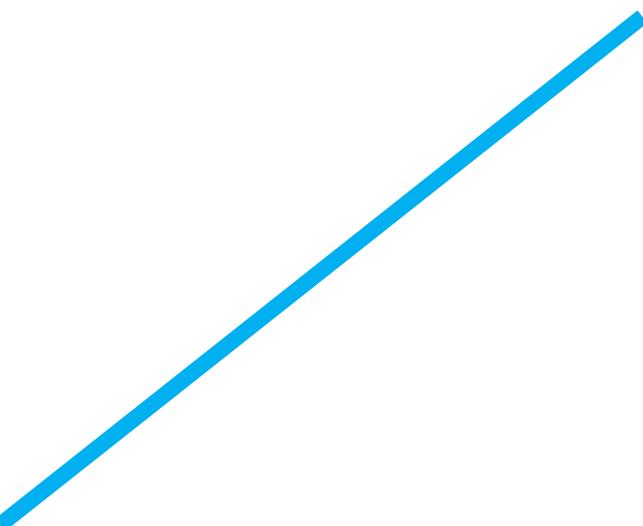


6. Autonomous Agents



박종화
suakii@gmail.com

What is autonomous agents?

자신을 둘러싼 환경에 따라
'스스로'

판단을 내리고 동작하는 것



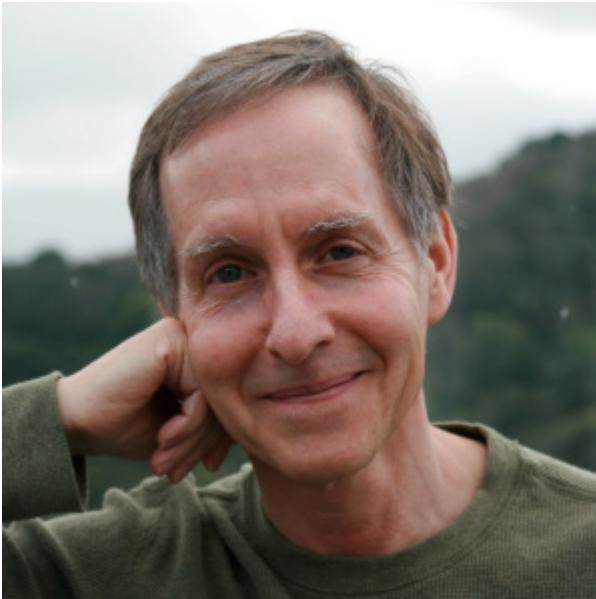
What is autonomous agents?

1. 제한적인
환경 인식 능력

2. 그 환경에서
얻은 정보를 바탕으
로 동작을 연산

3. 리더는
없다

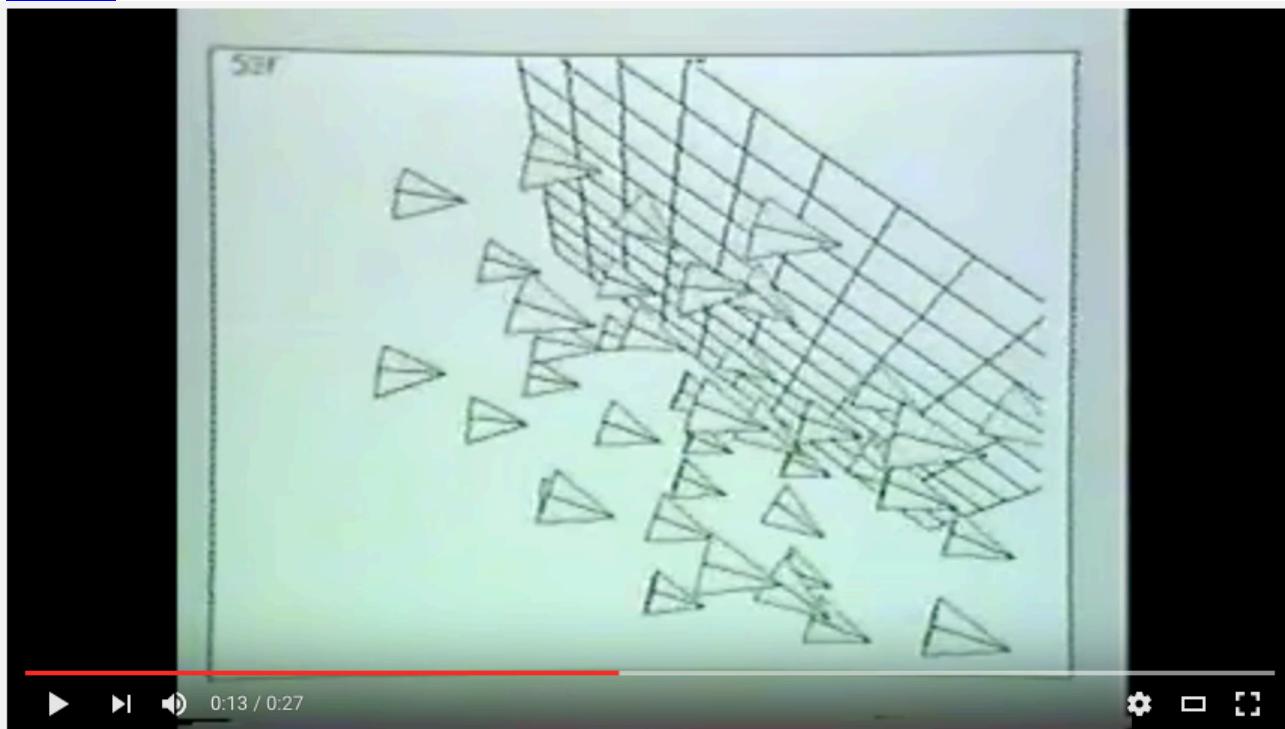
History



- Craig Reynolds
 - Reynolds, Craig W. (1987). "Flocks, herds, and schools: A distributed behavioral model"
 - <http://www.cs.toronto.edu/~dt/siggraph97-course/cwr87/>

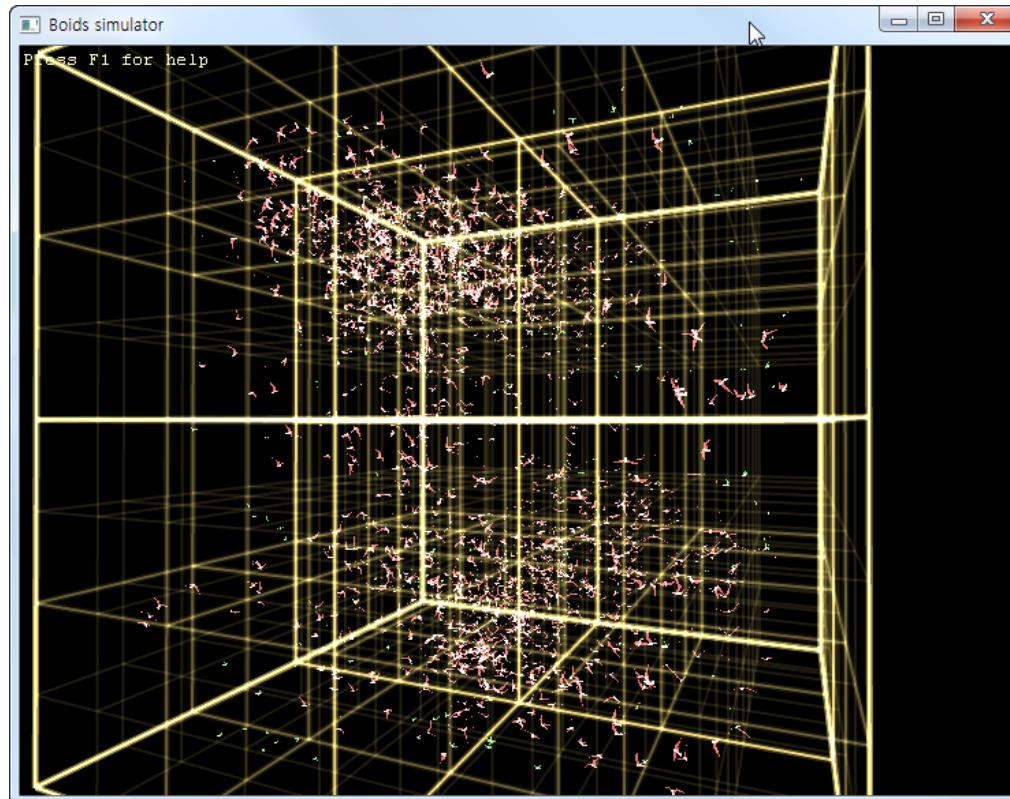
History

- Original 1986 Boids simulation
 - <https://www.youtube.com/watch?v=86iQiV3-3IA>



