DYNAMIC GJK

Homepage: https://sites.google.com/site/doc4code/

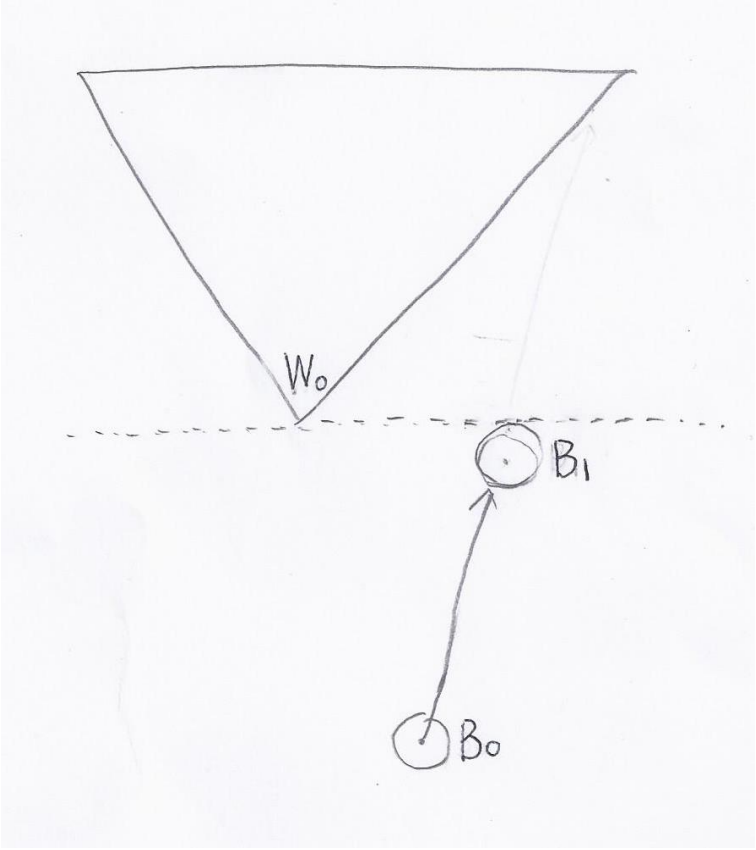
Email: sj6219@hotmail.com

2013/5/27

이 문서에서는 두 convex 물체가 이동할 때, 물체가 충돌하는 순간을 구하는 방법에 대해 구현해 보겠다.

