 0 \\ d & 0 \end{vmatrix} + \vec{k} \begin{vmatrix} a & b \\ d & e \end{vmatrix} \\
&= 0\vec{i} - 0\vec{j} + \vec{k} \begin{vmatrix} a & b \\ d & e \end{vmatrix} \\
&= (0, 0, \begin{vmatrix} a & b \\ d & e \end{vmatrix})
\end{aligned}
$$

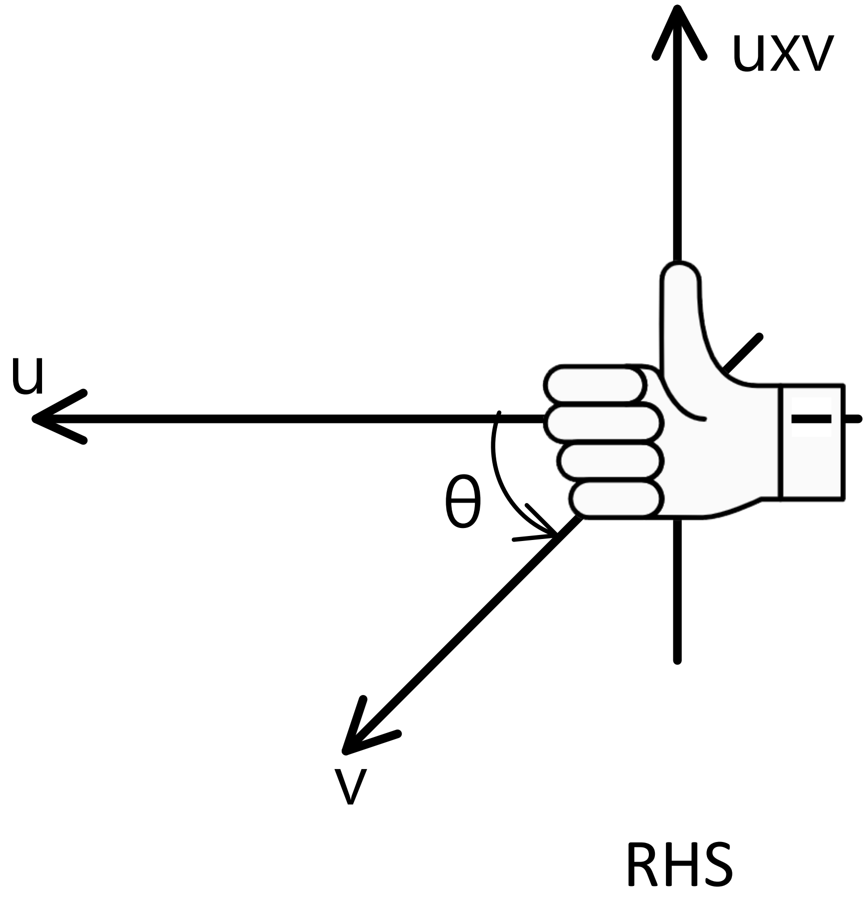
Cross Product

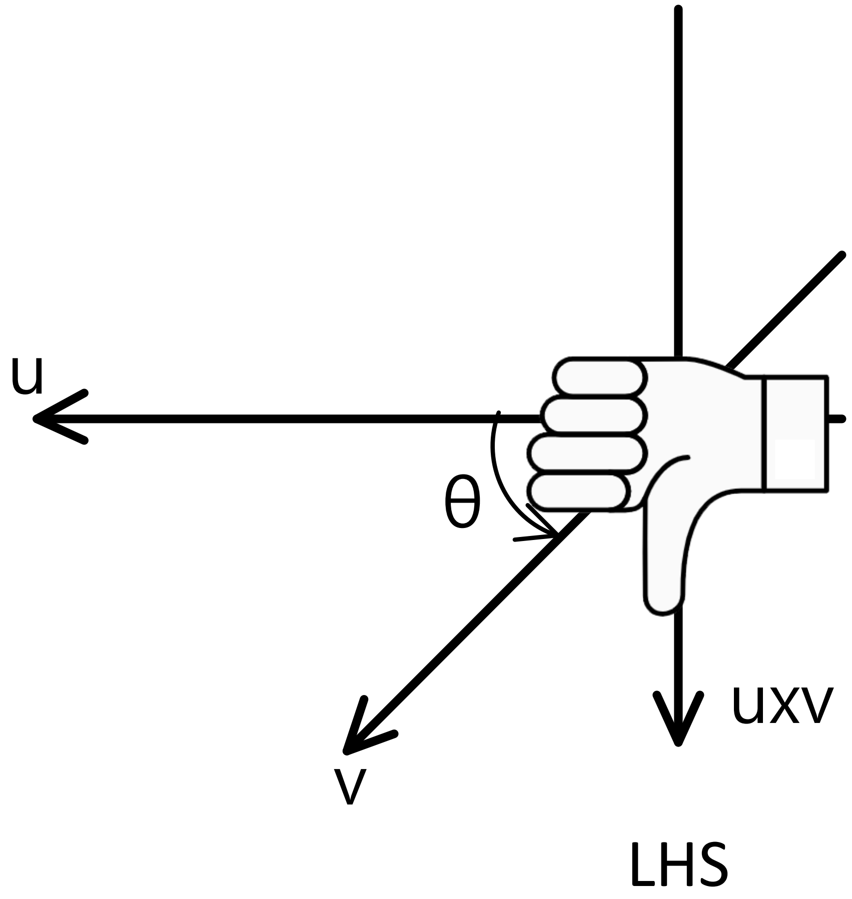
$$u = (u_1,\ u_2,\ u_3),\ v = (v_1,\ v_2,\ v_3)$$