Boids Simulator

- <http://www.decarpentier.nl/boids>



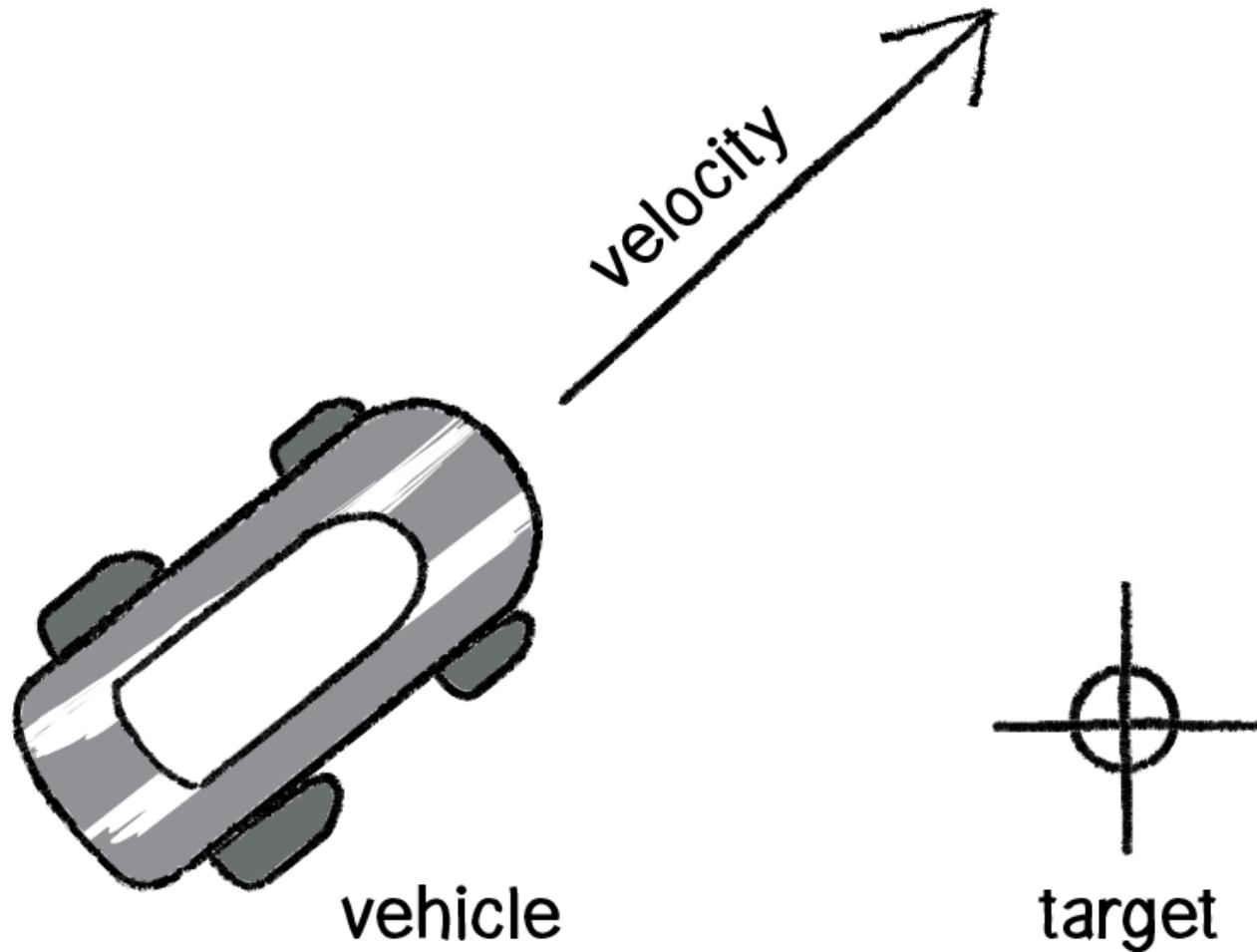
Vehicles and Steering

- class Vehicle {
 - PVector location;
 - PVector velocity;
 - PVector acceleration;
 -
 - }
- <http://www.red3d.com/cwr/steer/gdc99/>

Vehicles and Steering

- Action Selection(행동 선택)
 - 1개 또는 여러 개의 차량은 목적을 가지고, 그 목적을 기반으로 1개 또는 여러 개의 행동을 선택한다.
- Steering(조향)
 - 움직임과 관련된 연산(조향력)
- Locomotion(이동운동)

Steering Force

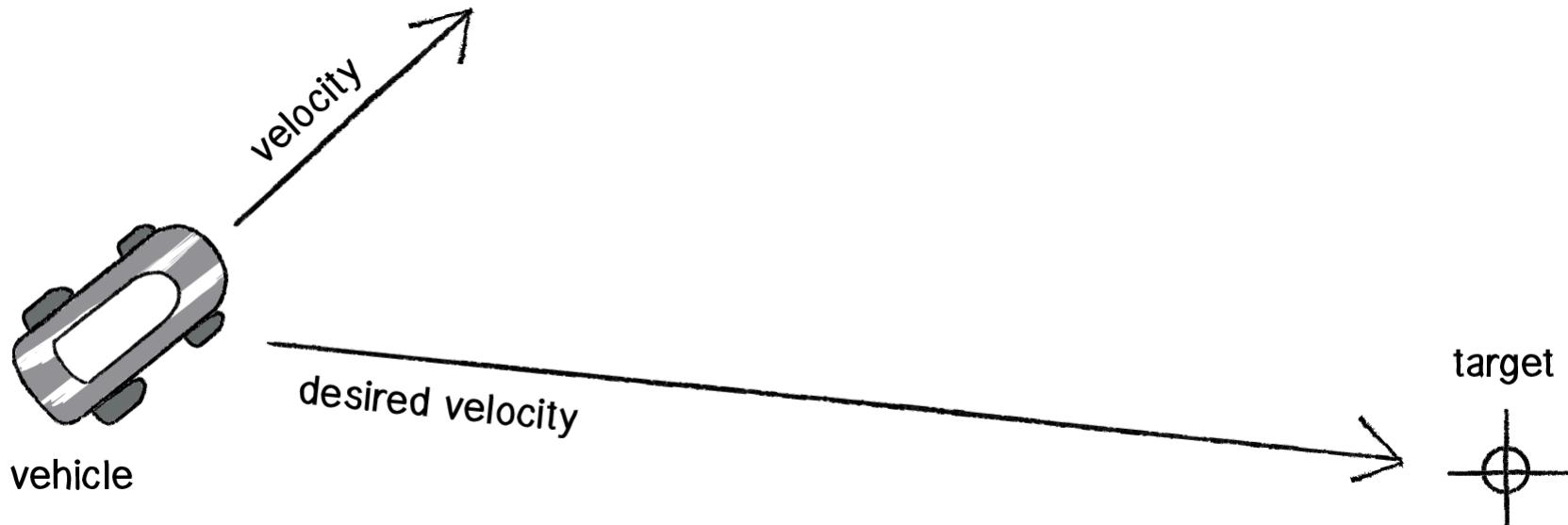


Steering Force

- 기존
 - 목표를 Attractor 객체로 만듬
 - 중력을 적용해 차를 끌어 당긴다.
- 이번 장
 - 차량 스스로 상태와 환경을 인식
 - 목표를 향해 조향
 - 차량 스스로가 어떻게 이동하고 싶은지 인식하고 해당 목표와 현재 이동 속도를 비교해 힘을 적용하자.

Steering Force

- **Steering force = desired velocity – current velocity**
- PVector steer = PVector.sub(desired, velocity)



- PVector desired = PVector.sub(target, location)

What's the problem?

- What if we have a very high-resolution window and the target is thousands of pixels away?

2.4. 순간이동(Teleport)

[편집]

| 순간이동 | 소환사 레벨 | 지원모드 | 재사용 대기시간 | 설명 |
|------|--------|------|----------|---|
| | 6 | 클래식 | 300초 | 4초 뒤 챔피언이 지정한 아군 미니언, 포탑 혹은 와드로 순간 이동합니다. |

- The vehicle desires to move towards the target at **maximum speed**.

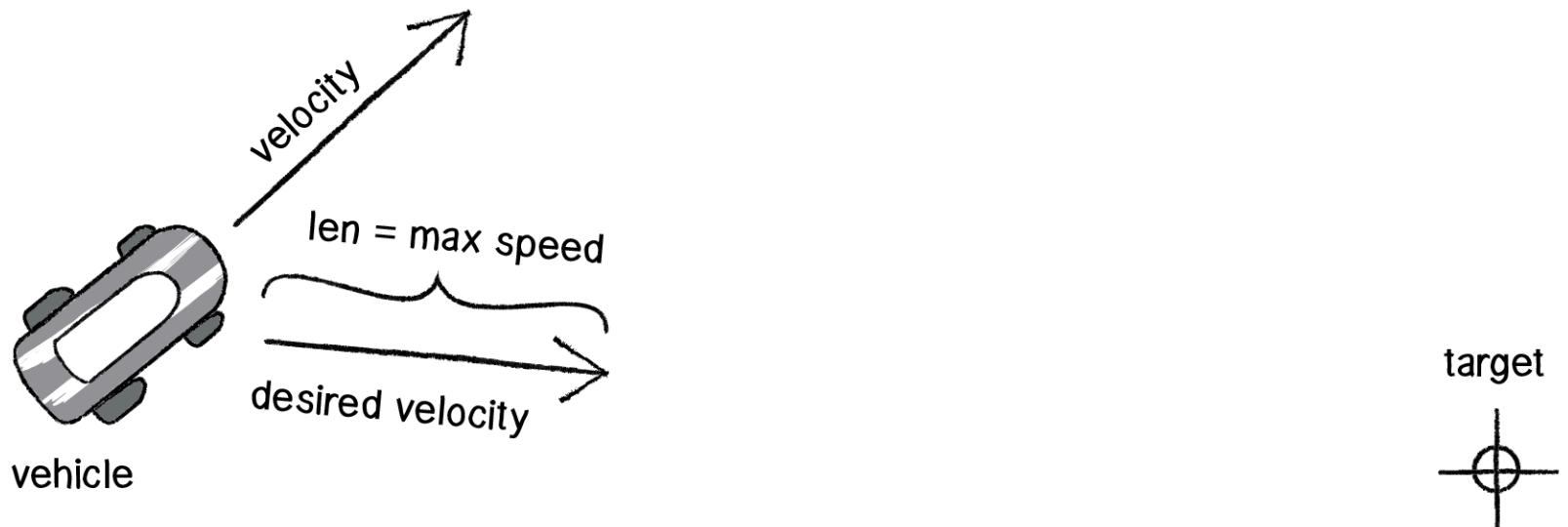
Steering Force

```
class Vehicle {  
    PVector location;  
    PVector velocity;  
    PVector acceleration;  
    float maxspeed;}  
  
PVector desired = PVector.sub(target,location);  
desired.normalize();  
desired.mult(maxspeed);
```

