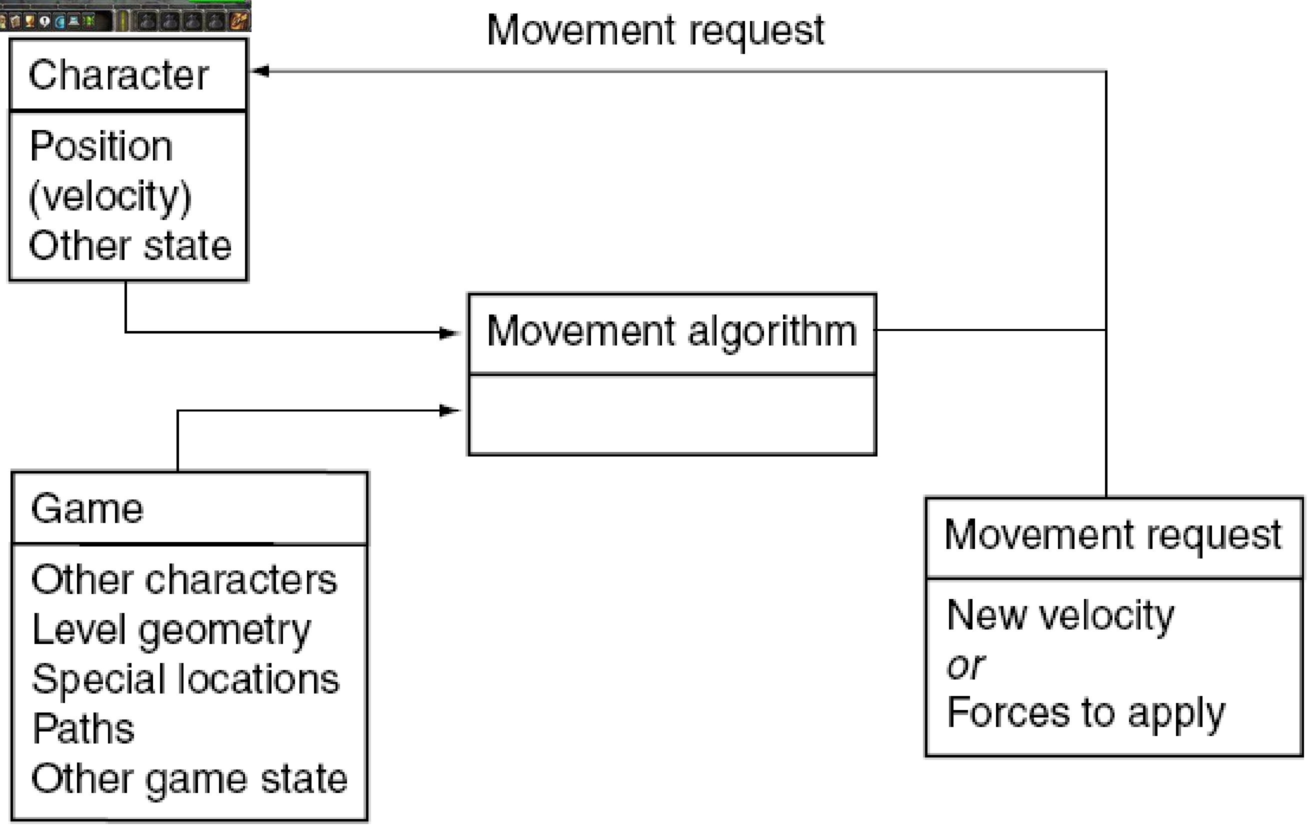
Movement: Basic Movement

Overall Movement Hierarchy



Overall Movement Hierarchy

simulates the motion and updates the position/velocity of the character

computes the desired direction of the motion at every frame

computes the actual direction of the motion at every frame

based on the state of the game decides action and computes a path

Movement request

Character

Position
(velocity)