$$u \times v = (u_2v_3 - u_3v_2,\ u_3v_1 - u_1v_3,\ u_1v_2 - u_2v_1)$$

$$|u \times v| = |u||v|\sin\theta$$




uxv
u
θ
v
RHS


u
θ
uxv
v
LHS

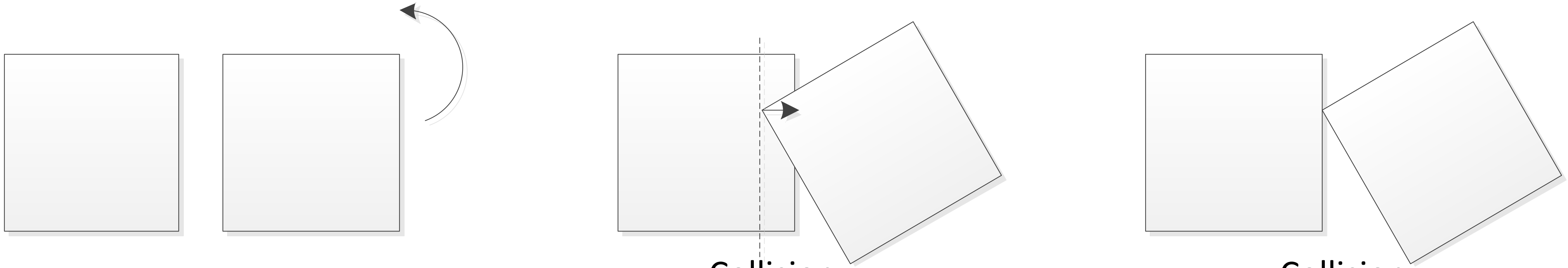
90 degree rotation in 2D space

$$\hat{i} = (1,0), \quad \hat{j} = (0,1)$$

$$|A| = \begin{vmatrix} \hat{i} & \hat{j} \\ a & b \end{vmatrix}$$

$$b\hat{i} - a\hat{j} = (b, -a)$$

