DCC192

2025/2

UFMG

Desenvolvimento de Jogos Digitais

A22: IA — Comportamentos de Navegação (*Steering Behaviours*)

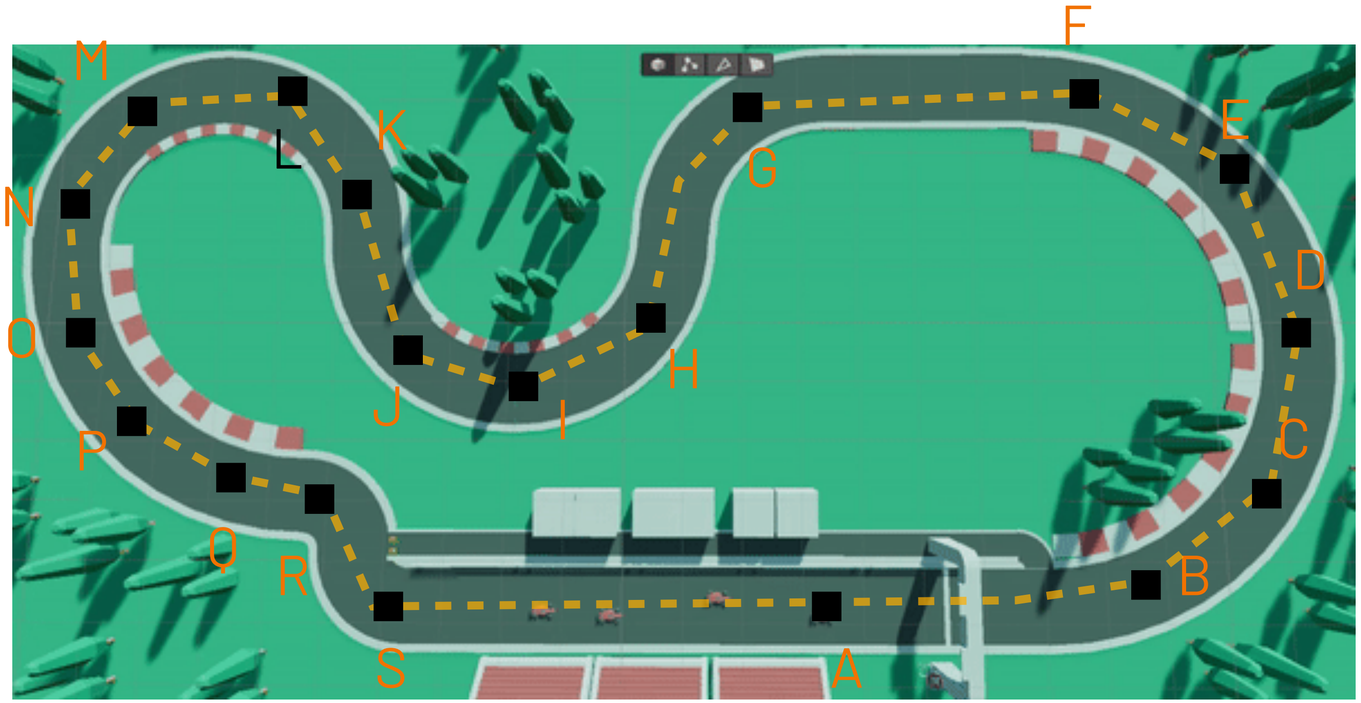
Prof. Lucas N. Ferreira

Plano de aula

- Força de direção (steering force)
- Comportamentos de navegação (steering behaviors)
  - Procurar
  - Fugir
  - Passear
  - Seguir um caminho

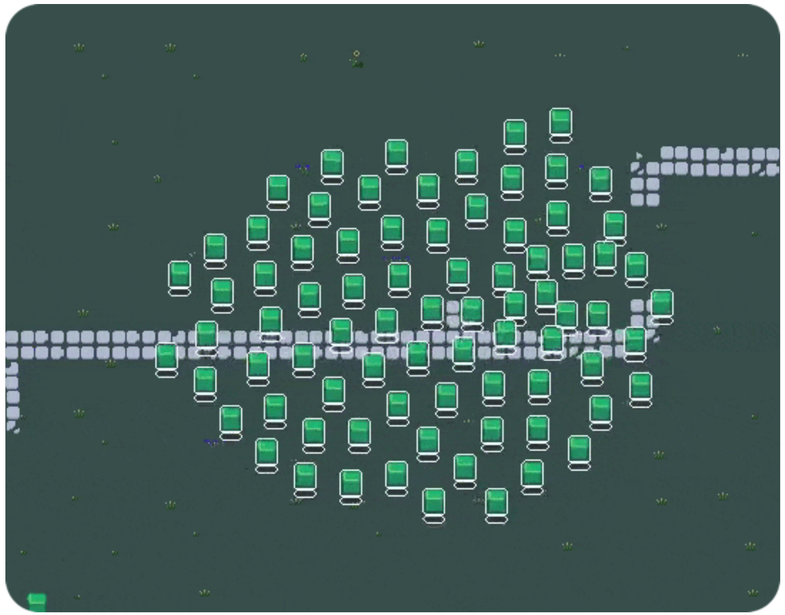
Motivação

Na última aula vimos como encontrar um caminho entre dois pontos A e B, mas como executar o movimento para seguir esse caminho de maneira natural?