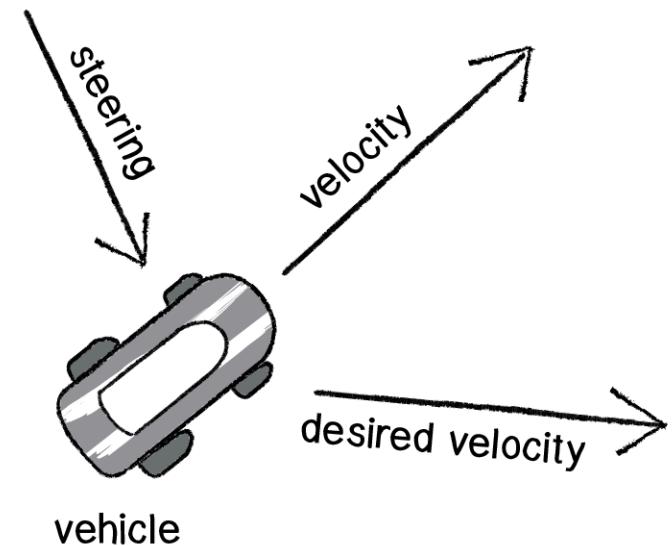
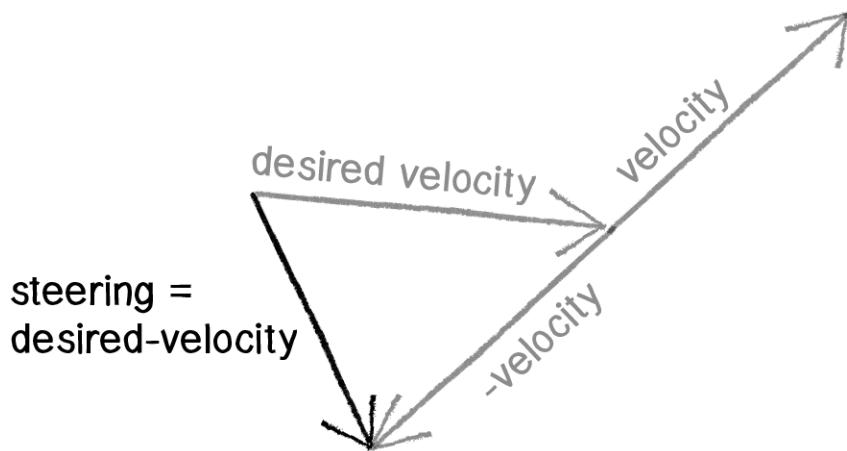
Steering Force

```
void seek(PVector target) {  
    PVector desired = PVector.sub(target,location);  
    desired.normalize();  
    desired.mult(maxspeed);  
  
    //Reynolds's formula for steering force  
    PVector steer = PVector.sub(desired,velocity);  
    applyForce(steer);  
}
```

Steering Force



Steering Force



Steering Force

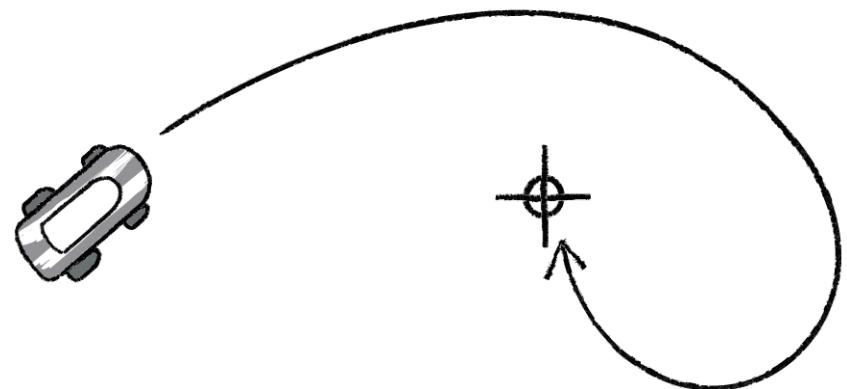


Steering Force

path with high max force



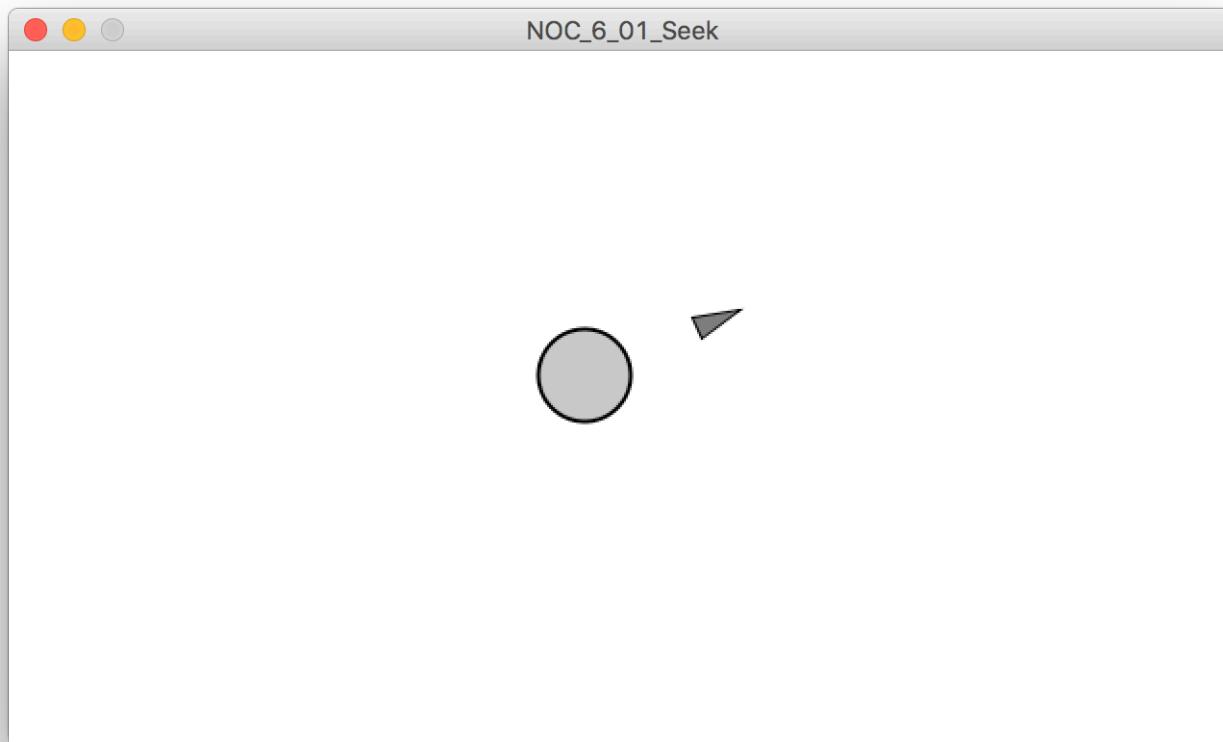
path with low max force



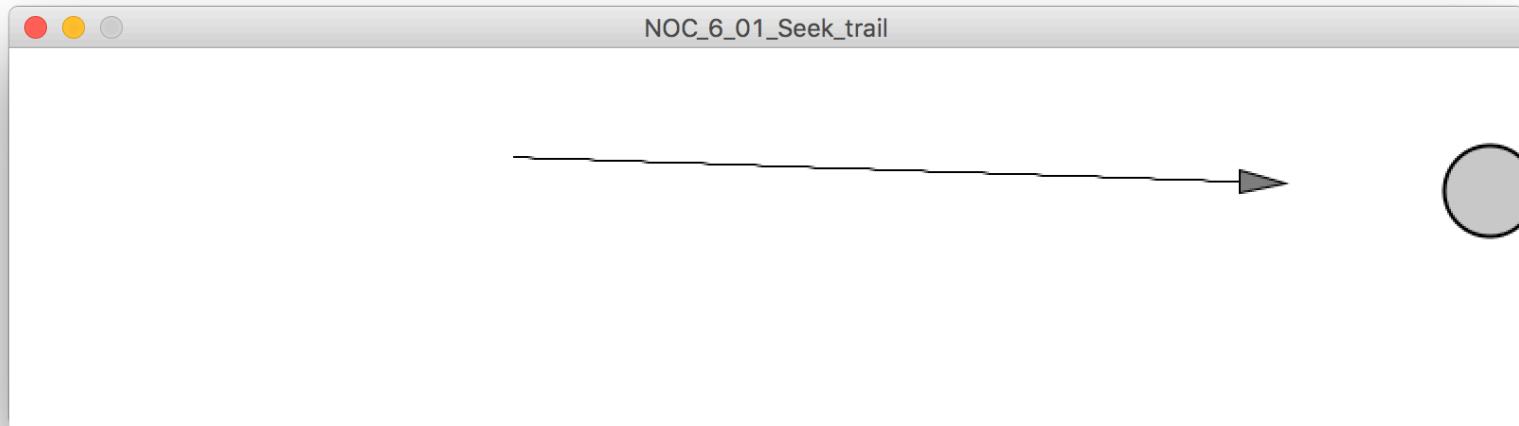
Steering Force

```
class Vehicle {  
    PVector location;  
    PVector velocity;  
    PVector acceleration;  
    float maxspeed;  
    float maxforce;  
}  
  
steer.limit(maxforce);  
applyForce(steer);
```

Example



Example



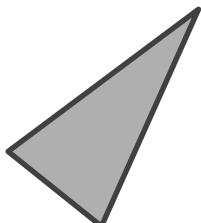
Arriving Behavior

문제 발견: Vehicle의 단진동

Why?

최대한 빨리 목적지에 간다!
최대한 빨리 목적지에 간다!
최대한 빨리 목적지에 간다!
최대한 빨리 목적지에 간다!

...



Vehicle의
생각

Arriving Behavior



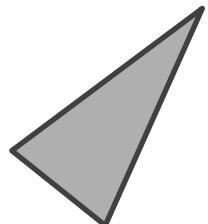
Arriving Behavior

바꿔야 한다.

How?

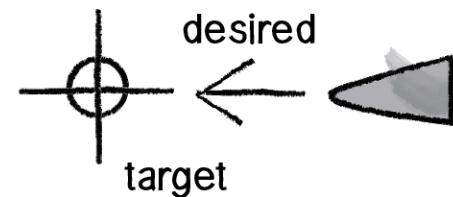
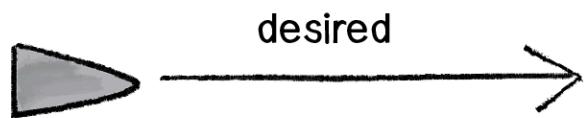
아주 멀리 있다. 최대한 빨리 목적지에 간다!
멀리 있다. 최대한 빨리 목적지에 간다!