Movement algorithm

Game

Other characters
Level geometry
Special locations
Paths
Other game state

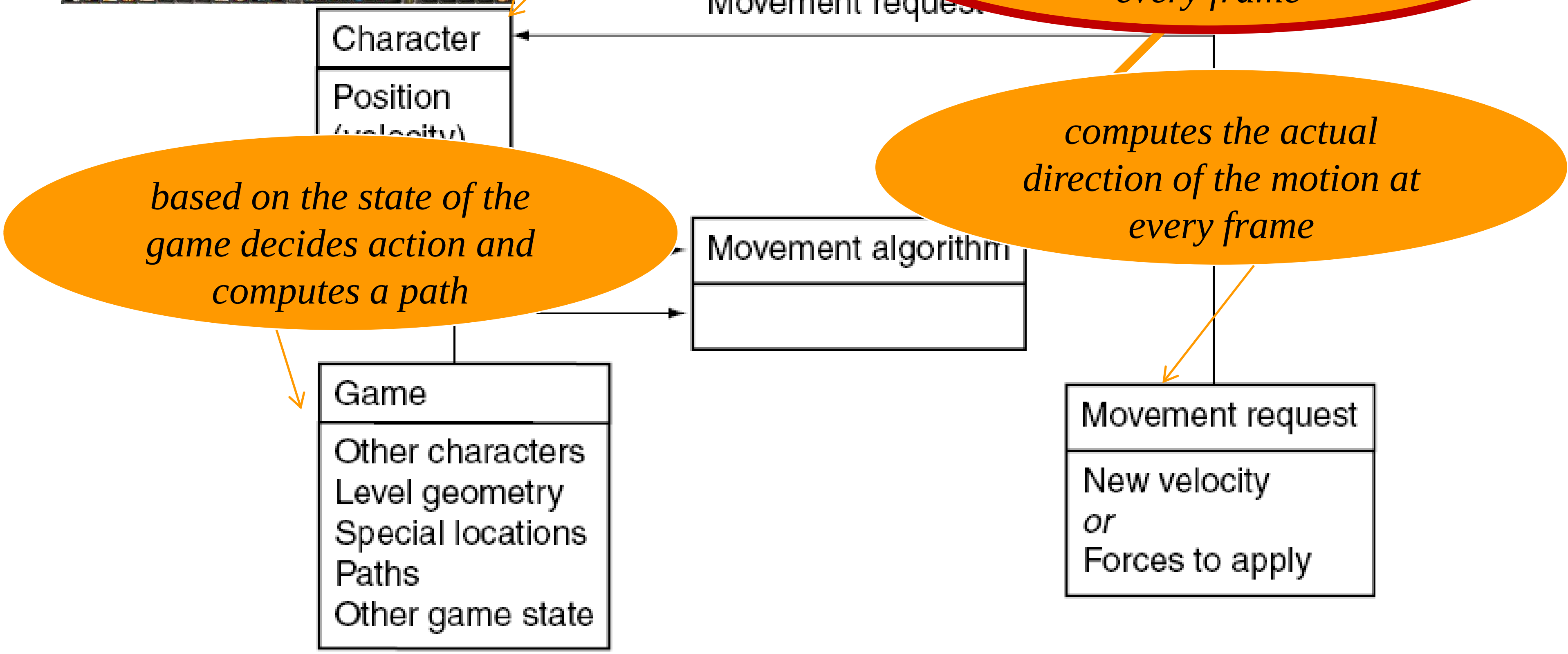
Movement request

New velocity
or
Forces to apply

from "Artificial Intelligence for Games" by I. Millington & J. y ge

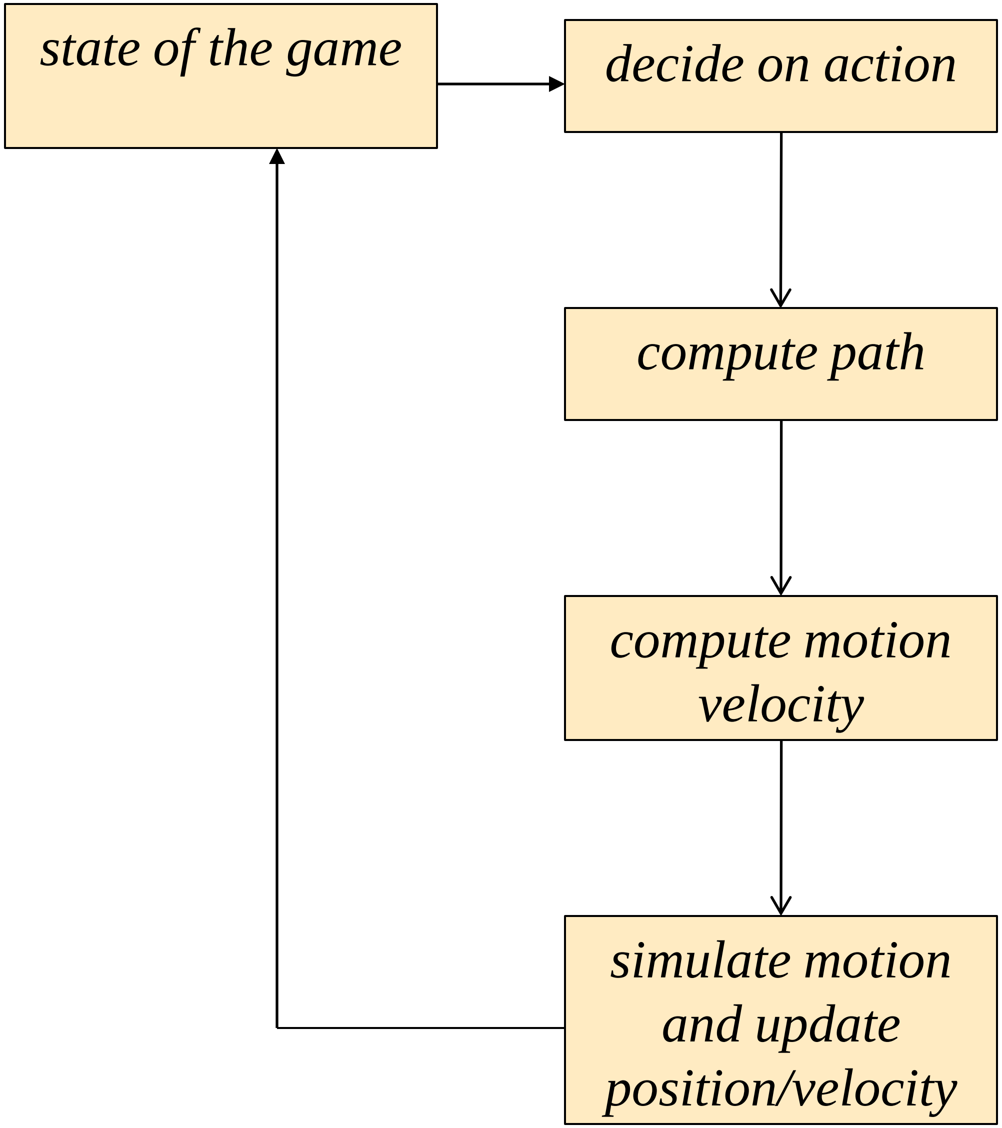
Overall Movement Hierarchy

simulates the motion and updates the position/velocity of the character

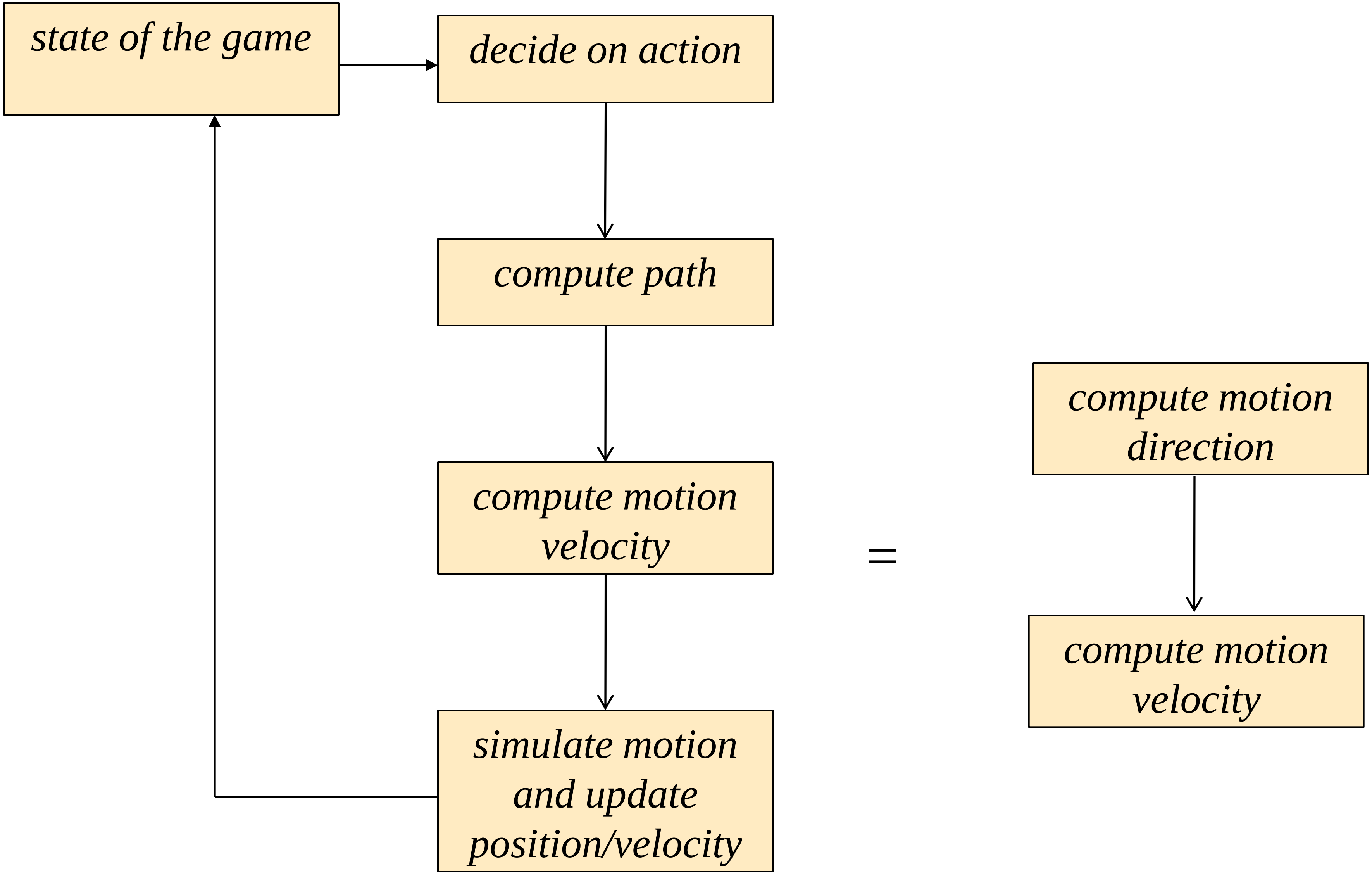