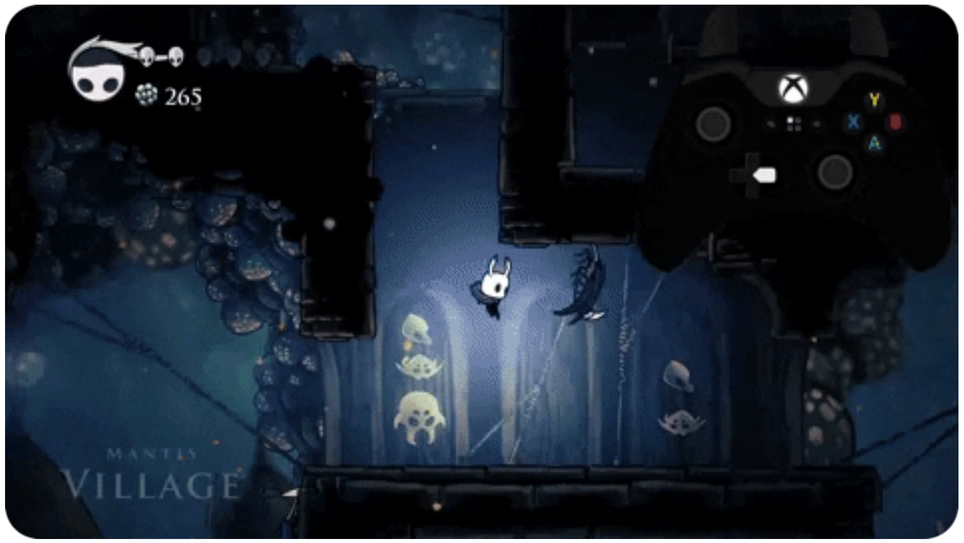
- Considerar a física do mundo
- Considerar os outros objetos ao seu redor

Aplicações



Unidades em jogos RTS

Carros em jogos de corrida



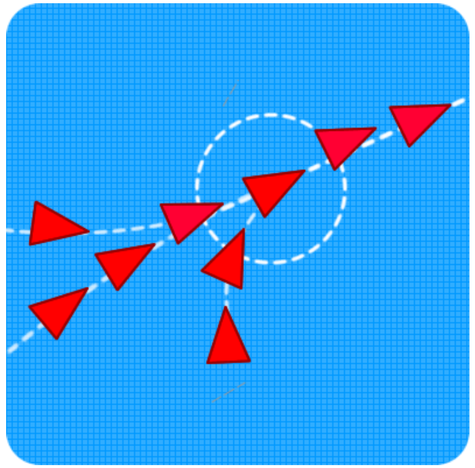
Inimigos voadores em jogos de ação

Entre outros ...

Comportamentos de navegação

Desenvolvido por Craig Reynolds nos anos 80.

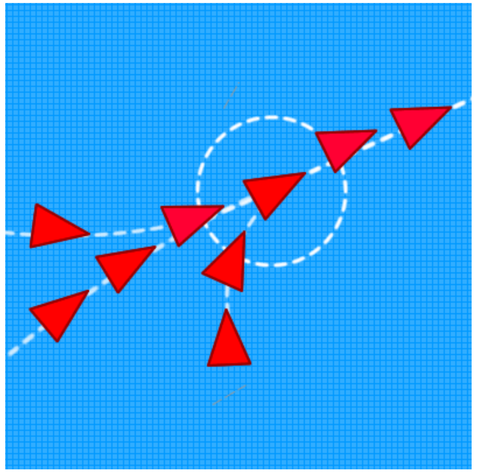
“Movimentar agentes autônomos de maneira orgânica ”



Sistemas complexos de agentes:

- Cada agente se move de maneira independente
- Um agente percebe apenas suas redondezas
- Movimentos reativos, sem planejamento

Comportamentos de navegação

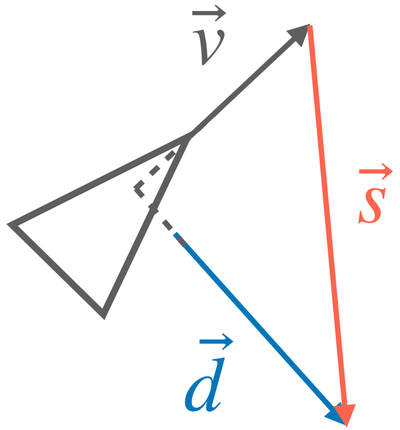


Reynolds modelou uma série de comportamentos naturais:

- Procurar/Fugir
- Passear
- Seguir um caminho
- Seguir um campo de fluxo
- Movimentação em rebanho
- ...

Força de direção

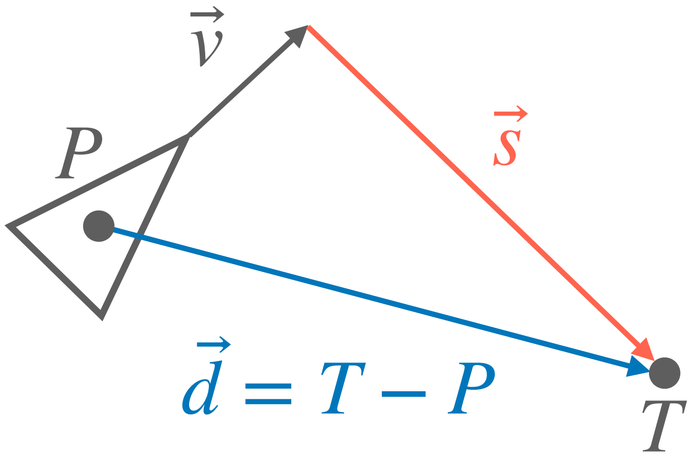
A base dos movimentos de navegação é a **força de direção** (*steering force*) $\vec{s} = \vec{d} - \vec{v}$, onde $\vec{d}$ é a **força desejada** (desired) e $\vec{v}$ é a velocidade atual do objeto.