가까이 있다. 좀 느리게 목적지에 간다!
...
다 왔다. 멈춘다!

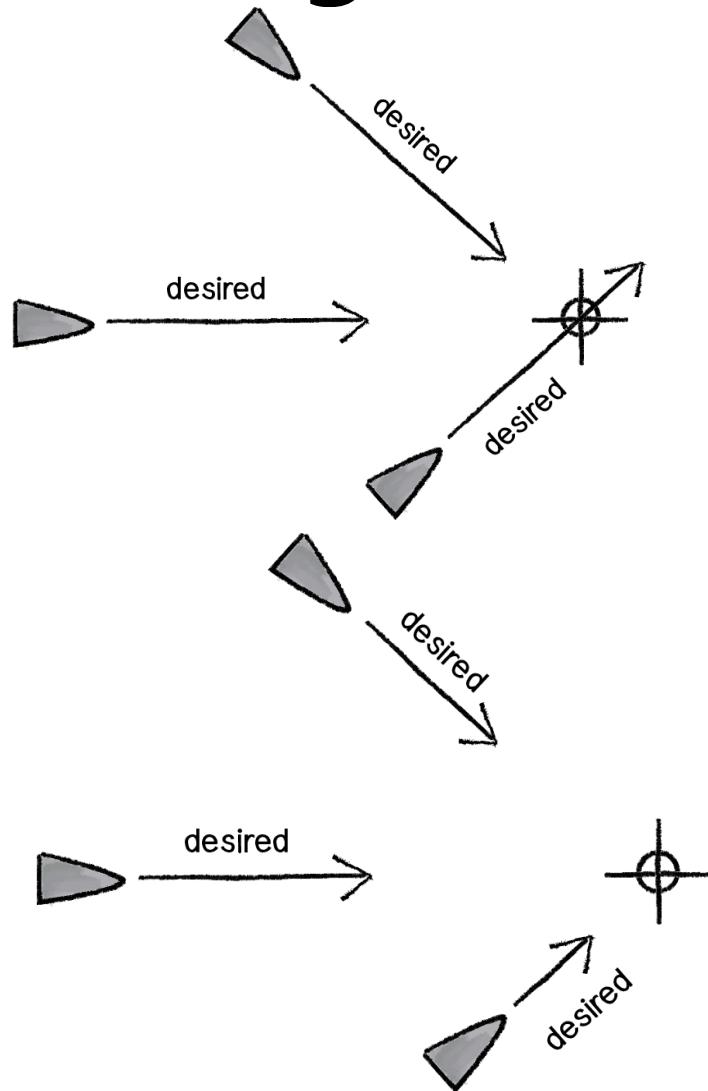


Vehicle의
생각

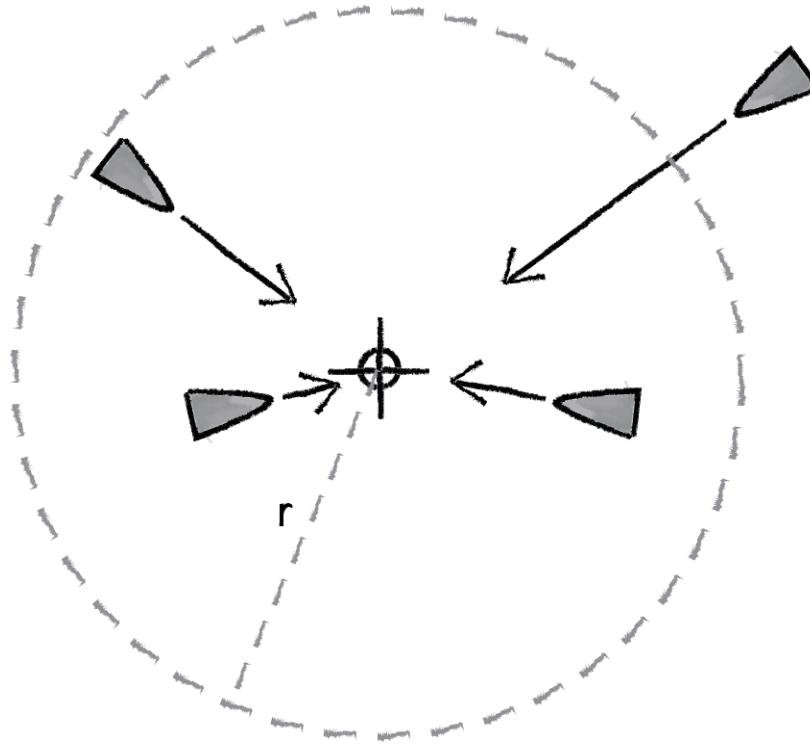
Arriving Behavior



Arriving Behavior



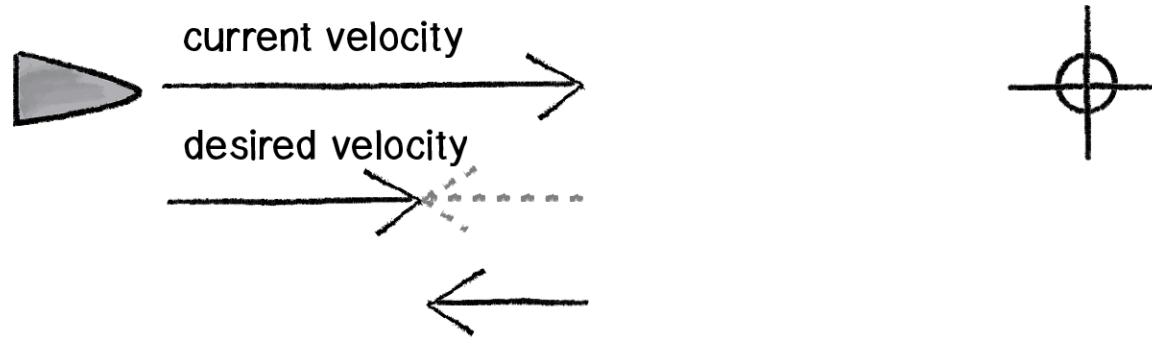
Arriving Behavior



Arriving Behavior

```
void arrive(PVector target) {  
    PVector desired = PVector.sub(target, location);  
    float d = desired.mag();  
    desired.normalize();  
  
    if (d < 100) {  
        float m = map(d, 0, 100, 0, maxspeed);  
        desired.mult(m);  
    } else {  
        desired.mult(maxspeed);  
    }  
    PVector steer = PVector.sub(desired, velocity); steer.limit(maxforce);  
    applyForce(steer);  
}
```