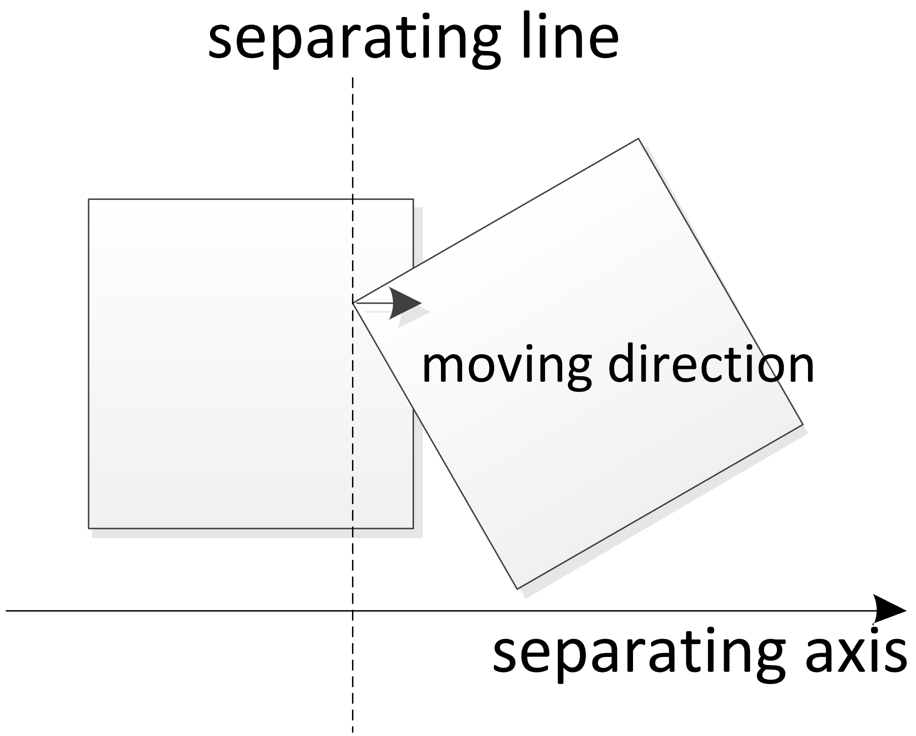
Physics Engine

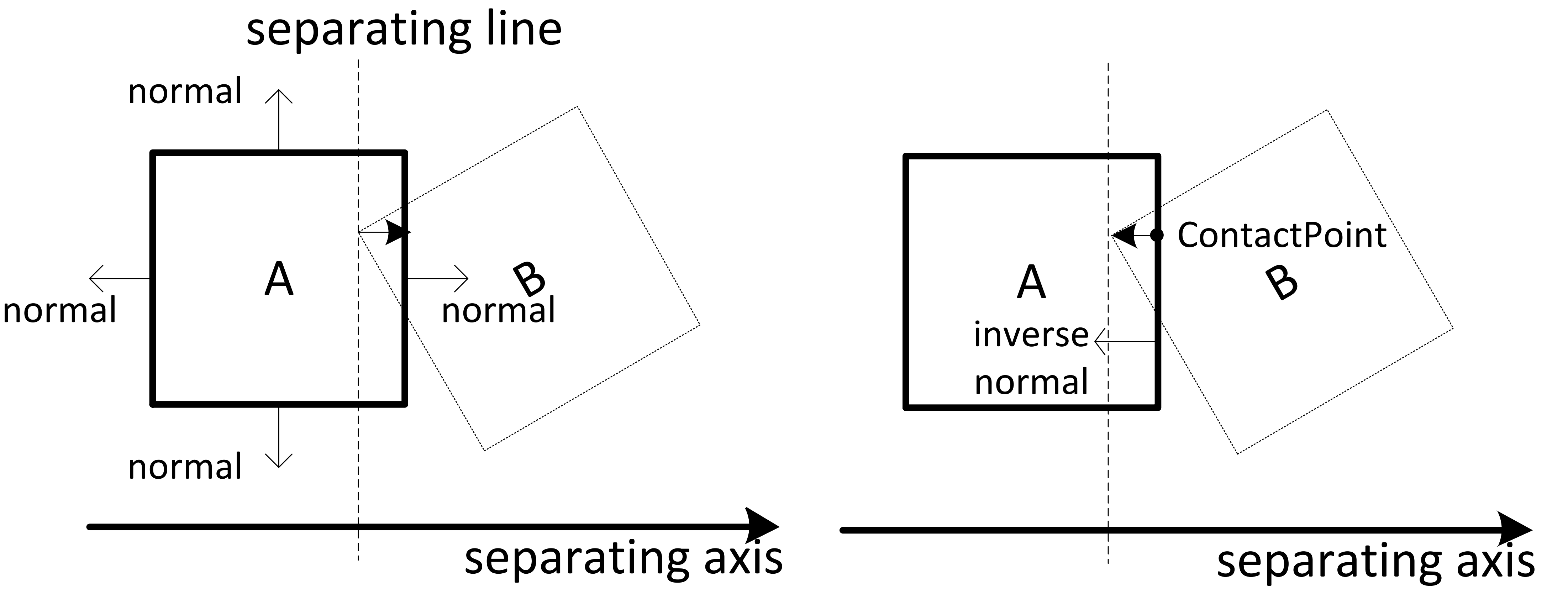


Simulation

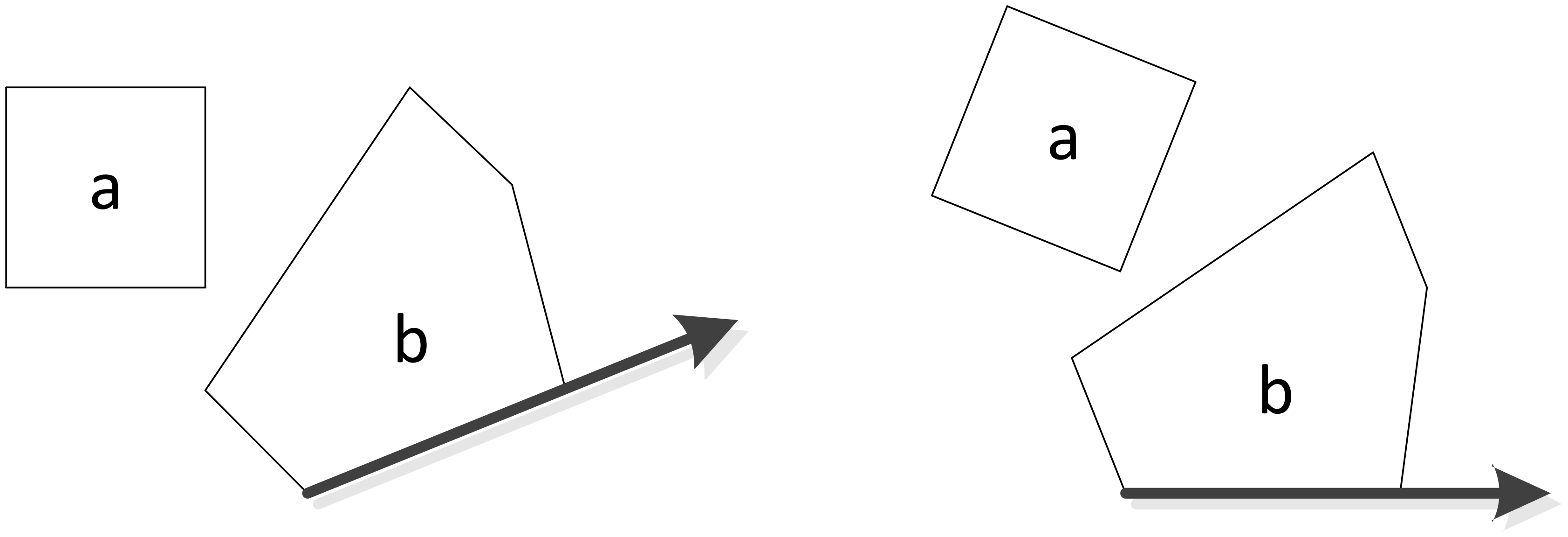
Collision Detection

Collision Response


separating line
moving direction
separating axis

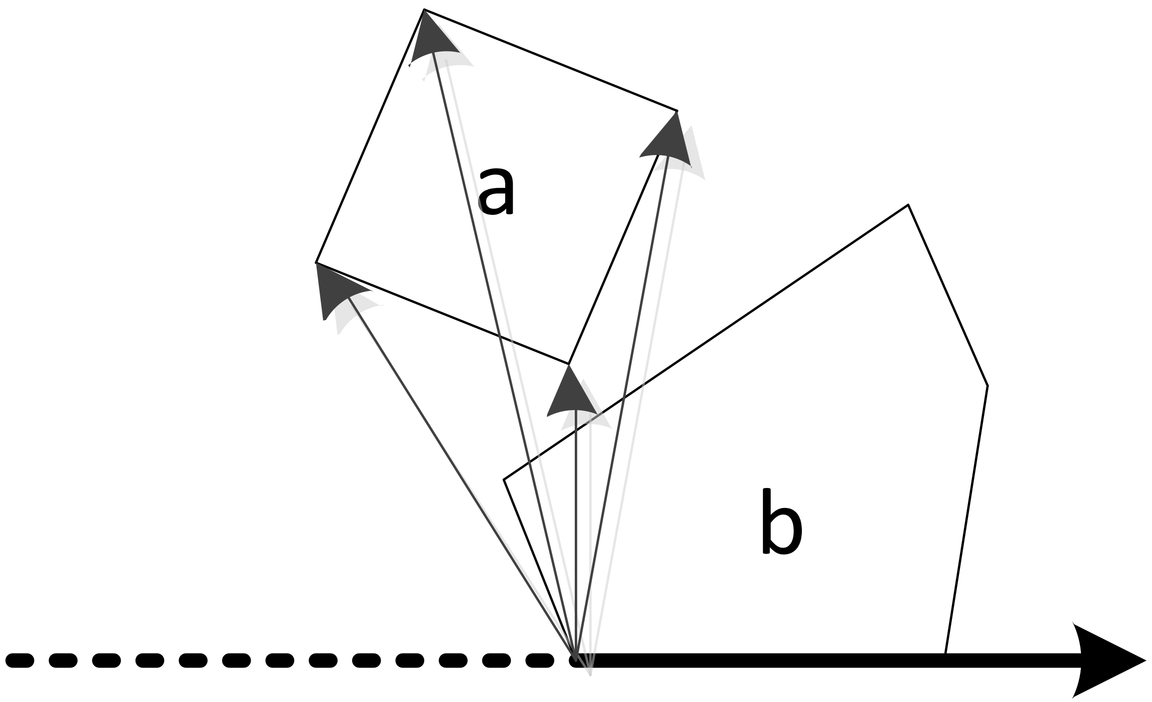

separating line
normal
normal
A
B
normal
normal
separating axis
ContactPoint
A
B
inverse
normal
separating axis