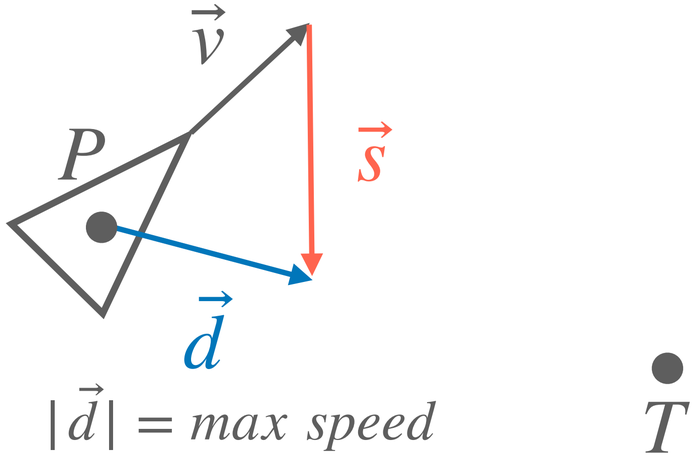


A definição de força de direção é a mesma para todos os comportamento de navegação, o que muda é a **força desejada** $\vec{d}$!

Procurar (seek)

Para o comportamento de **procurar**, a **força desejada** é o vetor $\vec{d} = T - P$, onde T é a posição de um dado alvo e P é a posição do objeto.





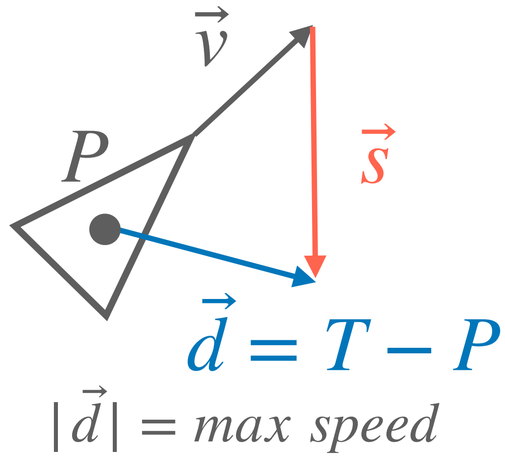
Para que a navegação respeite as propriedades físicas do objeto, a **força desejada** deve ser reescalada para sua velocidade máxima.

Implementação

Para implementar qualquer comportamento de navegação, basca executar os seguintes passos a cada atualização (update) do objeto em movimento (ex. NPC):

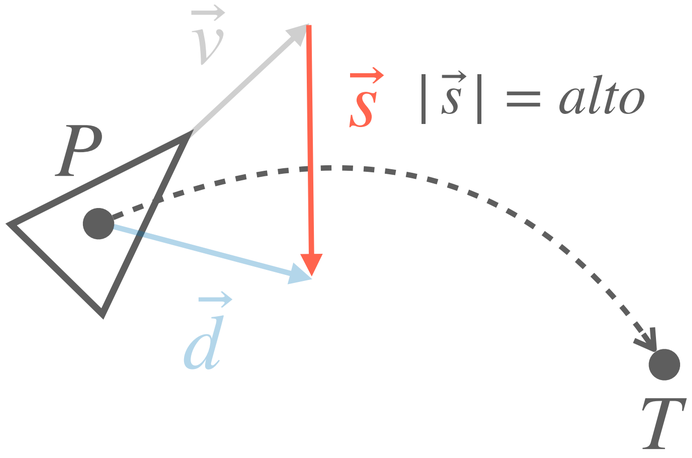
1. Calcular a força desejada $\vec{d}$ (depende do comportamento)
2. Calcular a força de direção (steering force) $\vec{s} = \vec{d} - \vec{v}$
3. Aplicar a força $\vec{s}$ ao objeto (`RigidBody::ApplyForce(`$\vec{s}$`)`)