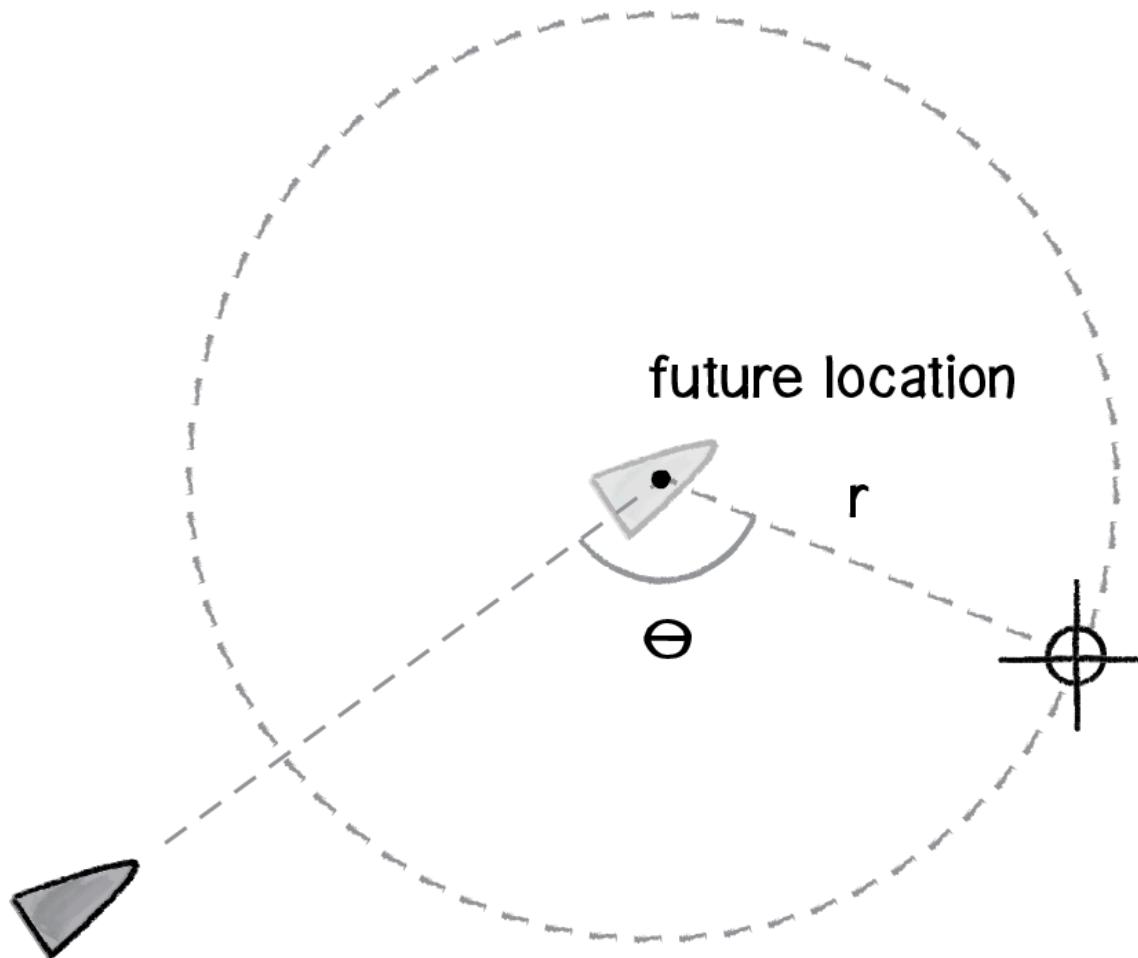
Arriving Behavior



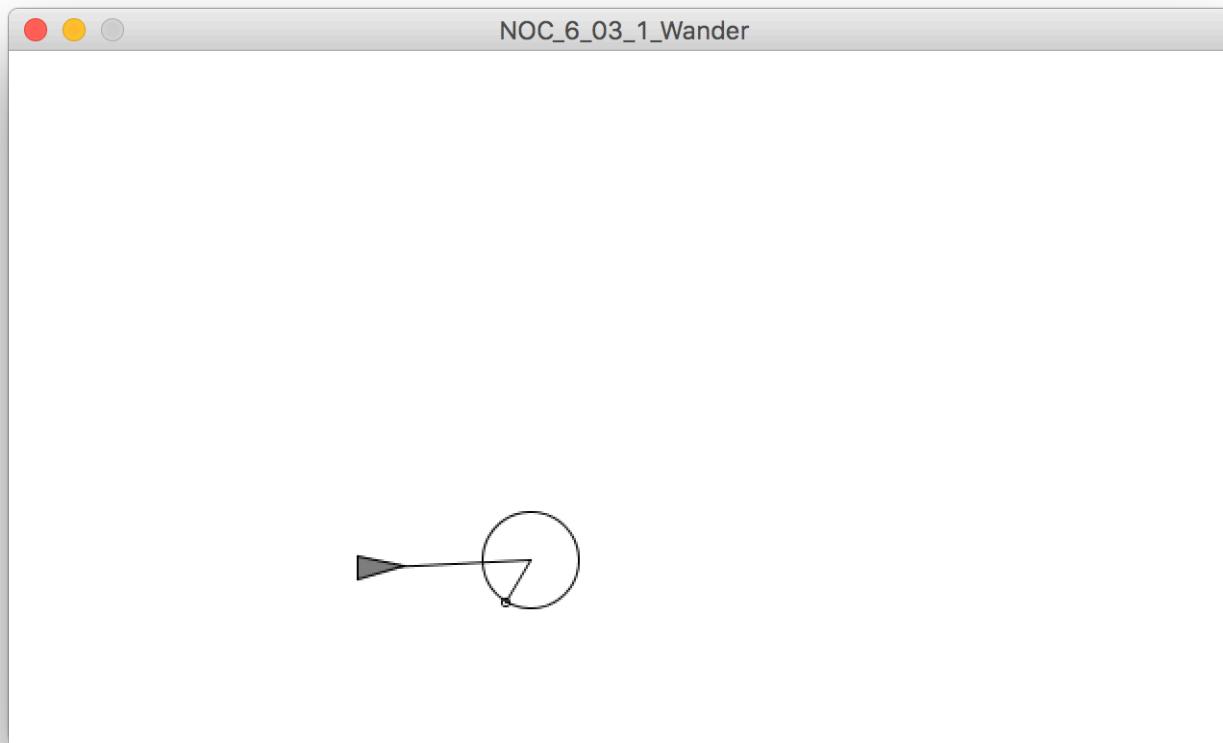
Desired Velocity

- “배회는 어느 정도 긴 시간 동안 임의 상태를 가지는 조향으로, 특정한 시점에서의 조향 방향은 다음 조향 방향과 관계를 가진다. 간단하게 각 시점에서의 임의 조향 방향을 생성하기만 해도 흥미롭고 재미있는 움직임이 만들어진다.”
- —Craig Reynolds