from "Artificial Intelligence for Games" by I. Millington & J. y ge

Overall Movement Hierarchy

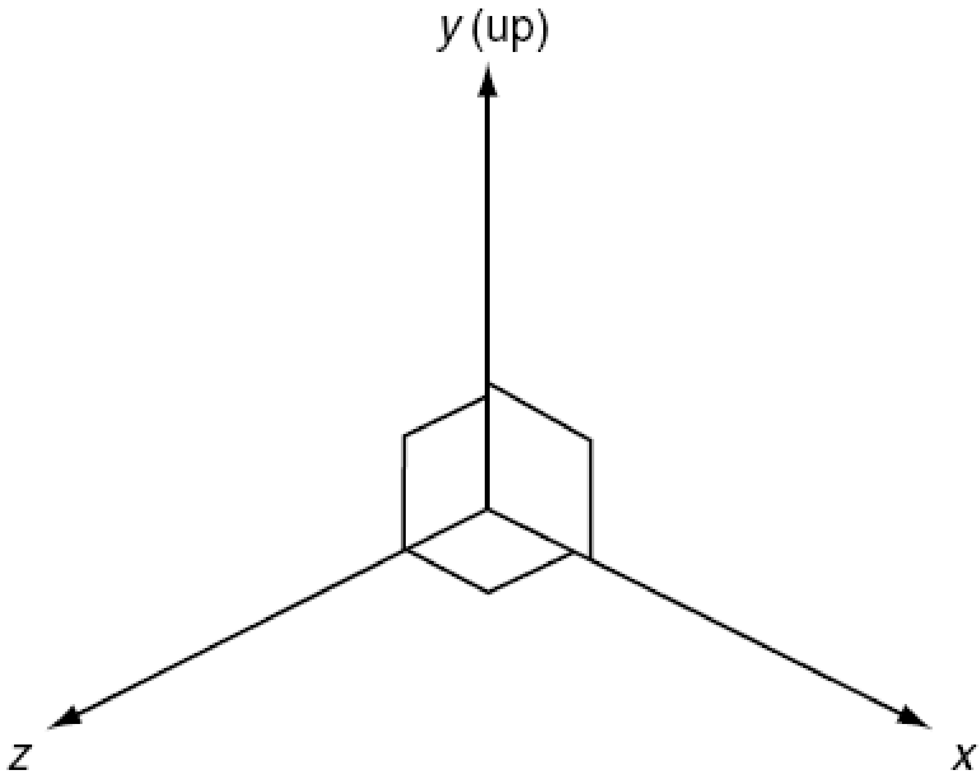


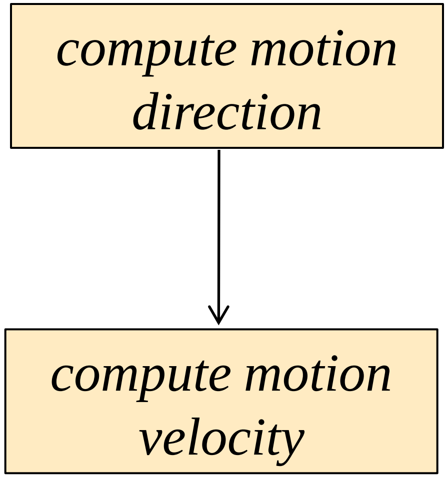
Overall Movement Hierarchy



Definition of the System

- Coordinate system for position *P=[x,z]* or *P=[x,y,z]*





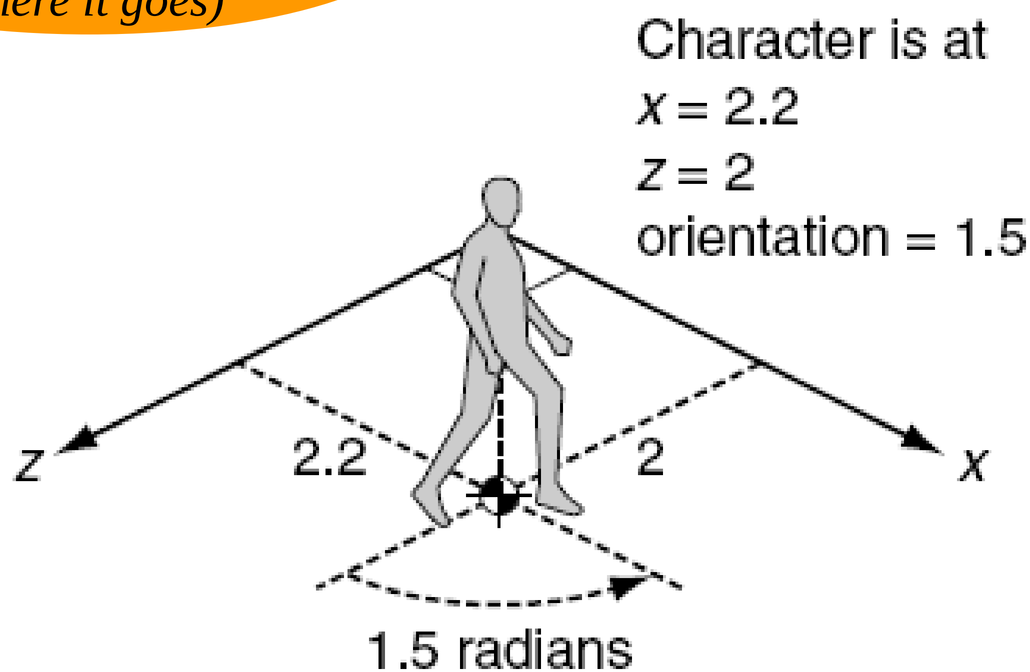
from "Artificial Intelligence for Games" by I. Millington & J. y ge