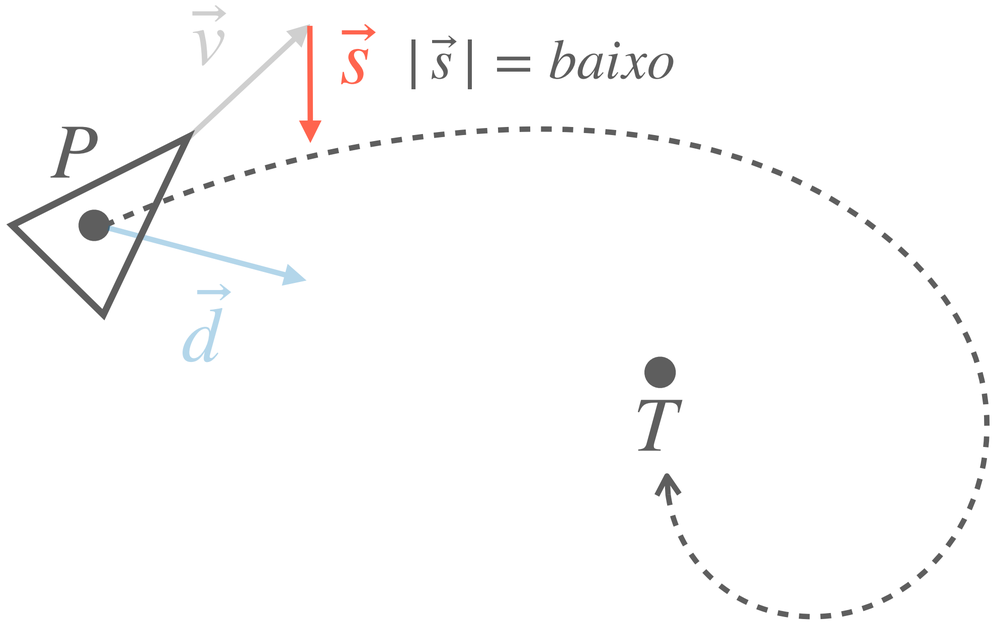
```
# Exemplo: Seek
Vector2 d = target - GetPosition();
Vector2 s  = d - v
mRigidBody->ApplyForce(s)

# Rotacionar em direção ao vetor velodidade
SetRotation(Math::Atan2(v.y, v.x));
```



Modelando Diferentes Habilidades de Direção

A habilidade de direção (handle) de um objeto pode ser controlada **limitando o comprimento** da **força de direção $\vec{s}$.**

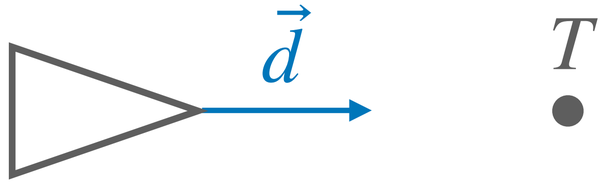




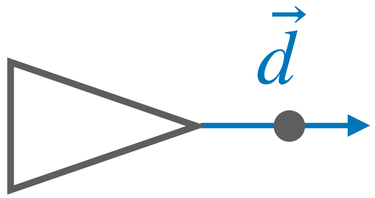
Chegar (arrive)

Atualmente, o comportamento de procurar reescala a **força desejada** com velocidade máxima a cada quadro:

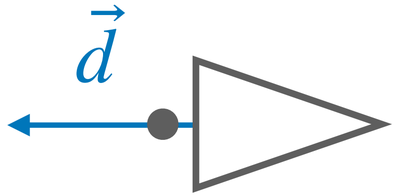
Quadro 1: mover o mais rápido possível!



Quadro 2: mover o mais rápido possível!



Quadro 3: mover o mais rápido possível!



Chegar (arrive)

Para **chegar** no alvo de maneira suave, é necessário reduzir o comprimento da **força desejada** $\vec{d}$ se a distância entre o objeto P e o alvo T for menor do que um dado raio r.

Quadro 1: Eu estou longe do alvo.

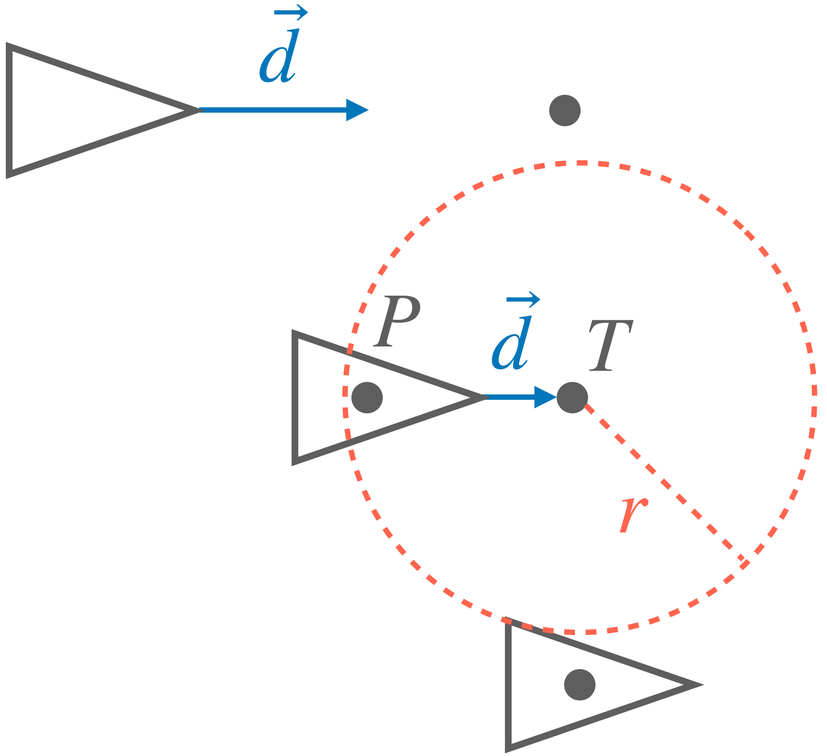
$$|d| = max\ speed$$

Quadro 2: Eu estou perto do alvo ($|d| < r$)