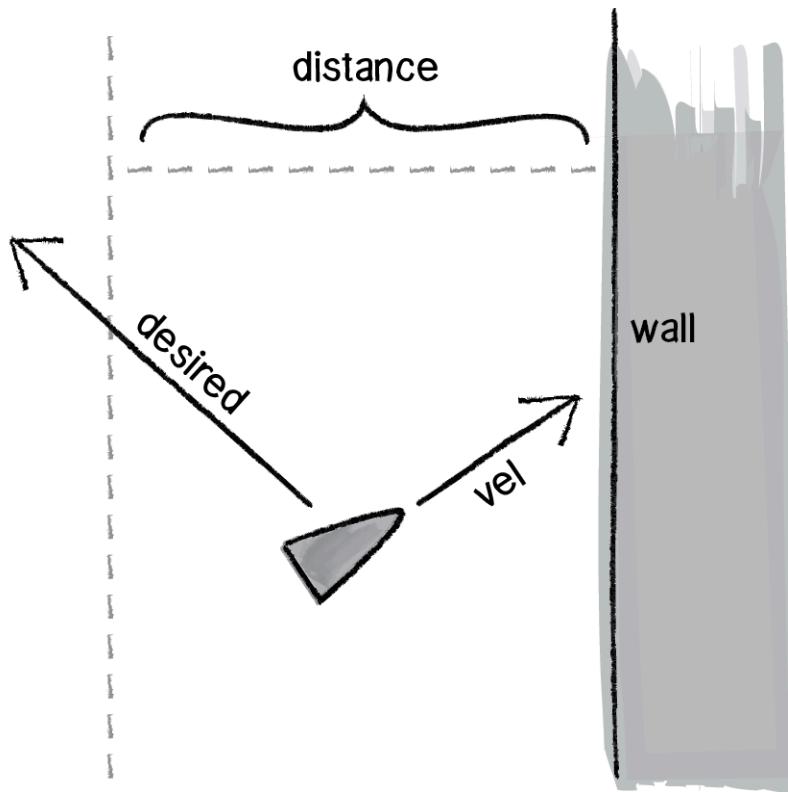
Desired Velocity



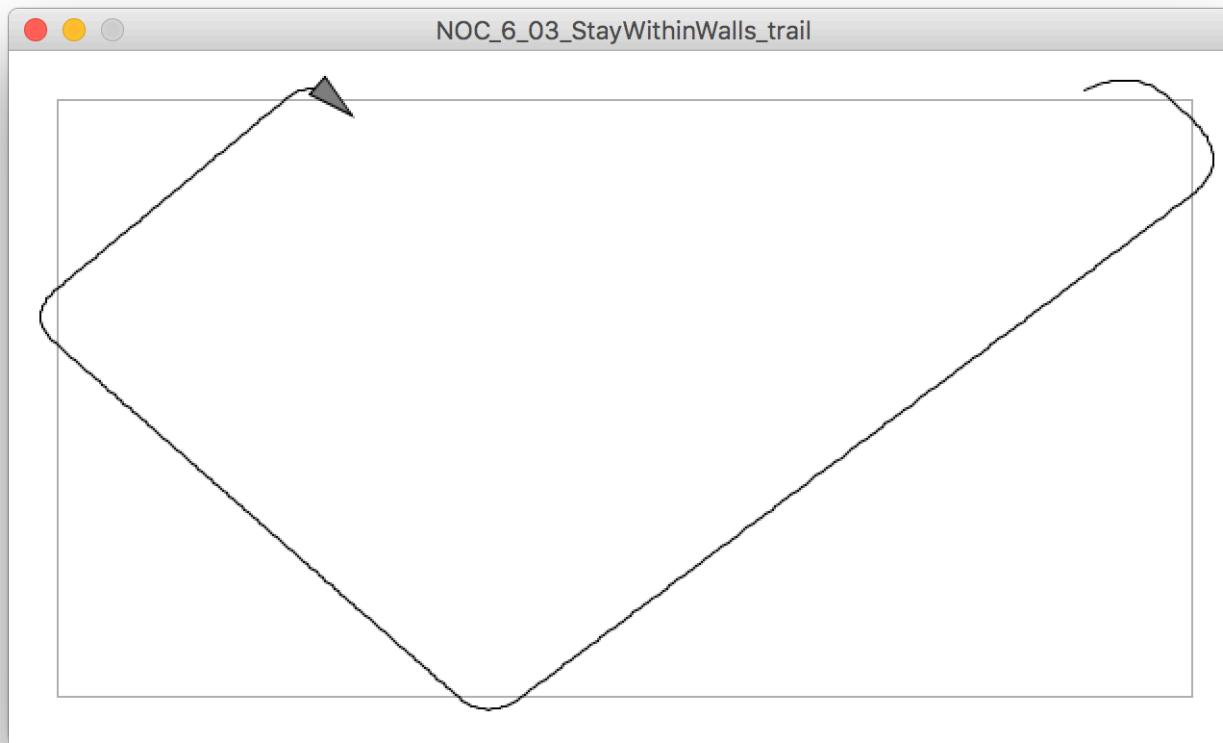
Desired Velocity



Desired Velocity



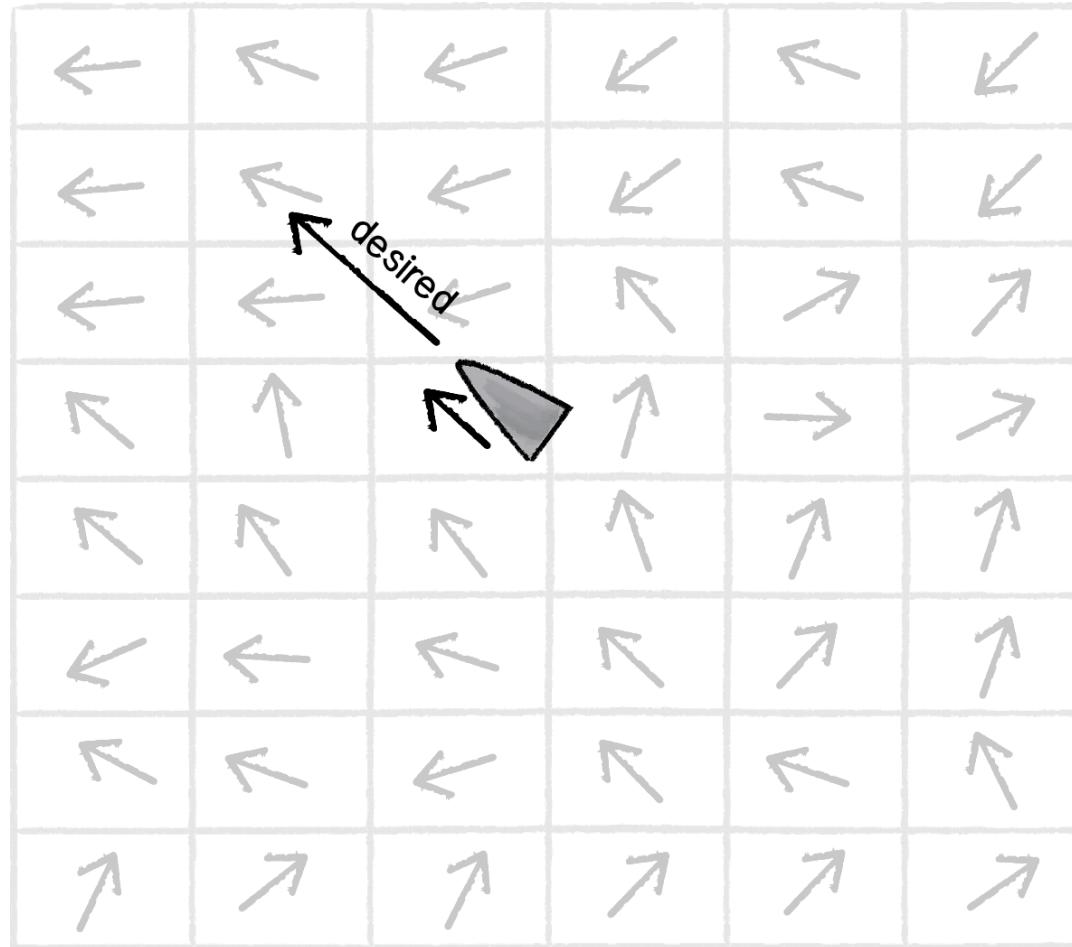
Desired Velocity



Desired Velocity

```
if (location. x < 25) {  
    PVector desired = new PVector(maxspeed, velocity.y);  
    PVector steer = PVector.sub(desired, velocity);  
    steer.limit(maxforce);  
    applyForce(steer);  
}
```

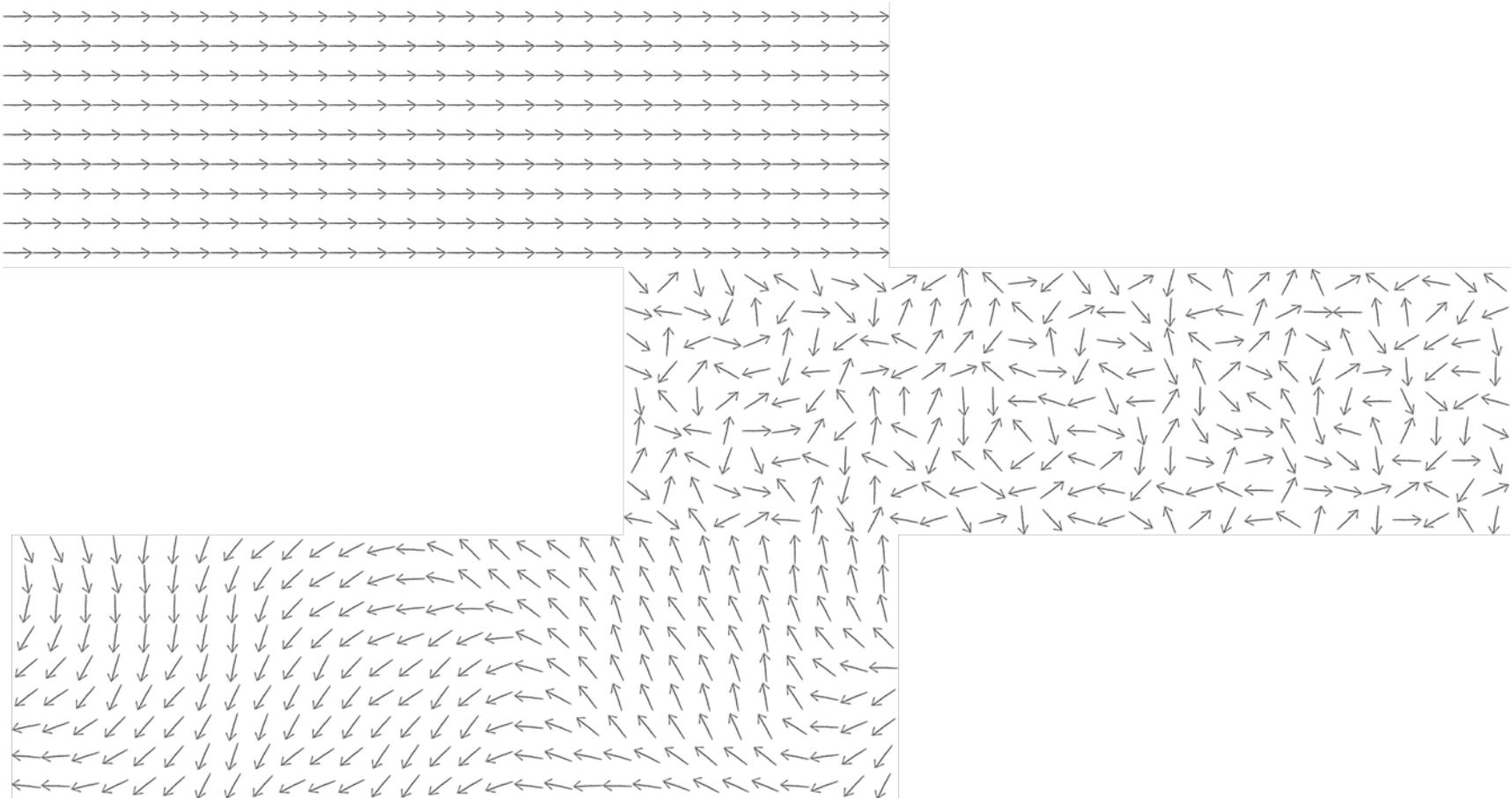
Flow Fields



Flow Fields

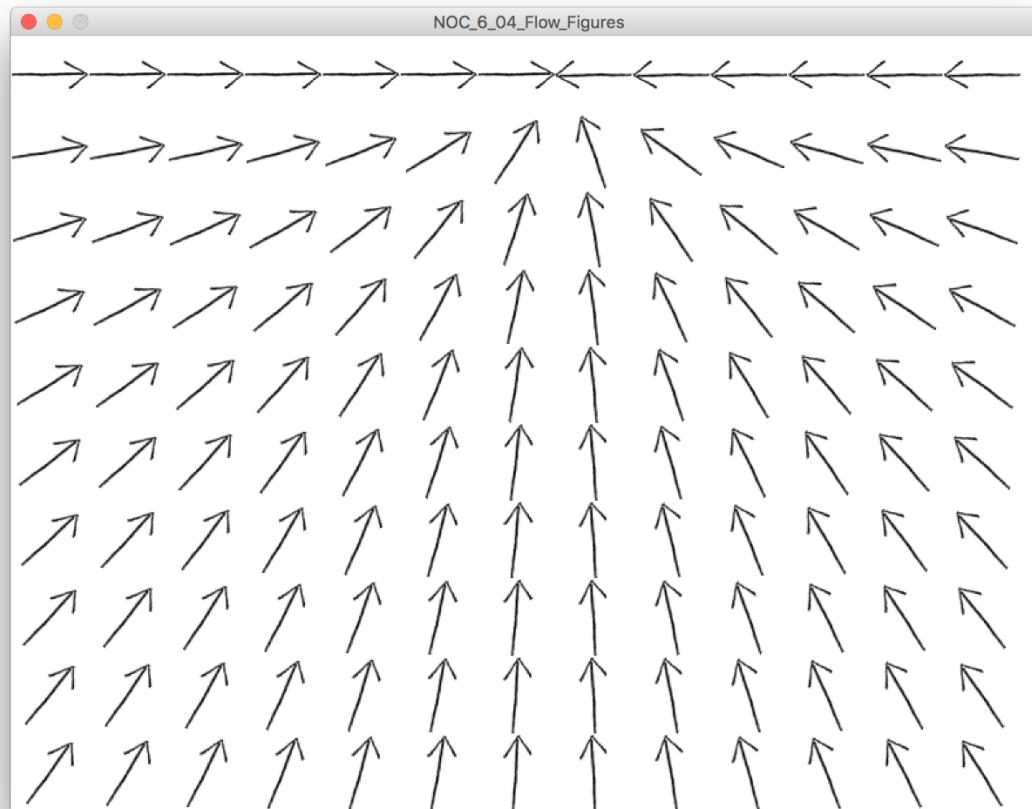
```
class FlowField {  
  
    PVector[][] field;  
  
    int cols, rows;  
  
    int resolution;  
  
    FlowField() {  
        resolution = 10;  
        cols = width/resolution;  
        rows = height/resolution;  
        field = new PVector[cols][rows];  
    }  
}
```