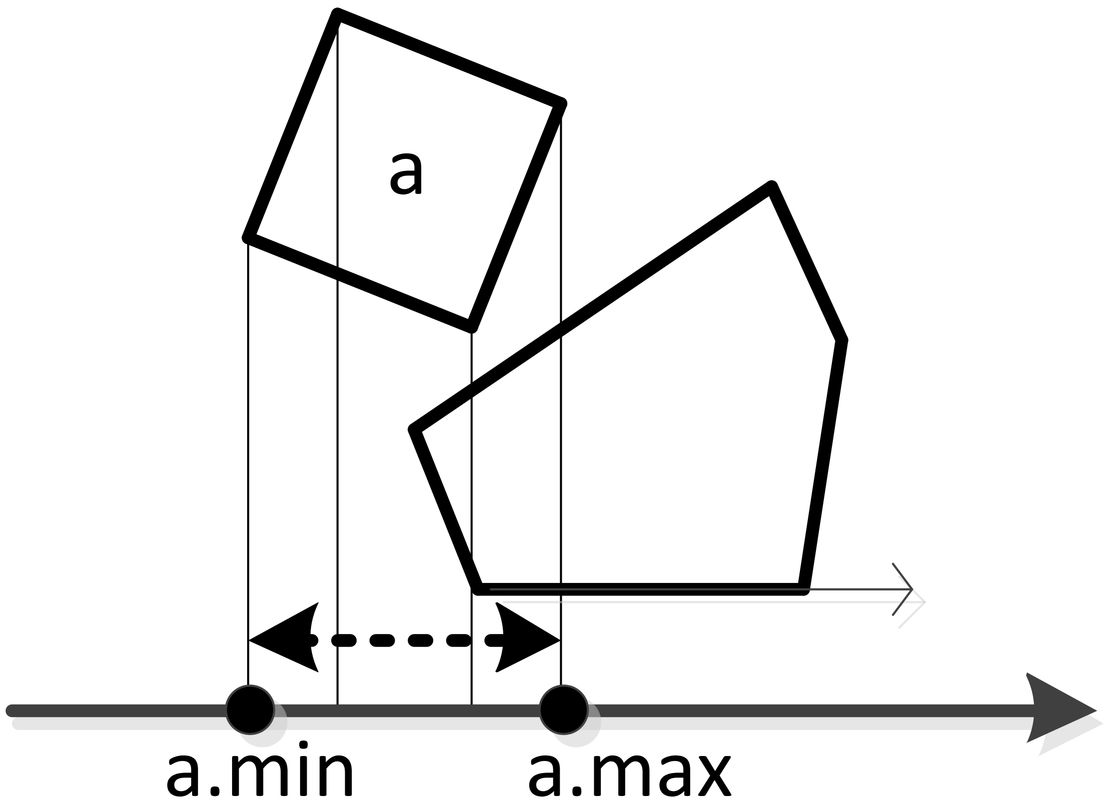
Polygon Collision Detection

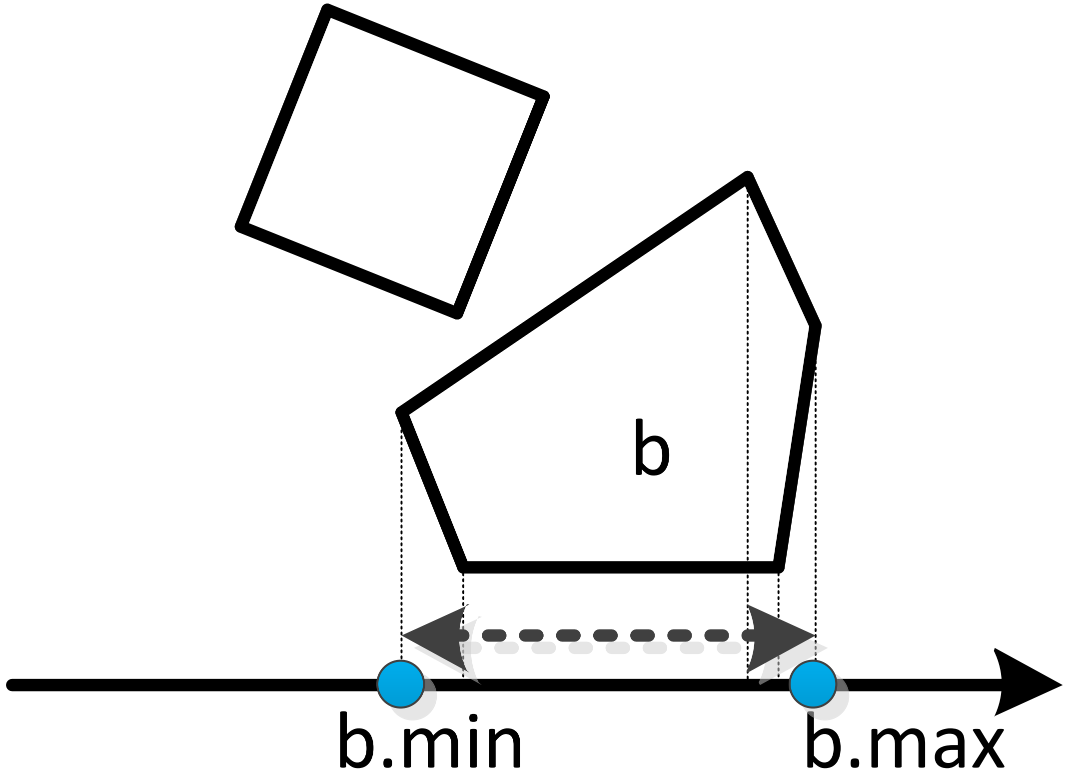


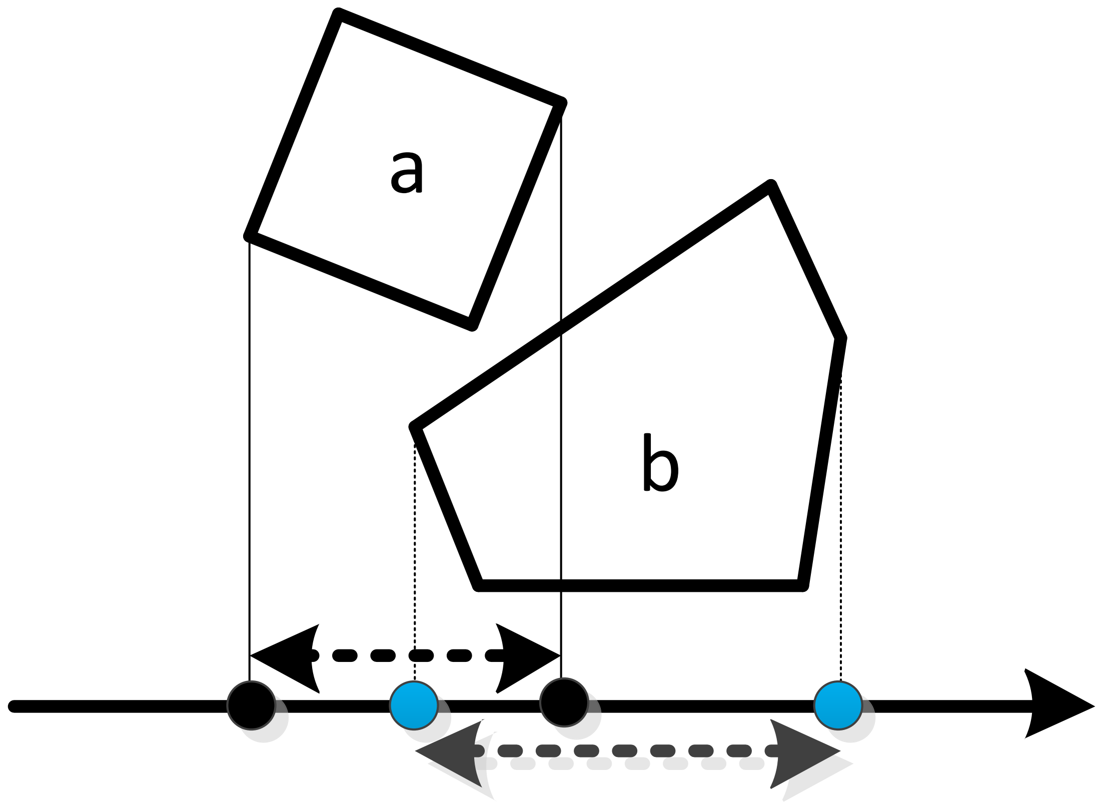
Dongseo University 동서대학교

Project all vertices to a target edge vector

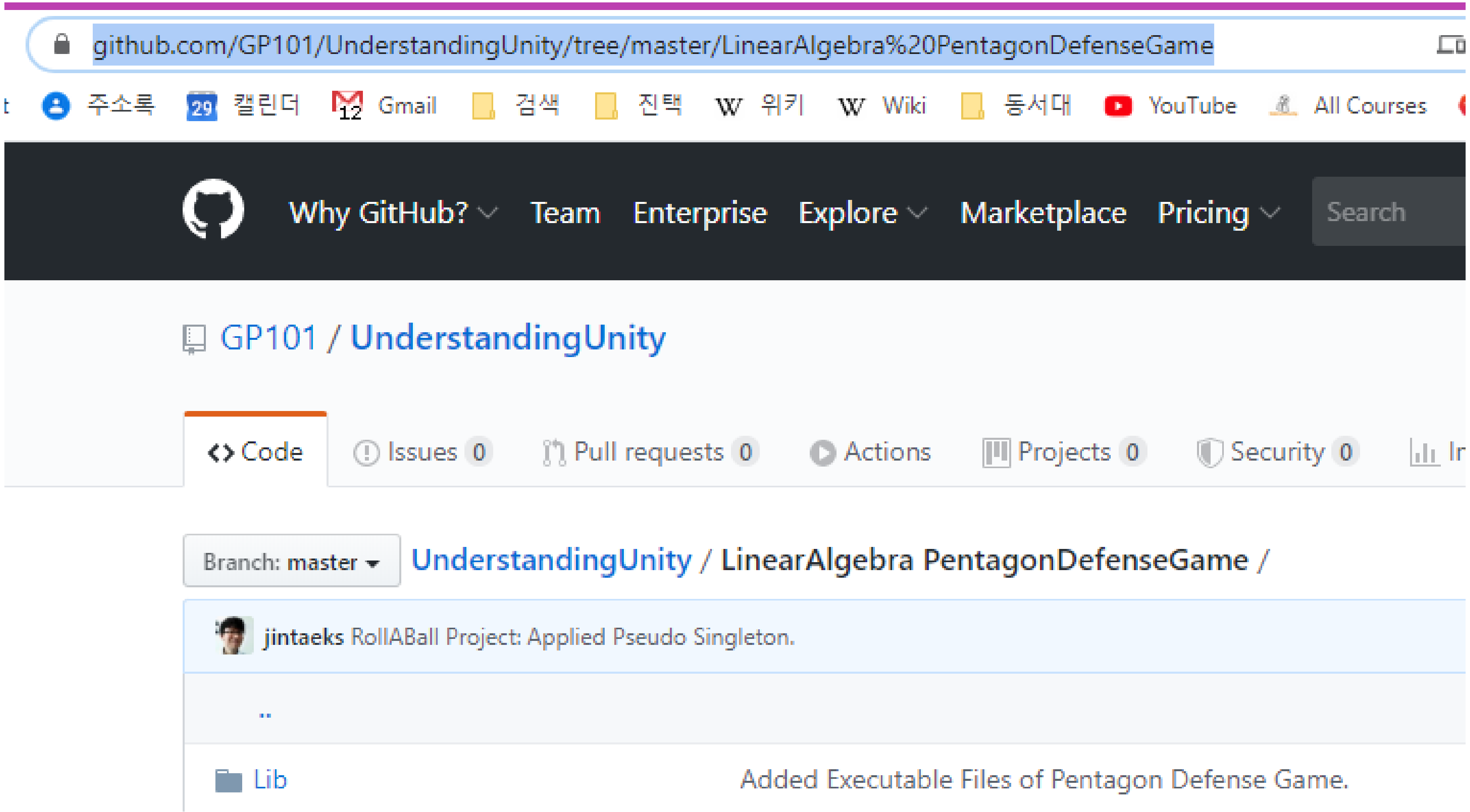



a
a.min
a.max


b
b.min
b.max


a
b

Implementation

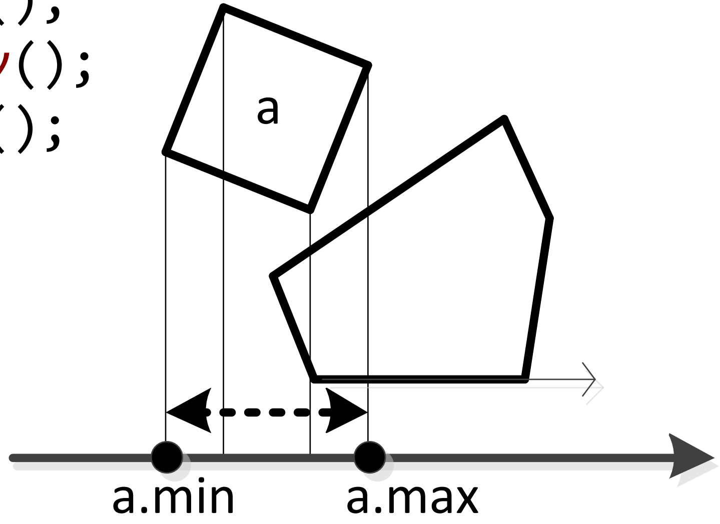


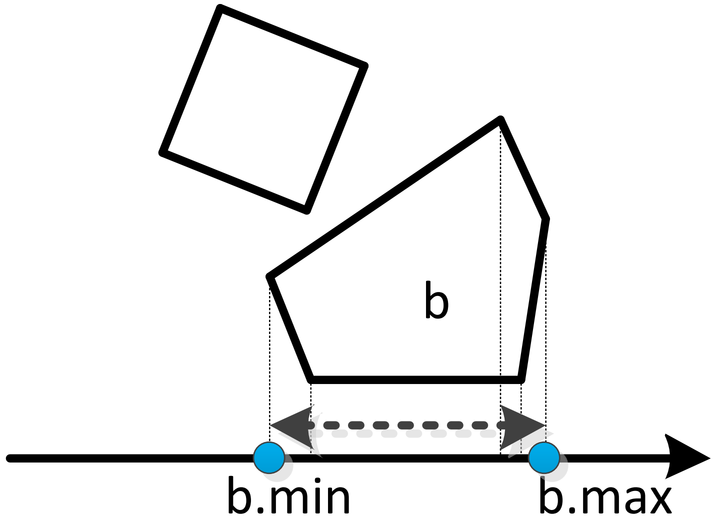

```cpp
using KPolygon = std::vector<KVector2>;
```

```cpp
class KVector2
{
public:
    static KVector2 zero;
    static KVector2 one;
    static KVector2 right;
    static KVector2 up;