$$|d| = \frac{|d|}{r} * max\ \ speed$$

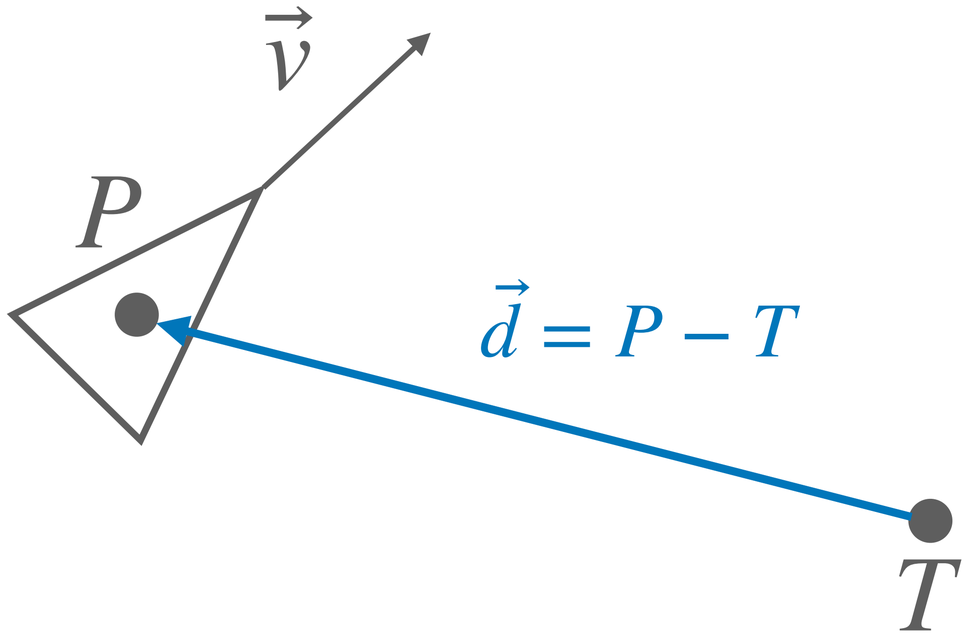
Quadro 3: Eu cheguei no alvo ($|d| < r$)

$$|d| = \frac{|d|}{r} * max\ \ speed$$



Exercício 1: Fugir (Flee)

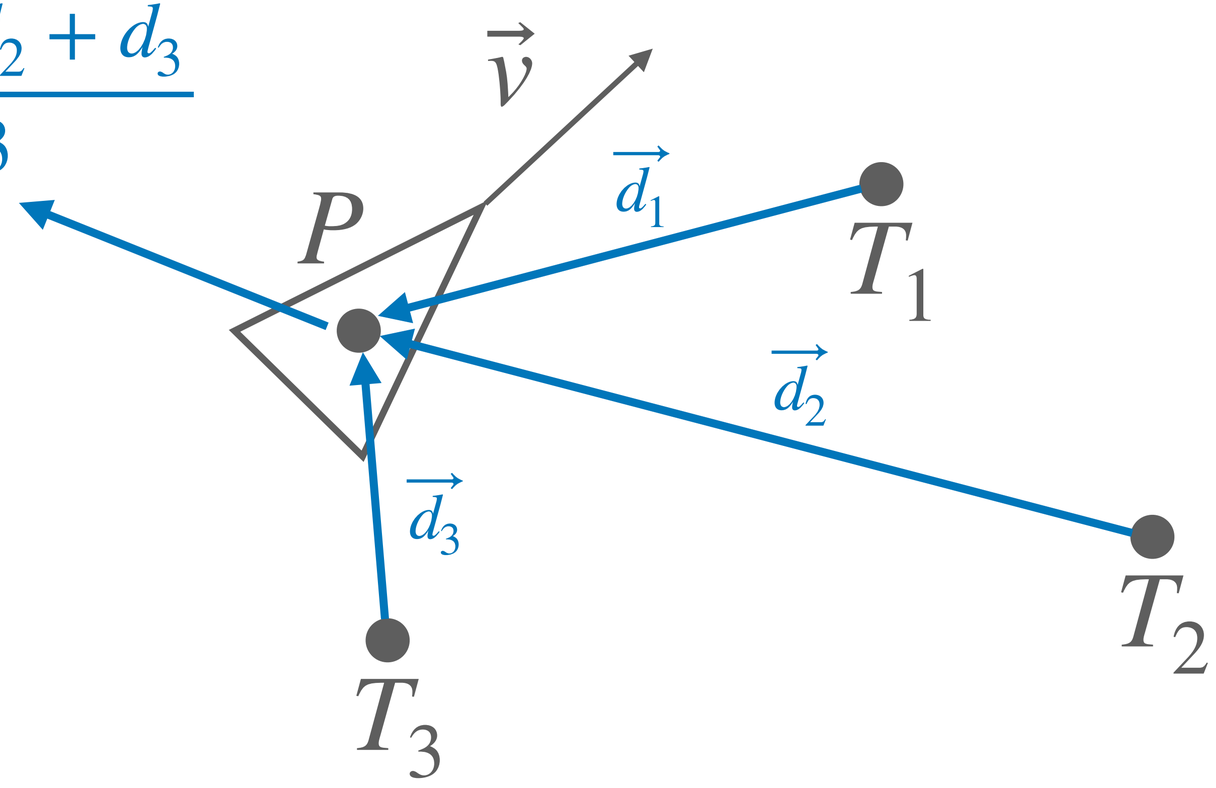
Para o comportamento de **fugir** de um alvo T, a **força desejada** é o vetor $\vec{d} = ?$