Flow Fields



Flow Fields

- ex-FlowFigures



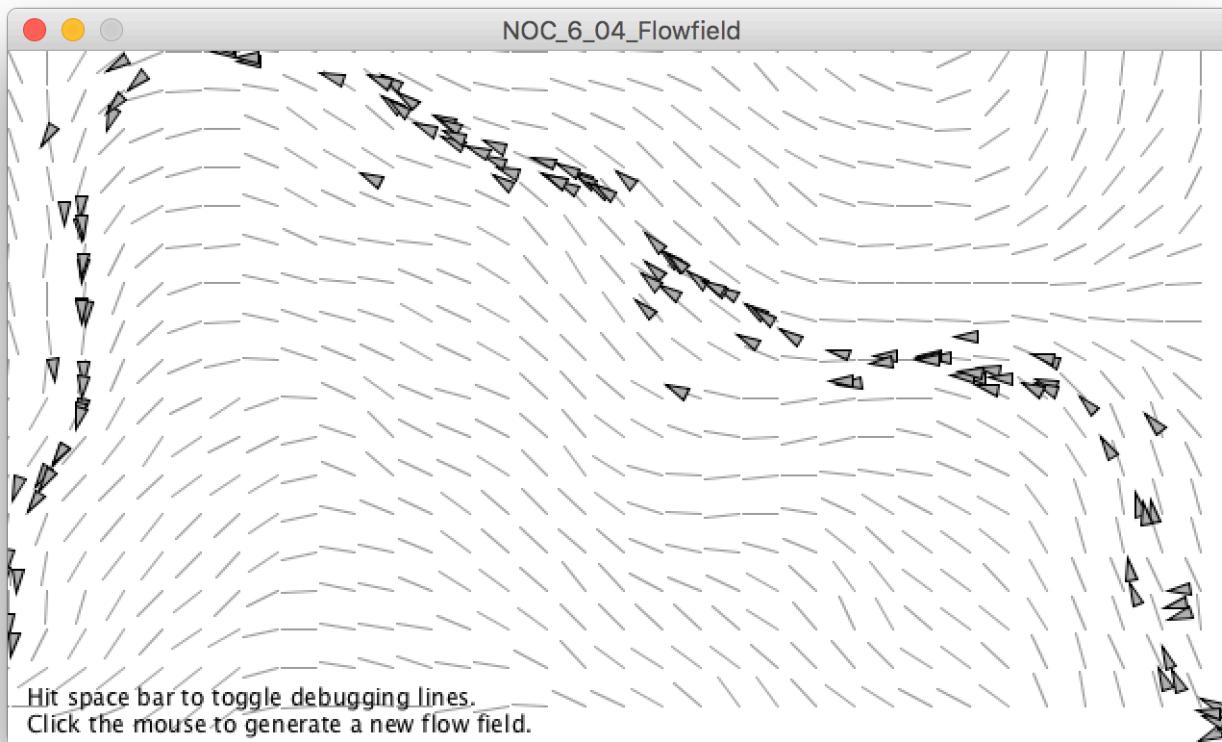
Flow Fields

```
PVector lookup(PVector lookup) {  
  
    int column = int(constrain(lookup.x/resolution,0,cols-1));  
    int row = int(constrain(lookup.y/resolution,0,rows-1));  
  
    return field[column][row].get();  
}
```

```
class Vehicle {  
  
    void follow(FlowField flow) {  
  
        PVector desired = flow.lookup(location);  
        desired.mult(maxspeed);  
  
        PVector steer = PVector.sub(desired, velocity);  
        steer.limit(maxforce);  
        applyForce(steer);  
    }  
}
```

```
class FlowField {  
  
    PVector[][] field;  
    int cols, rows;  
    int resolution;  
  
    FlowField(int r) {  
        resolution = r;  
        cols = width/resolution;  
        rows = height/resolution;  
        field = new PVector[cols][rows];  
        init();  
    }  
  
    void init() {  
        float xoff = 0;  
        for (int i = 0; i < cols; i++) {  
            float yoff = 0;  
            for (int j = 0; j < rows; j++) {  
                float theta = map(noise(xoff,yoff),0,1,0,TWO_PI);  
                field[i][j] = new PVector(cos(theta),sin(theta));  
  
                yoff += 0.1;  
            }  
            xoff += 0.1;  
        }  
    }  
  
    PVector lookup(PVector lookup) {  
  
        int column = int(constrain(lookup.x/resolution,0,cols-1));  
        int row = int(constrain(lookup.y/resolution,0,rows-1));  
        return field[column][row].get();  
    }  
}
```

Flow Fields



The Dot Product

$$\begin{aligned}\vec{A} &= (a_x, a_y) \\ \vec{B} &= (b_x, b_y)\end{aligned}$$



$$\vec{A} \cdot \vec{B} = a_x * b_x + a_y * b_y$$

```
PVector a = new PVector(-3,5);
PVector b = new PVector(10,1);

float n = a.dot(b);
```

```
public float dot(PVector v) {
    return x*v.x + y*v.y + z*v.z;
}
```



Dot Product

$$\begin{aligned}\vec{A} \cdot \vec{B} &= \| \vec{A} \| * \| \vec{B} \| * \cos(\theta) \\ \vec{A} \cdot \vec{B} &= a_x * b_x + a_y * b_y\end{aligned}$$



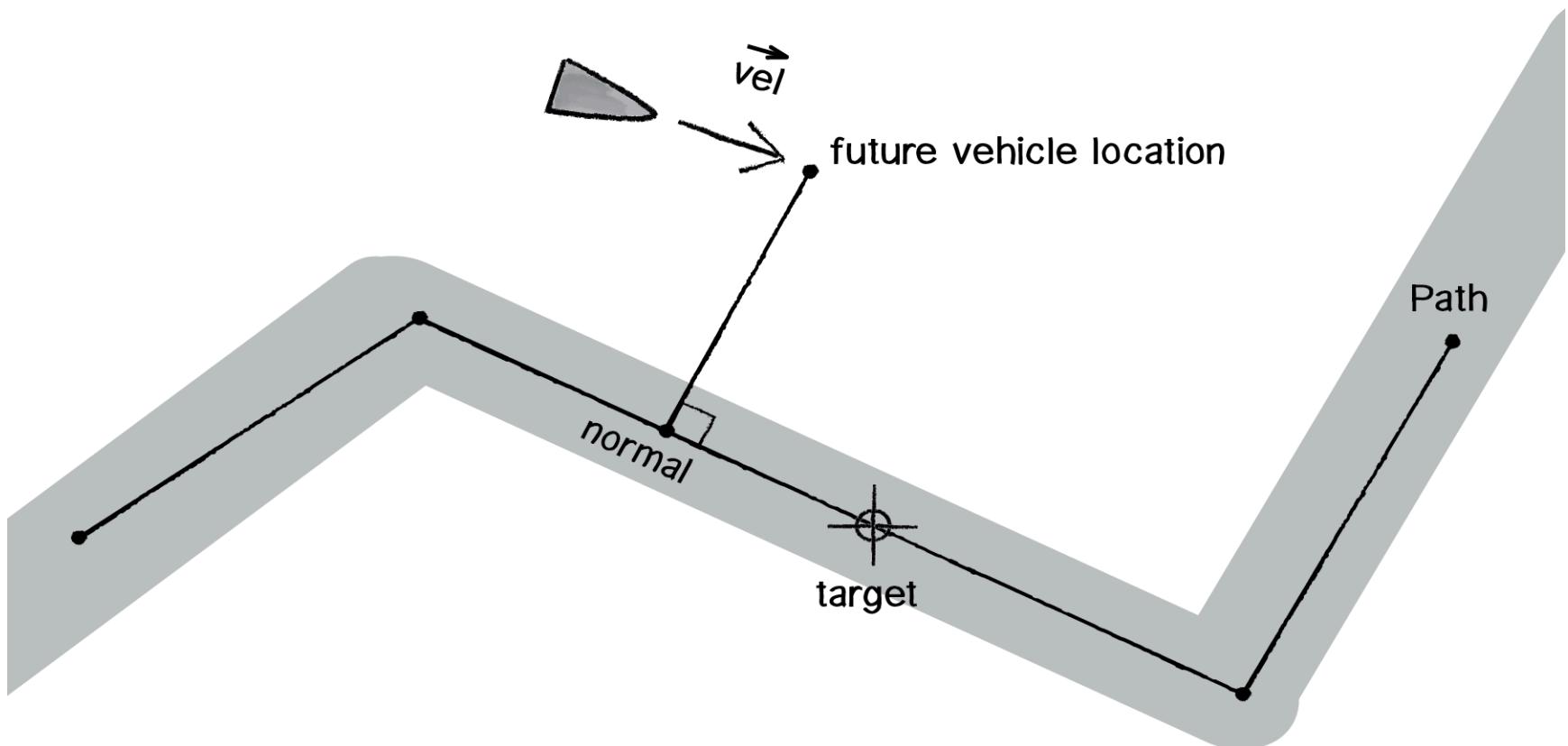
$$\begin{aligned}a_x * b_x + a_y * b_y &= \| \vec{A} \| * \| \vec{B} \| * \cos(\theta) \\ \theta &= \cos^{-1} ((\vec{A} \cdot \vec{B}) / (\| \vec{A} \| * \| \vec{B} \|))\end{aligned}$$

Dot Product

```
PVector a = new PVector(10,2);
PVector b = new PVector(4,-3);
float theta = acos(a.dot(b) / (a.mag() * b.mag()));
```

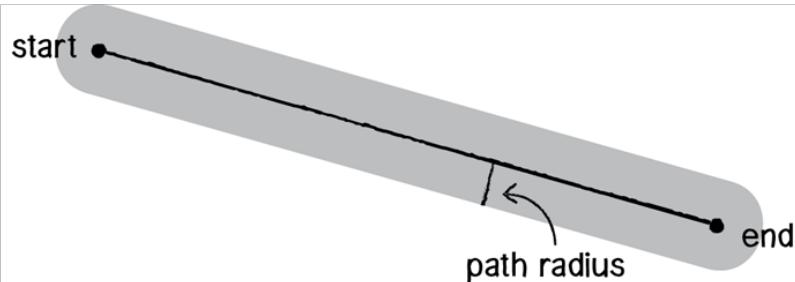
```
static public float angleBetween(PVector v1, PVector v2) {
    float dot = v1.dot(v2);
    float theta = (float) Math.acos(dot / (v1.mag() * v2.mag()));
    return theta;
}
```

Path Following not Path Finding



Path Following

직선 주위의 radius

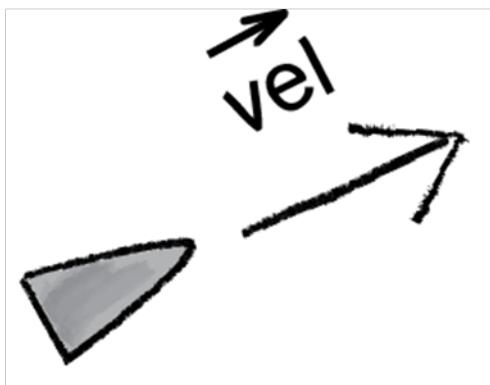


작은 radius는
Vehicle이
Path에서 더욱 가깝게 움직임을
뜻함

```
class Path {  
    PVector start;  
    PVector end;  
  
    float radius;  
  
    Path() {  
        radius = 20;  
        start = new PVector(0,height/3);  
        end = new PVector(width,2*height/3);  
    }  
  
    void display() { // Display the path.  
        strokeWeight(radius*2);  
        stroke(0,100);  
        line(start.x,start.y,end.x,end.y);  
        strokeWeight(1);  
        stroke(0);  
        line(start.x,start.y,end.x,end.y);  
    }  
}
```

Path Following

우리의 주인공을 정의하자.