Definition of the System

- Coordinate system for orientation *Ψ (in rads)*

orientation is where it looks (not necessarily where it goes)

character is reduced to a point



compute motion direction

↓

compute motion velocity

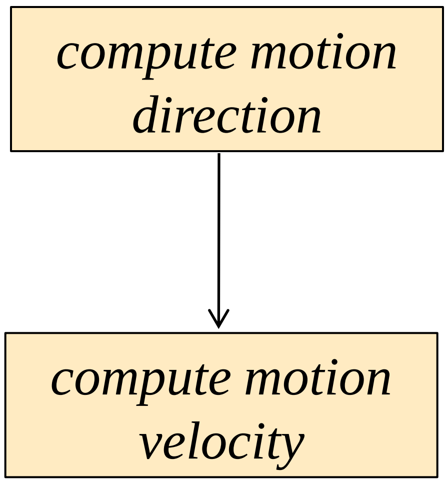
from "Artificial Intelligence for Games" by I. Millington & J. y ge

Definition of the System

- Kinematic movement

State of system:

position P = [x,z]

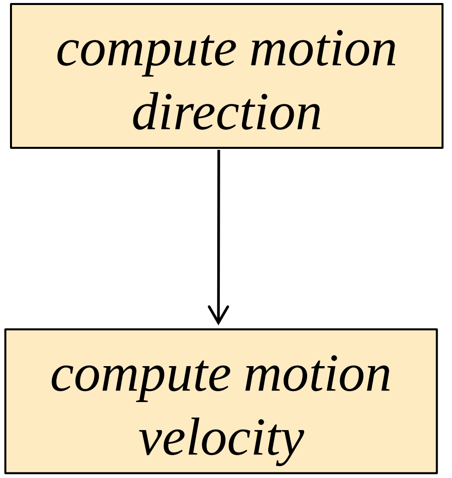


Definition of the System

- Kinematic movement

Output of system:

velocity V = [Vx,Vy] =[magnitude v, direction θ], orientation Ψ



Definition of the System

- Kinematic movement

Output of system:
velocity V = [Vx,Vy] =[magnitude v, direction θ],
orientation Ψ

What Ψ should be set to?

compute motion direction

↓

compute motion velocity

Definition of the System

- Kinematic movement

Output of system:

velocity $V = [V_x, V_y] = [$magnitude v, direction $\theta]$, orientation Ψ

Any problems with this approach?

compute motion direction

↓

compute motion velocity

Definition of the System

- Kinematic movement

Output of system:

velocity V = [Vx,Vy] =[magnitude v, direction θ], orientation Ψ

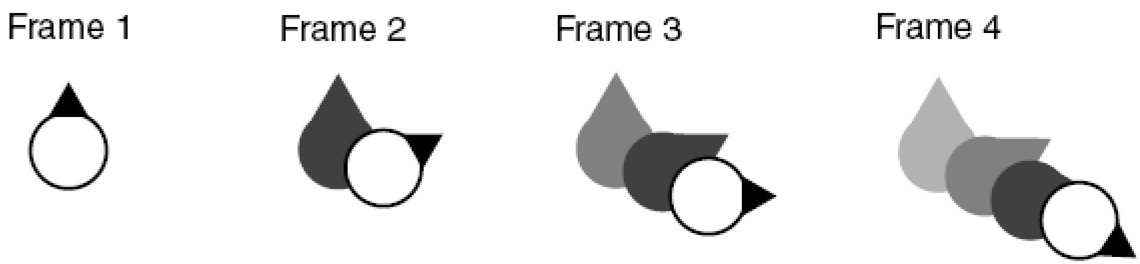
Any solutions?

compute motion direction

compute motion velocity

Definition of the System

- Kinematic movement



from "Artificial Intelligence for Games" by I. Millington & J. y nge

Output of system:

velocity V = [Vx,Vy] =[magnitude v, direction θ], orientation Ψ

compute motion direction

compute motion velocity

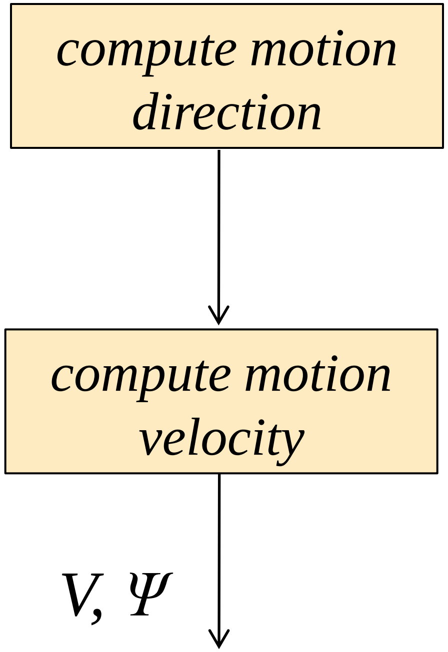
instantaneous changes to orientation & velocity can be smoothed out here by interpolation

Definition of the System

- Kinematic movement

Output of system:

velocity $V = [V_x, V_y] = [\text{magnitude } v, \text{direction } \theta]$, *orientation* Ψ

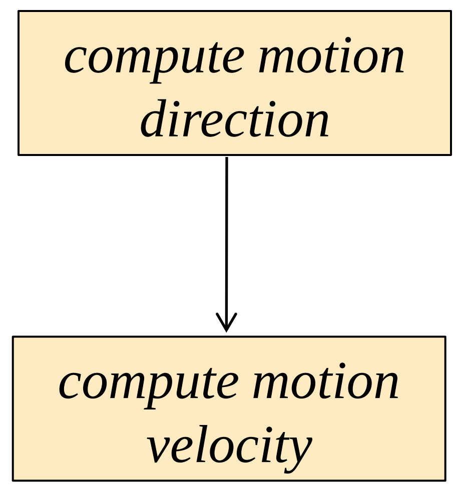


Output of the overall system: V, Ψ

Definition of the System

- Dynamic movement

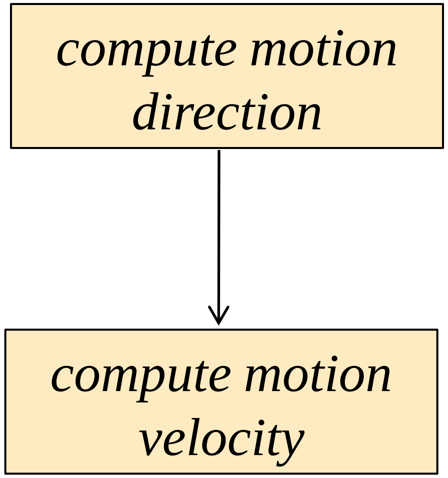
State of system:
position P = [x,z],
velocity V = [Vx,Vy] =[magnitude v, direction θ],
orientation Ψ
angular speed $d\Psi$