    static KVector2 Lerp(const KVector2& begin, const KVector2& end, float ratio);
    static float Dot(const KVector2& left, const KVector2& right);

public:
    float   x;
    float   y;

public:
    KVector2(float tx=0.0f, float ty=0.0f) { x = tx; y = ty; }
    KVector2(int tx, int ty) { x = (float)tx; y = (float)ty; }
    float Length() const;
    void Normalize();
    float& operator[](int i);
};
```

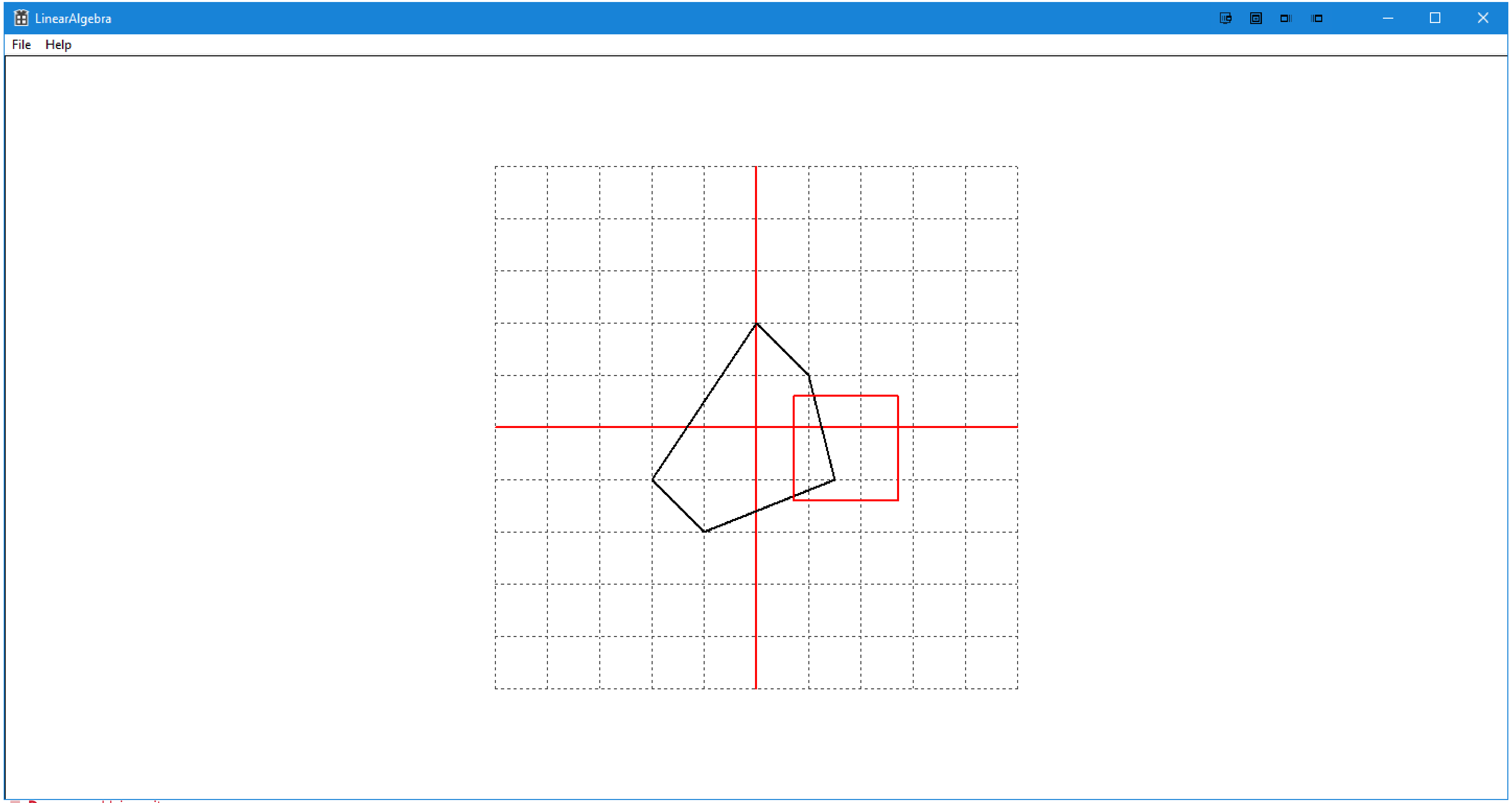
```cpp
bool _IntersectInternal(const KPolygon& a, const KPolygon& b)
{
    // loop over the vertices(-> edges -> axis) of the first polygon
    for (auto i = 0u; i < a.size() + 0; ++i) {
        // calculate the normal vector of the current edge
        // this is the axis will we check in this loop
        auto current = a[i];
        auto next = a[(i + 1) % a.size()];
        auto edge = next - current;

        KVector2 axis{};
        axis[0] = -edge[1];
        axis[1] = edge[0];
        //axis.Normalize();
```

```cpp
        auto aMaxProj = -std::numeric_limits<float>::infinity();
        auto aMinProj = std::numeric_limits<float>::infinity();
        auto bMaxProj = -std::numeric_limits<float>::infinity();
        auto bMinProj = std::numeric_limits<float>::infinity();
        for (const auto& v : a) {
            auto proj = KVector2::Dot(axis, v);
            if (proj < aMinProj) aMinProj = proj;
            if (proj > aMaxProj) aMaxProj = proj;
        }

        for (const auto& v : b) {
            auto proj = KVector2::Dot(axis, v);
            if (proj < bMinProj) bMinProj = proj;
            if (proj > bMaxProj) bMaxProj = proj;
        }

        if (aMaxProj < bMinProj || aMinProj > bMaxProj) {
            return false;
        }
    }

    return true;
}
```