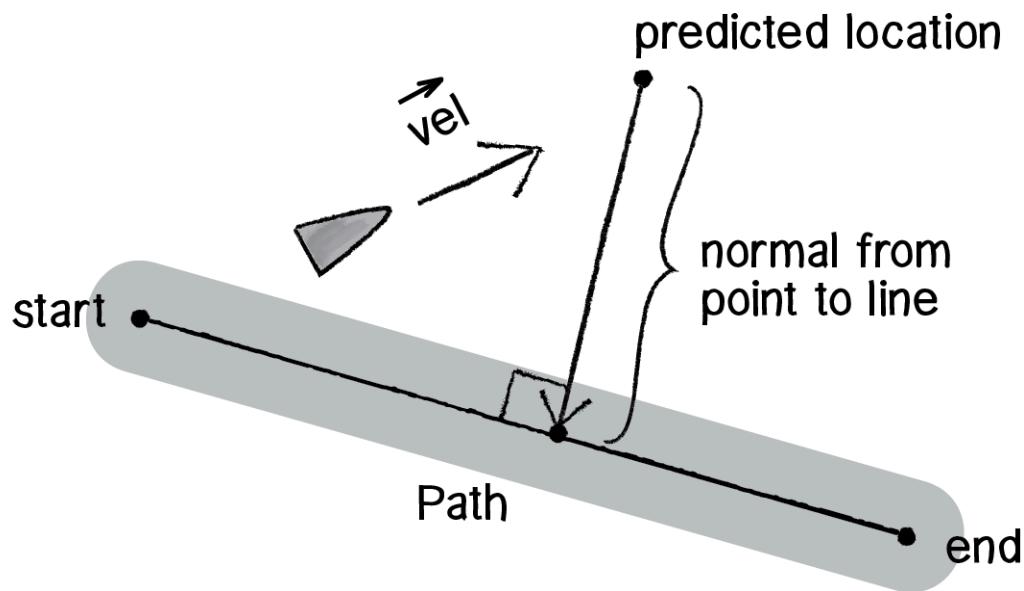
```
PVector predict = vel.get();
```

```
predict.normalize();  
predict.mult(25);
```

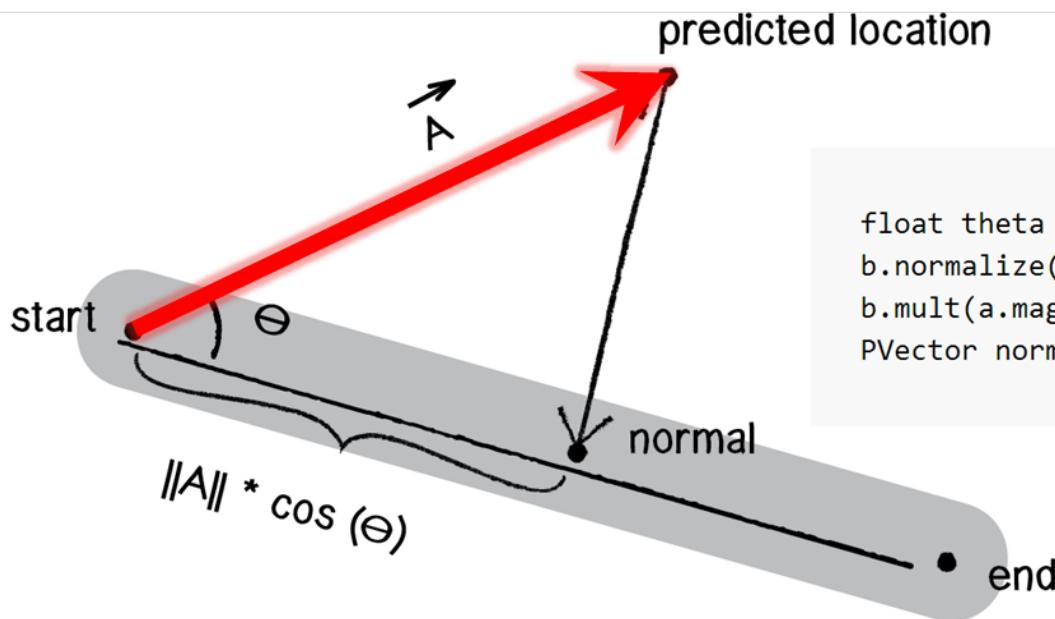
```
PVector predictLoc = PVector.add(loc, predict);
```

이 녀석이 라인을 따라갈 것이다.

Path Following



Path Following



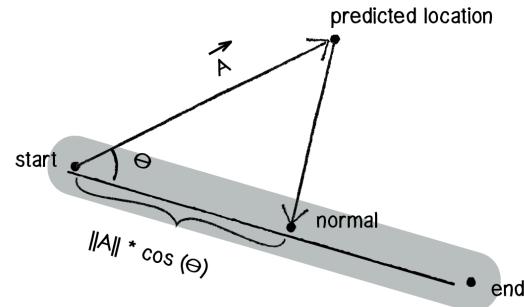
```
float theta = PVector.angleBetween(a,b);
b.normalize();
b.mult(a.mag()*cos(theta));
PVector normalPoint = PVector.add(path.start,b);
```

Path Following

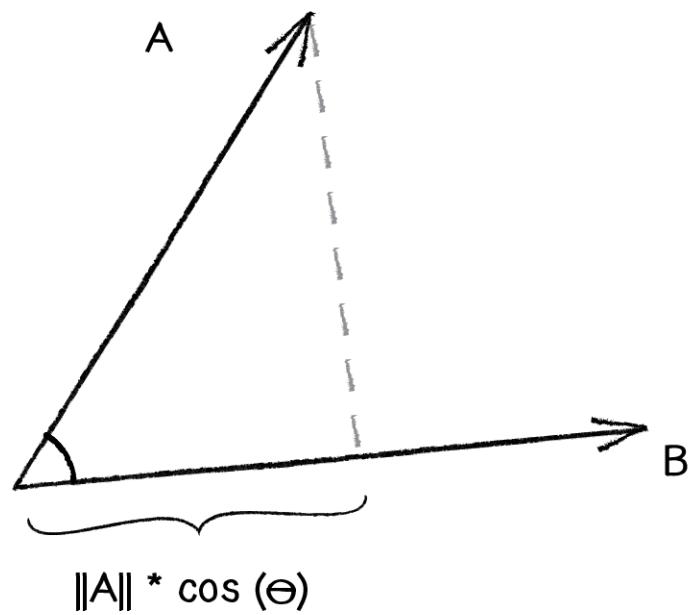
$\vec{A} \cdot \vec{B} = \| \vec{A} \| * \| \vec{B} \| * \cos(\theta)$ 에서 B 가 단위벡터이므로

$\vec{A} \cdot \vec{B} = \| \vec{A} \| * \cos(\theta)$ 라고 고쳐 쓸 수 있다.

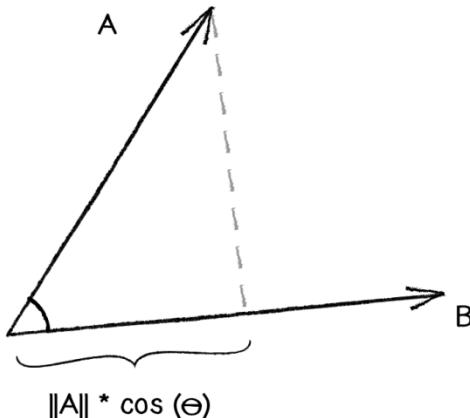
```
float theta = PVector.angleBetween(a, b);  
b.normalize();  
  
b.mult(a.dot(b));  
  
PVector normalPoint = PVector.add(path.start, b);
```



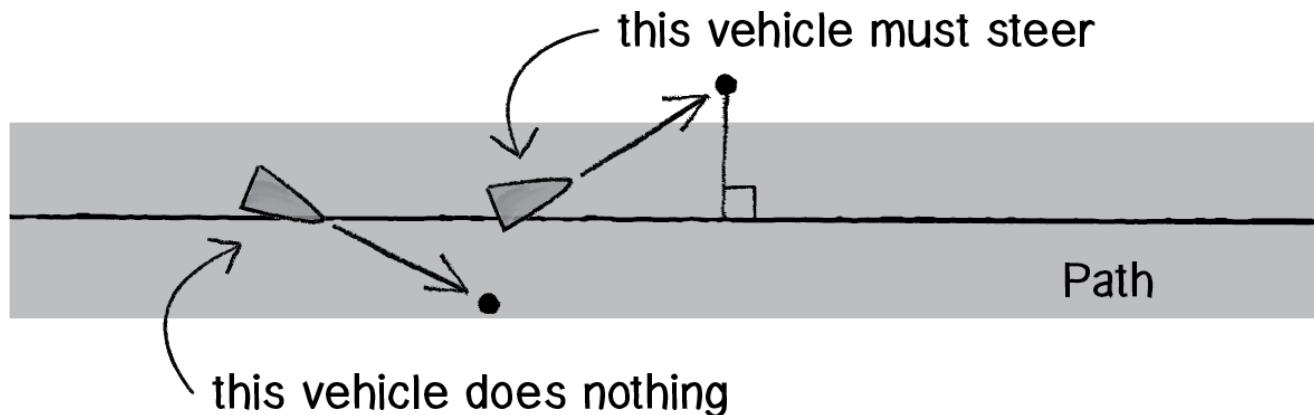
Path Following



Path Following



Vehicle과 Path 사이의 거리를 판단
Steer를 작용할 때를 결정할 수 있다.



Path Following

```
float distance = PVector.dist(predictLoc, normalPoint);
if (distance > path.radius) {
    b.normalize();
    b.mult(25);

    PVector target = PVector.add(normalPoint, b);

    seek(target);
}
```

이때 target은
normalpoint에서
조금 앞에 두어,
Vehicle이 계속
path를 따라
전진할 수
있도록 한다.

Distance가 path에서
radius보다 멀리
떨어져 있을 경우
Steer가 작용된다.