Definition of the System

- Dynamic movement

Output of system:
acceleration A = [Ax,Az],
angular acceleration ddΨ



Definition of the System

- Dynamic movement

Output of system:
acceleration A = [Ax,Az],
angular acceleration ddΨ

compute motion direction

↓

compute motion velocity

Also limit V, dΨ to their max values

Output of the overall system: *V, Ψ*

Position Update According to Motion Equation

- Continuous formulation for constant acceleration:

$$a = \frac{dv}{dt}$$

$$\int_0^t a\,dt = \int_0^t \frac{dv}{dt}\,dt$$

$$at = v(t) - v_0$$

$$v(t) = v_0 + at$$

$$v = \frac{dP}{dt}$$

$$\int_0^t v\,dt = \int_0^t \frac{dP}{dt}\,dt$$

$$\int_0^t (v_0 + at)\,dt = P(t) - P_0$$

$$v_0 t + \frac{1}{2}at^2 = P(t) - P_0$$

$$P(t) = P_0 + v_0 t + \frac{1}{2}at^2$$

compute motion velocity

simulate motion and update position/velocity

Position Update According to Motion Equation

- Continuous formulation for constant acceleration:

$$a = \frac{dv}{dt}$$

$$\int_0^t a\,dt = \int_0^t \frac{dv}{dt}\,dt$$

$$at = v(t) - v_0$$

$$v(t) = v_0 + at$$

$$v = \frac{dP}{dt}$$

same derivation for orientation Ψ and angular speed $d\Psi$

$$\int_0^t (v_0 + at)\,dt = P(t) - P_0$$

$$v_0 t + \frac{1}{2}at^2 = P(t) - P_0$$

$$P(t) = P_0 + v_0 t + \frac{1}{2}at^2$$

compute motion velocity

↓

simulate motion and update position/velocity

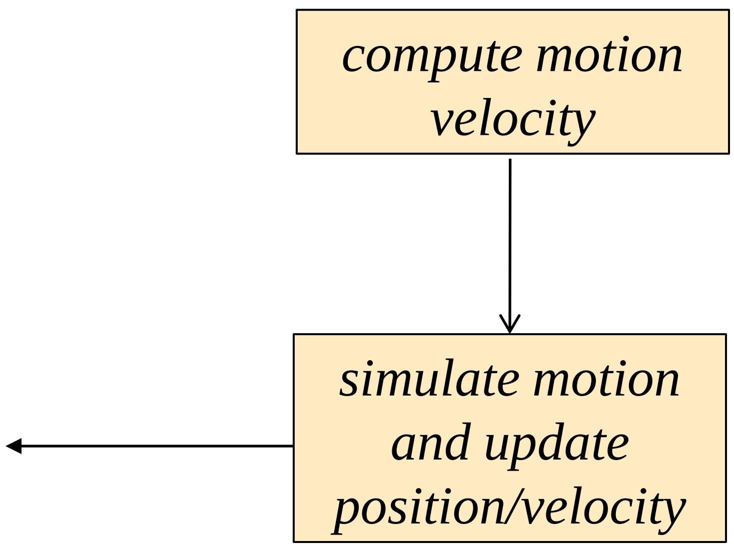
Position Update According to Motion Equation

- Discrete formulation:

Units for Δt?

$$V[t+1] = V[t] + A[t]\Delta t$$
$$P[t+1] = P[t] + V[t]\Delta t + 0.5A[t]\Delta t^2$$

$$d\Psi[t+1] = d\Psi[t] + dd\Psi[t]\Delta t$$
$$\Psi[t+1] = \Psi[t] + d\Psi[t]\Delta t + 0.5dd\Psi[t]\Delta t^2$$



Position Update According to Motion Equation

- Discrete formulation simplified:

$$V[t+1] = V[t] + A[t]\Delta t$$
$$P[t+1] = P[t] + V[t]\Delta t$$

Why can we do it?

$$d\Psi[t+1] = d\Psi[t] + dd\Psi[t]\Delta t$$
$$\Psi[t+1] = \Psi[t] + d\Psi[t]\Delta t$$

compute motion velocity

↓

simulate motion and update position/velocity

Kinematic Seek Behavior

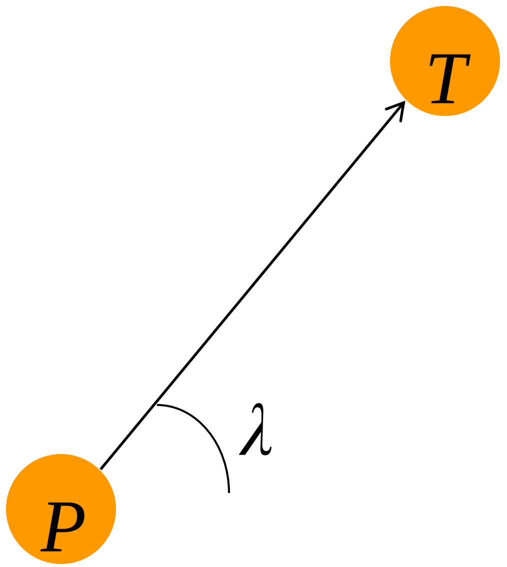
- Move towards a target point T

  $V=[v, \theta]= [\text{max. speed}, \lambda]$

  $\Psi=\lambda$

compute motion direction

How do we do flee?