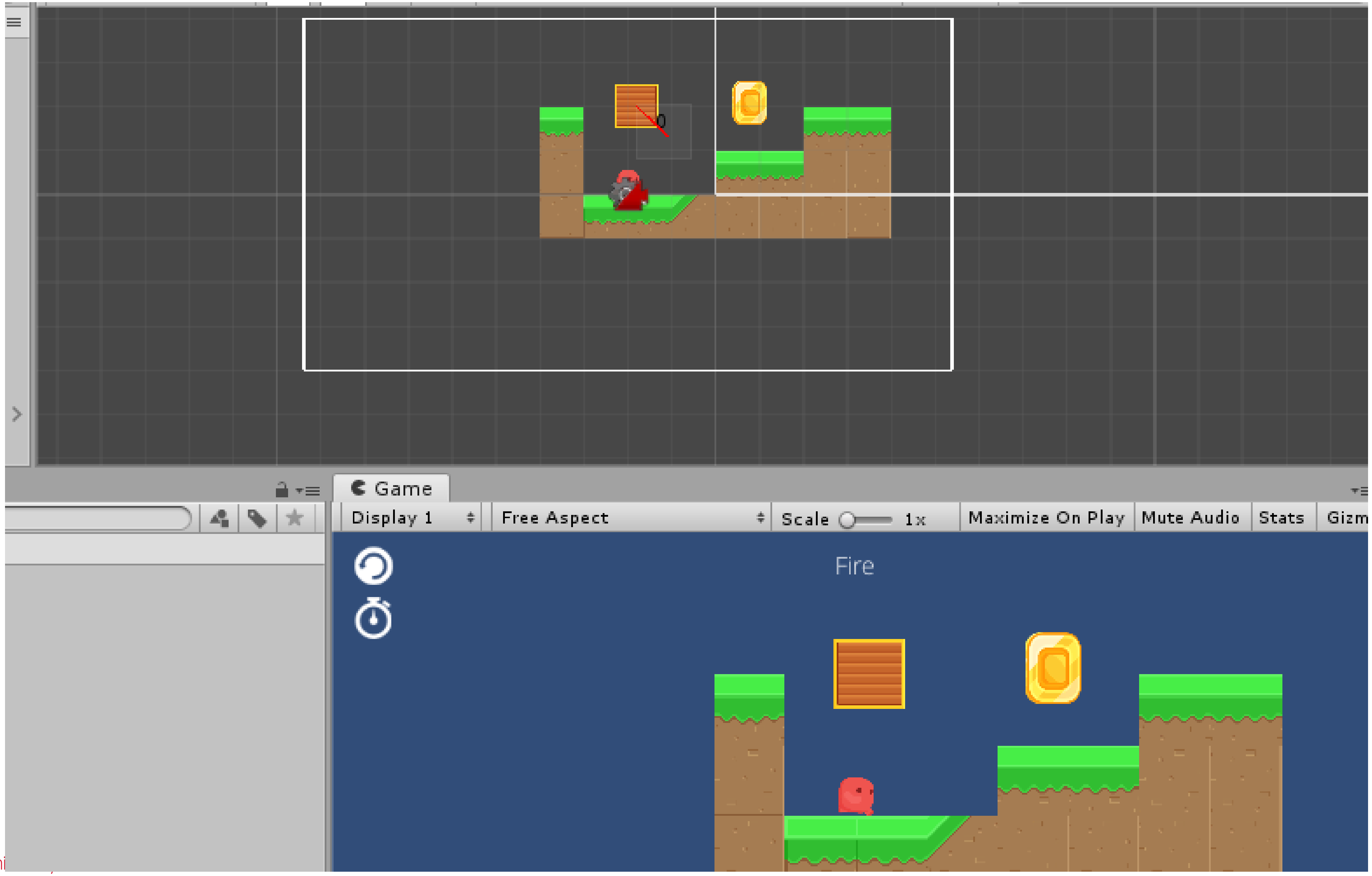




Demo in Windows GDI



Demo in Unity



✓ https://github.com/GP101/UnderstandingUnity/tree/master/LinearAlgebra%20PentagonDefenseGame

MY BRIGHT FUTURE

DSU Dongseo University 동서대학교