Exercício 2: Fugir de Múltiplos Alvos

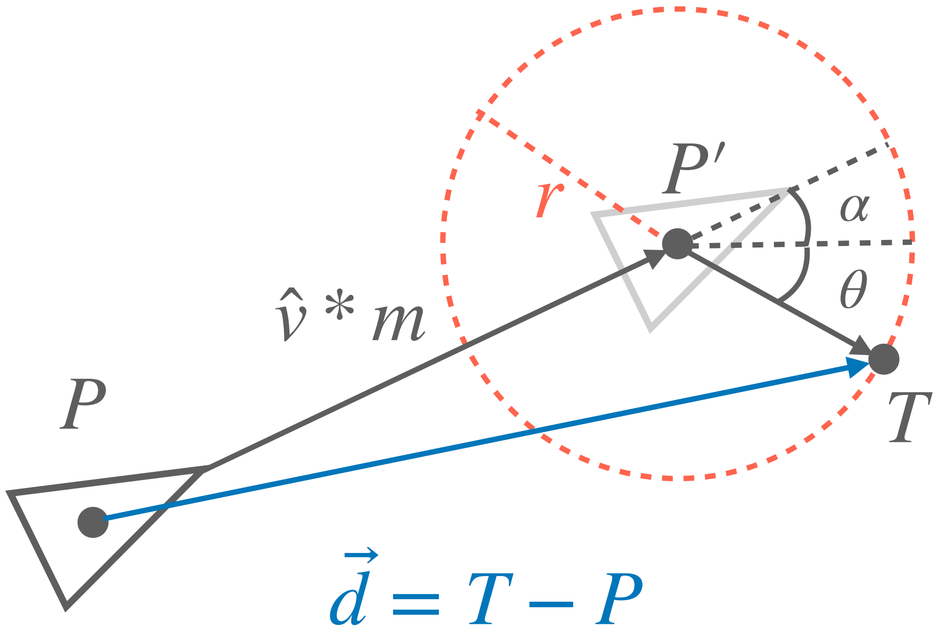
Para o comportamento de **fugir** de múltiplos alvos T_i, a **força desejada** é o vetor $\vec{d} = ?$

$$\vec{d} = \frac{\overrightarrow{d_1} + \overrightarrow{d_2} + \overrightarrow{d_3}}{3}$$



Passear (wander around)

Para o comportamento de **passear aleatoriamente**, a **força desejada** tem como direção um ponto T aleatório em um círculo com centro na posição futura P' e raio r.

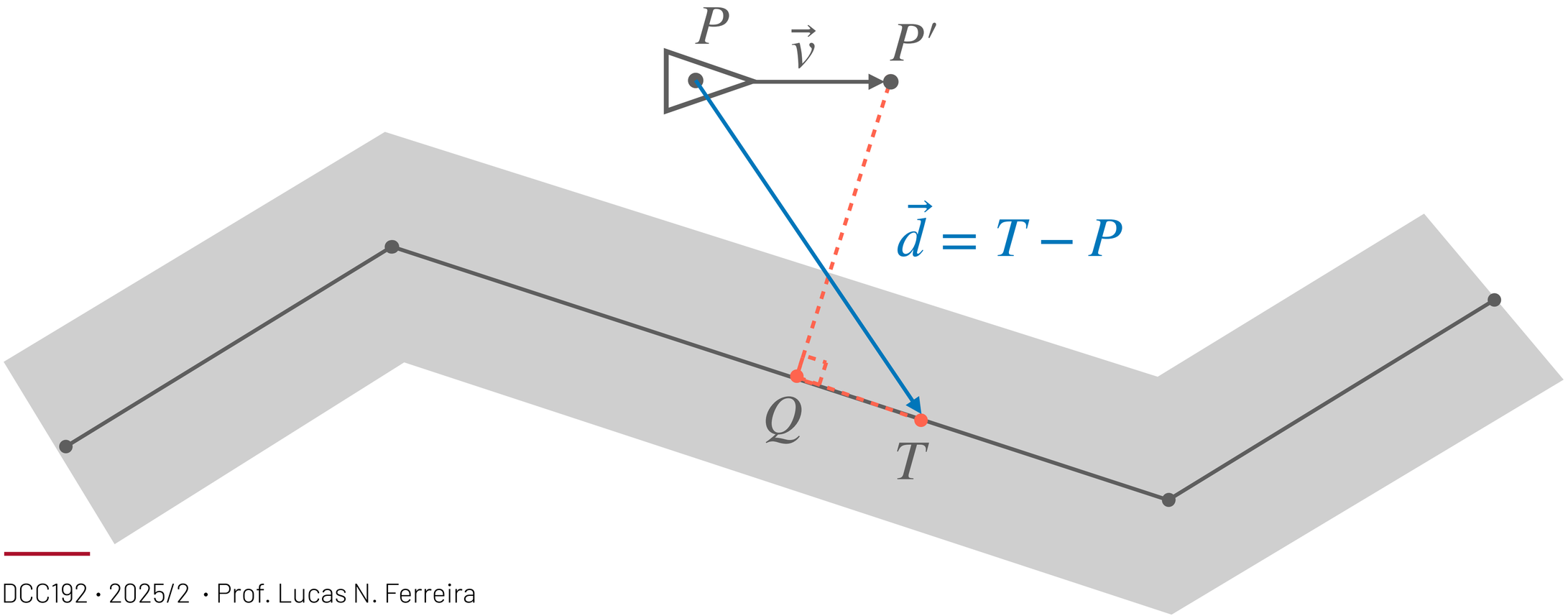


$$P' = P + (\vec{v} * m)$$
$$\theta \mathrel{+}= \texttt{random}(-c, c)$$
$$T_x = P' + r * cos(\alpha + \theta)$$
$$T_y = P' + r * sin(\alpha + \theta)$$
$$\texttt{seek}(T)$$