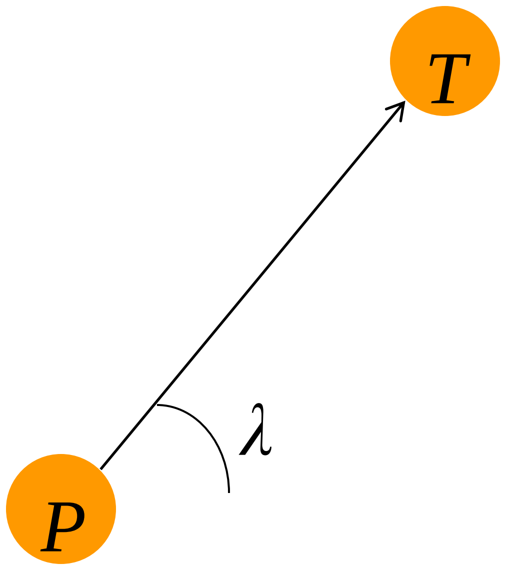
Kinematic Flee Behavior

- Move away from a target point T

  $V=[v, \theta]=[max.\ speed, \lambda+\pi]$

  $\Psi=\lambda+\pi$

compute motion direction



Kinematic Seek Behavior

- Move towards a target point T

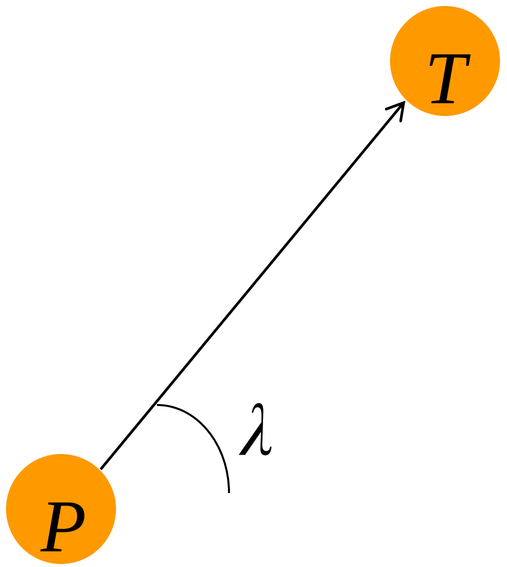
  $V=[v, \theta]= [max.\ speed, \lambda]$

  $\Psi=\lambda$

compute motion direction

Problems approaching a moving target?

Solutions?



Kinematic Arrival Behavior

compute motion direction

- Approach a target point T

  if d < arrival radius

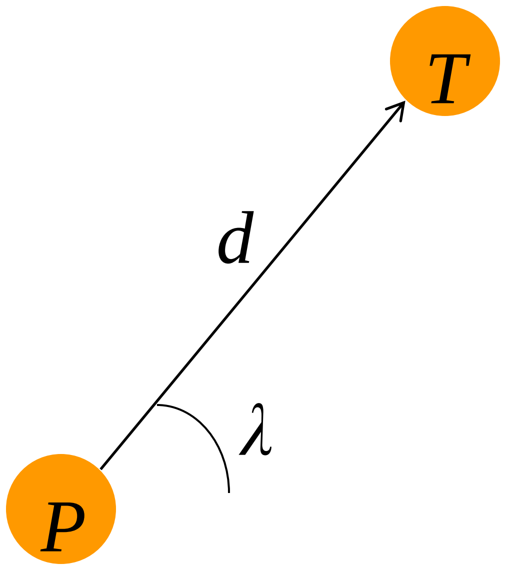
  $V=[v, \theta]= [0,\lambda]$

  $\Psi=\lambda$

  else

  $V=[v, \theta]= [K*d,\lambda]$ *for some constant K*

  $\Psi=\lambda$



Kinematic Wander Behavior

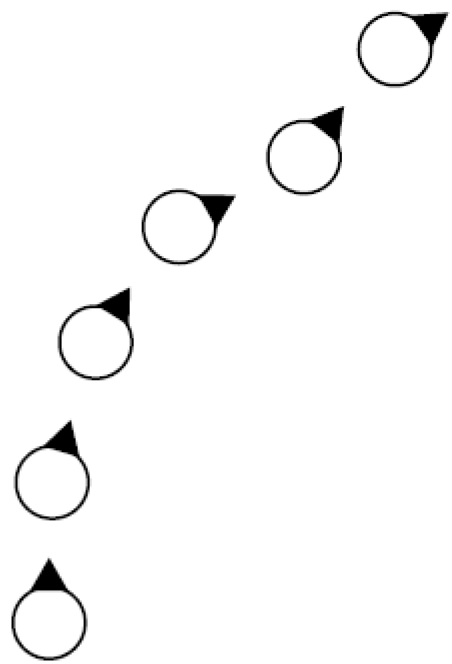
compute motion direction

- Random wandering

  $d\Psi$ = *random with bias towards 0*

  $\Psi = \Psi + d\Psi$

  $V = [v, \theta] = [max.\ speed, \Psi]$



from "Artificial Intelligence for Games" by I. Millington & J. y ge

Dynamic Seek Behavior

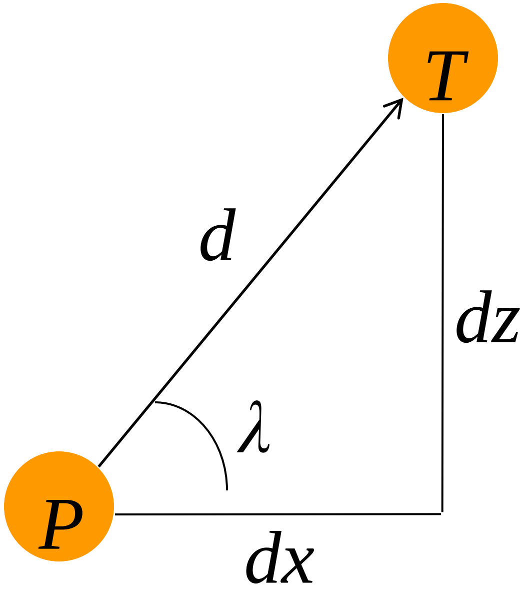
- Move towards a target point *T*

  $A = max.\ accel * normalize([dx, dz])$

  $dd\Psi = K(\Psi - \lambda)$ *limited by max. angular acceleration*

compute motion direction

Take care for angle wrapping



Dynamic Seek Behavior

- Move towards a target point *T*

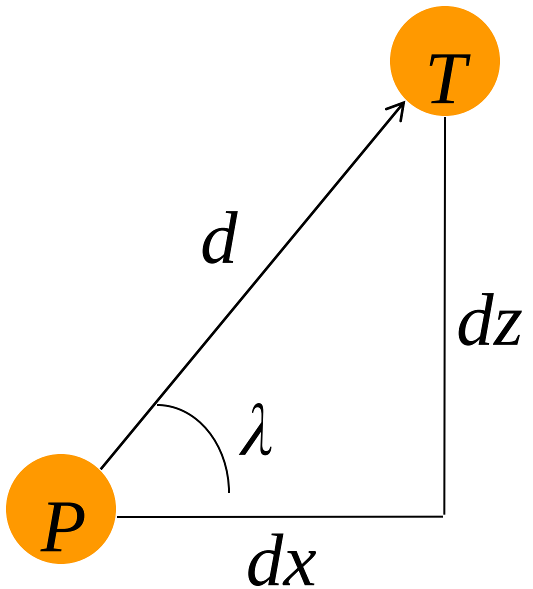
  $A = max.\ accel * normalize([dx, dz])$

  $dd\Psi = K(\Psi - \lambda)$ *limited by max. angular acceleration*

compute motion direction

Take care for angle wrapping

How do we do flee?