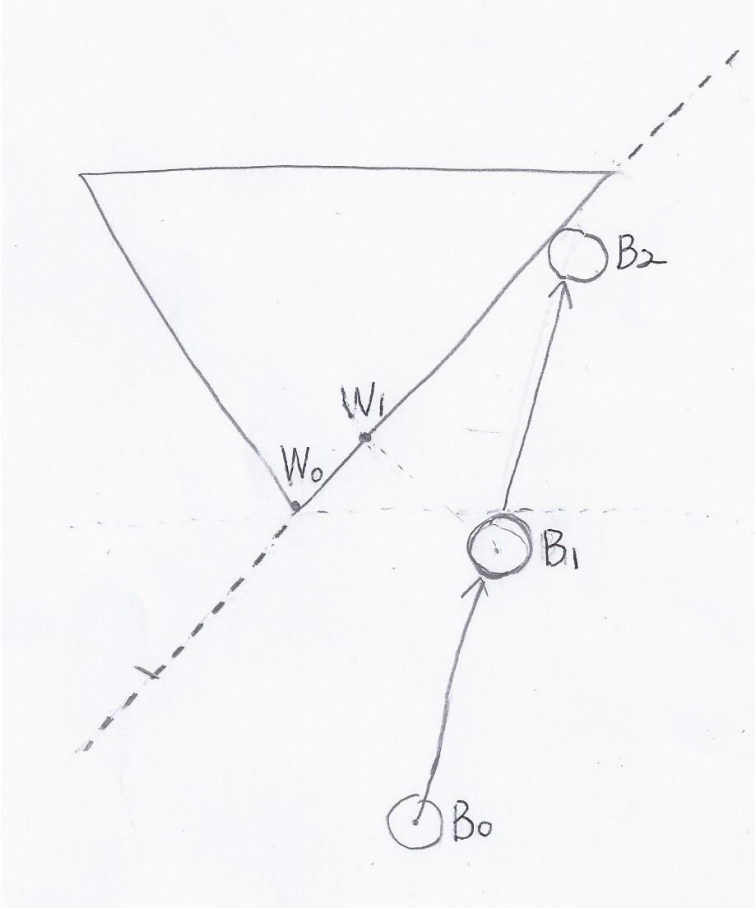
ALGORITHM

우선, 간단한 원과 삼각형의 충돌부터 생각해 보자.



그림에서 물체 A 는 삼각형이고 물체 B 는 원이다. μ_B 의 값을 원의 반지름으로 설정하면 물체 B 는 점으로 표현할 수 있다

물체 B 의 시작위치 B0 에서 가장 가까운 점은 W0 이다. 그래서, 점선으로 표현된 평면까지 물체 B 를 이동시키면 그 위치는 B1 이 된다.



이제 B1 에 가장 가까운 물체 A 의 위치는 W1 이다. 이번에도 점선으로 표현된 평면까지 이동하면 물체 B 의 위치는 B2 가 된다.

이 알고리즘을 두 개의 convex 한 물체의 충돌에도 적용할 수 있다.

다음은 물체 A 가 정지되어있고, 물체 B 가 velocity 의 속도로 움직일 때, 충돌하는 시각을 계산하는 알고리즘이다. 함수값이 0 이면 물체가 이미 충돌된 상태이기 때문에 velocity 방향으로 움직일 수 없는 것을 의미한다. 함수값이 1 이면 1 초 이내에는 충돌이 발생하지 않는 것을 의미한다.

구현방법은 Advanced Collision Detection.pdf 문서의 마지막 부분에 있는 **PenetrationDepth** 알고리즘과 거의 비슷하다.

```
float DynamicGJK(Polytope A, Polytope B, Vector velocity)
{
```

```
        Vector v = (임의의 길이가 1 인 벡터);
        float upper_bound = FLT_MAX;
        Set<Vector> W = ϕ;
        float t = 0;
        for ( ; ; ) {
                Vector w = SupportMapping(A,B,-v) – t * velocity;
                if ((v · w) > (μA + μB)) {
                        if ((v · w) – (μA + μB) > (1 – t) * (v · velocity))
                                return 1; // v is separating axis
                        t += ((v · w) – (μA + μB))/(v · velocity) ;
                        W = ϕ;
                }
                if ((1 – εrel) * upperbound ≤ (v · w))
                        return t; // v is contact axis
                v = υ(conv(W∪{w}));
                W = (Convex Hull 이 v 를 포함하는 가장 작은 W∪{w}의 부분집합);
                upper_bound = √(v · v) ;
                if (upper_bound <= (μA + μB))
                        return t; // v is contact axis
                v = v / upper_bound; // normalize
        }
}
```

if ((v · w) > (μA + μB)) 부분에서는 두 물체가 떨어져 있는 경우에 **t** 의 값을 증가시키고 **W** 의 값을 다시 초기화한다.

if ((1- ε_{rel}) * upper_bound ≤ v · w) 부분에서는 새로 구한 **w** 의 값이 이전 값에 비해 더 이상 가까워지지 않은 경우에 반복을 중단한다.

if (upper_bound <= (μA + μB)) 부분에서는 새로 구한 **v** 의 크기가 충분히 작으면 반복을 중단한다.

함수 값이 1 미만일 때, **t * velocity + (upper_bound+μB)* v** 로 충돌 점을 구할 수 있다.