- m é um fator de projeção de velocidade
- c é uma constante de variação angular
- α é a rotação atual do objeto

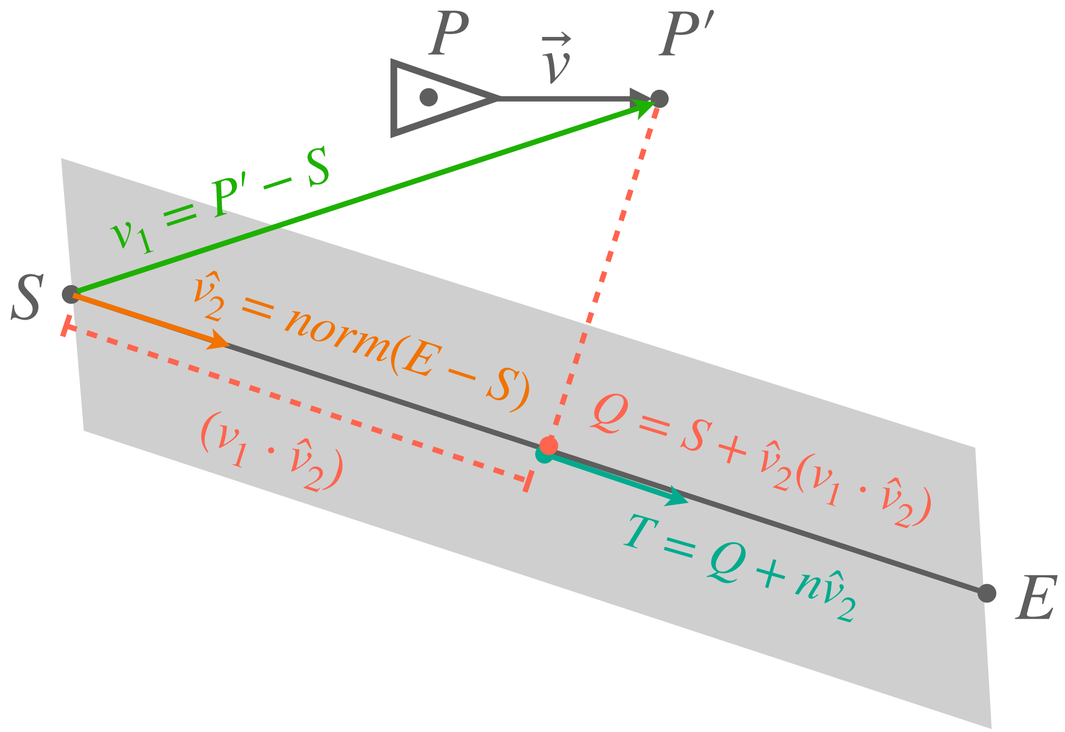
Seguir um caminho (Path following)

Para o comportamento de **seguir um caminho**, a **força desejada** $\vec{d}$ tem como direção um ponto T a frente da projeção Q da posição futura P' em um dado caminho.



Seguir um caminho (Path following)

Vamos assumir um caminho com apenas um segmento de caminho $\overrightarrow{SE}$ para facilitar o calculo de P', Q e T