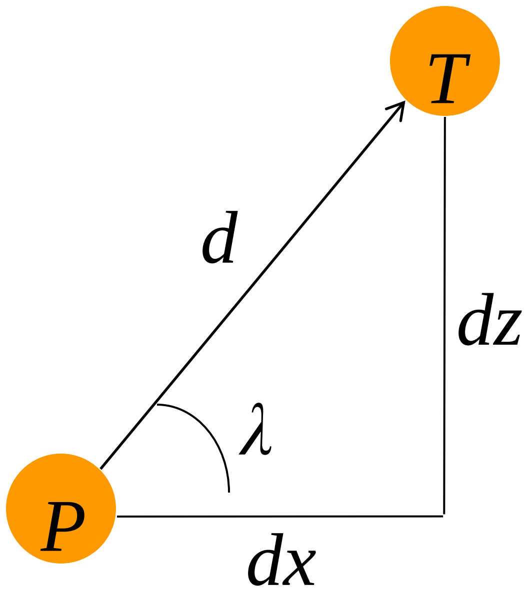
Dynamic Flee Behavior

compute motion direction

- Move away from a target point T

$$A = -max.\ accel * normalize([dx, dz])$$

$$dd\Psi = K(\Psi - \lambda - \pi) \text{ limited by max. angular acceleration}$$



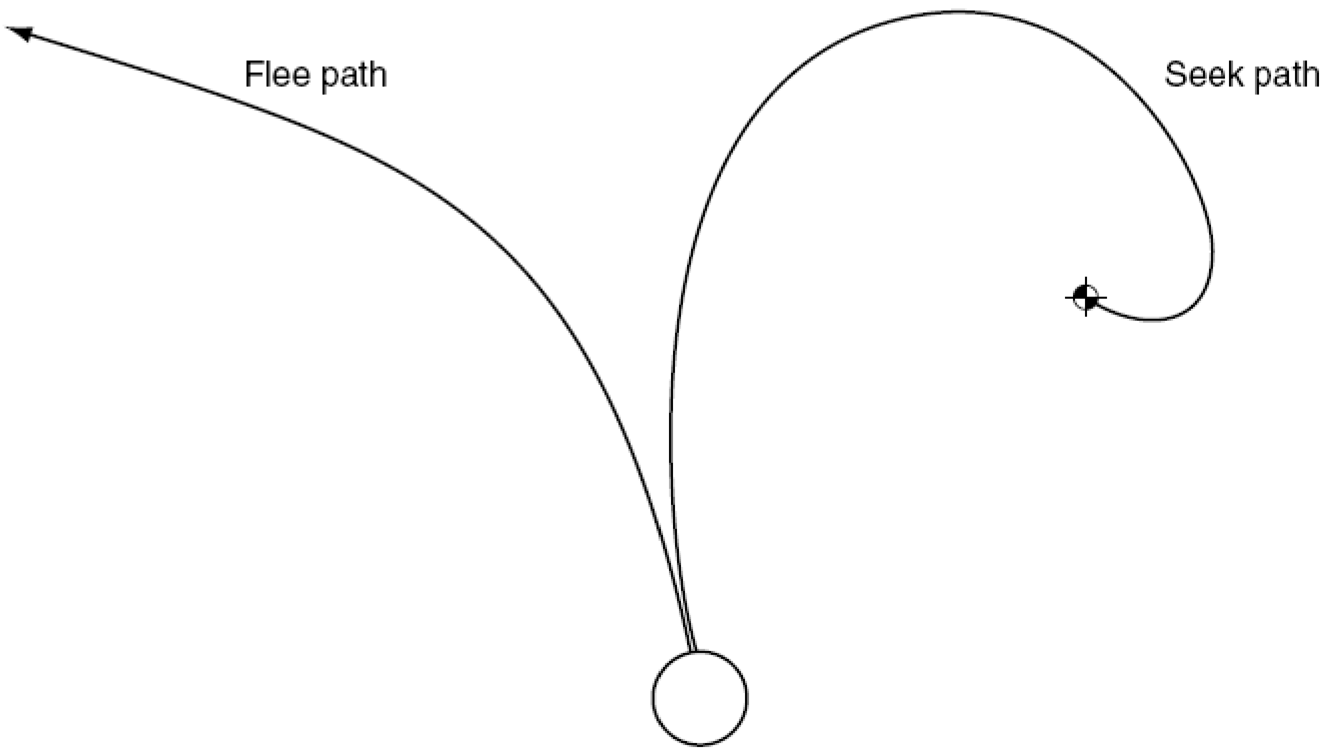
Dynamic Flee Behavior

compute motion direction

- Move away from a target point *T*

  $A = -max.\ accel * normalize([dx, dz])$

  $dd\Psi = K(\Psi - \lambda - \pi)$ *limited by max. angular acceleration*



from "Artificial Intelligence for Games" by I. Millington & J. y ge

Dynamic Flee Behavior

compute motion direction

- Move away from a target point T

  $A = -max.\ accel * normalize([dx, dz])$

  $dd\Psi = K(\Psi - \lambda - \pi)$ *limited by max. angular acceleration*

http://www.red3d.com/cwr/steer/SeekFlee.html

Dynamic Arrive Behavior

- Approach a target point *T*

  if d < *arrival radius then* $v_{des}=0$

  else $v_{des}=K*d$ *for some constant K*

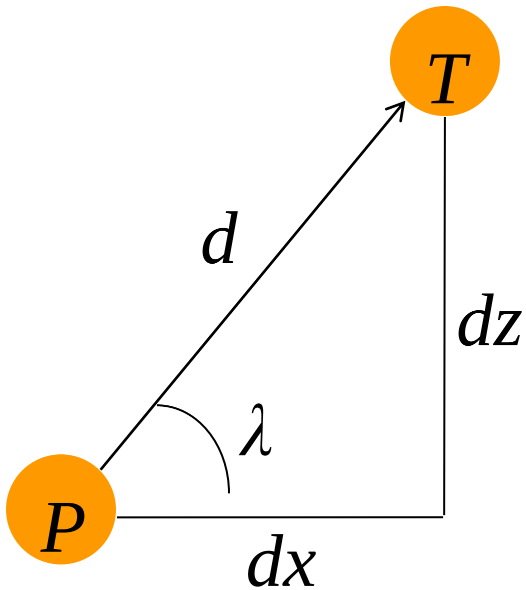
  $A=K_1*(v_{des}-v_{current})*normalize([dx,dz])$

  $dd\Psi=K(\Psi-\lambda-\pi)$ *limited by max. angular acceleration*

compute motion direction

K_1 can be set to 1/T (T is in secs)

Meaning of T?