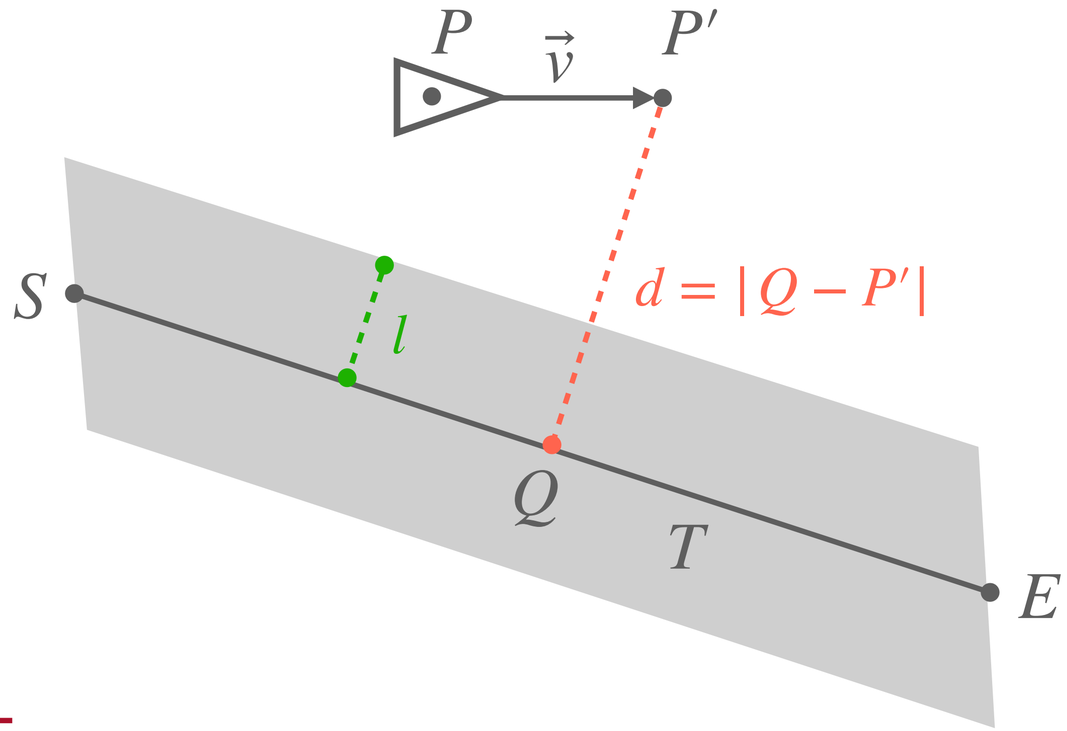


$$P' = P + (\vec{v} * m)$$
$$v_1 = P' - S$$
$$\hat{v}_2 = \texttt{norm}\,(E - S)$$
$$Q = S + \hat{v}_2 * (v_1 \cdot \hat{v}_2)$$
$$T = Q + \hat{v}_2 * n$$

- m é um fator de projeção de velocidade
- n é a distância desejada entre Q e T

Seguir um caminho (Path following)

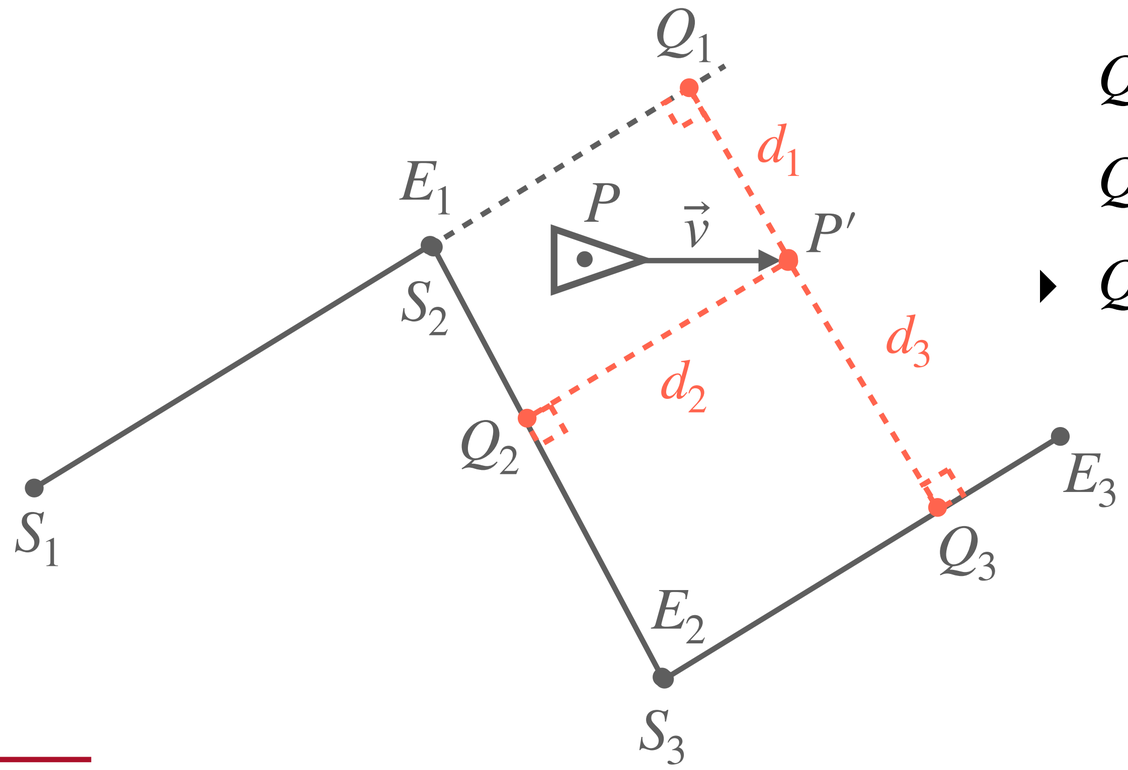
O objeto persegue T apenas se a distância d entre a posição futura P' for maior do que a largura l do caminho. Caso contrário, o objeto se move normalmente de acordo com sua velocidade $\vec{v}$




```
d = |Q - P'|
if d > l:
  seek(T)
```

Seguir um caminho (Path following)

Para seguir um caminho com mais de um segmento, calculados a projeção Q_s para todos os segmentos s e escolhemos aquela que está contida em $\overrightarrow{SE}_s$ com menor distância $d_s = |Q_s - P'|$



- Q_1 tem a menor distância d_1, mas não está em $\overrightarrow{SE}_1$
- Q_2 está em $\overrightarrow{SE}_2$, mas $d_2 > d_3$
- Q_3 está em $\overrightarrow{SE}_3$ e tem a menor distância! [escolhido]

Próxima aula

A23: Geração Procedural de Conteúdo

- Métodos Construtivos
  - Funções de Ruído
  - Gramáticas Generativas
- Métodos Baseados em Busca
  - Algoritmos Evolutivos
- Métodos Baseados em Aprendizado