Dynamic Arrive Behavior

compute motion direction

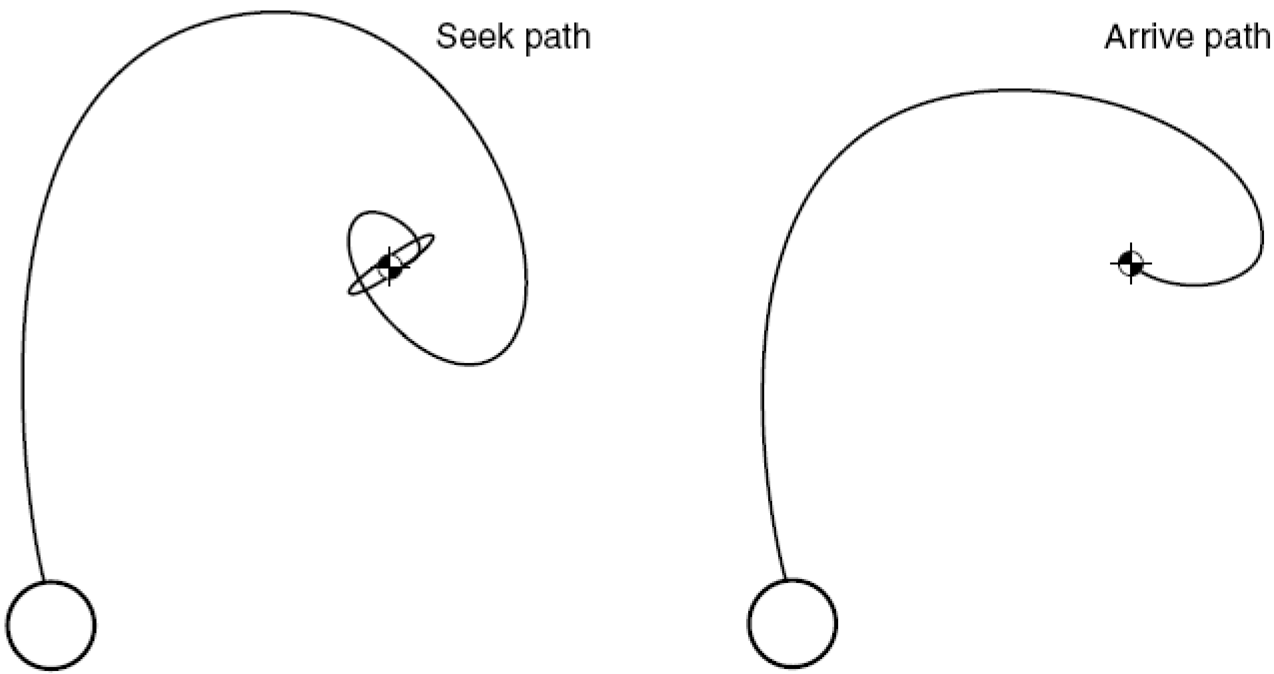
- Approach a target point T

  if d < arrival radius then $v_{des}=0$

  else $v_{des}=K*d$ *for some constant K*

  $A=K_1*(v_{des}-v_{current})*normalize([dx,dz])$

  $dd\Psi=K(\Psi-\lambda-\pi)$ *limited by max. angular acceleration*



from “Artificial Intelligence for Games” by I. Millington & J. y ge

Dynamic Arrive Behavior

compute motion direction

- Approach a target point *T*

  if d < arrival radius then $v_{des}=0$

  else $v_{des}=K*d$ *for some constant K*

  $A=K_1*(v_{des}-v_{current})*normalize([dx,dz])$

  $dd\Psi=K(\Psi-\lambda-\pi)$ *limited by max. angular acceleration*

http://www.red3d.com/cwr/steer/Arrival.html

Dynamic Arrive Behavior

compute motion direction

- Approach a target point T

  if $d <$ *arrival radius then* $v_{des}=0$

  else $v_{des}=K*d$ *for some constant* K

  $A=K_1*(v_{des}-v_{current})*normalize([dx,dz])$

  $dd\Psi=K(\Psi-\lambda-\pi)$ *limited by max. angular acceleration*

How would you implement velocity matching behavior?

Dynamic Pursue Behavior

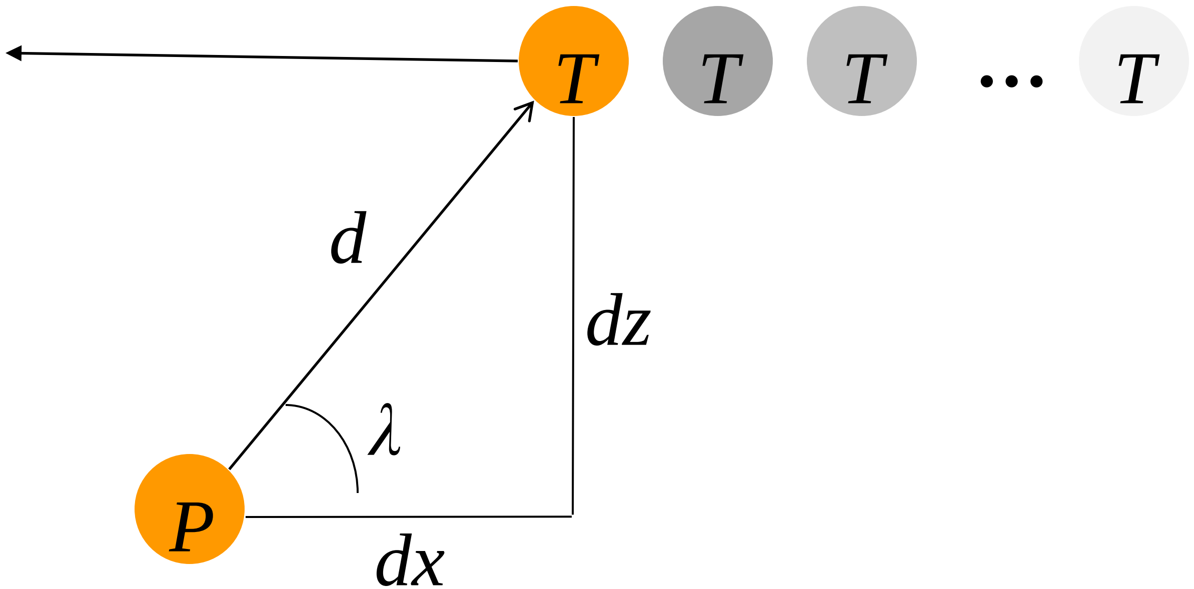
- Pursue a moving target *T* that isn't close

compute motion direction

Where will seek direct the character?

What's the problem?

Solutions?



Dynamic Pursue Behavior

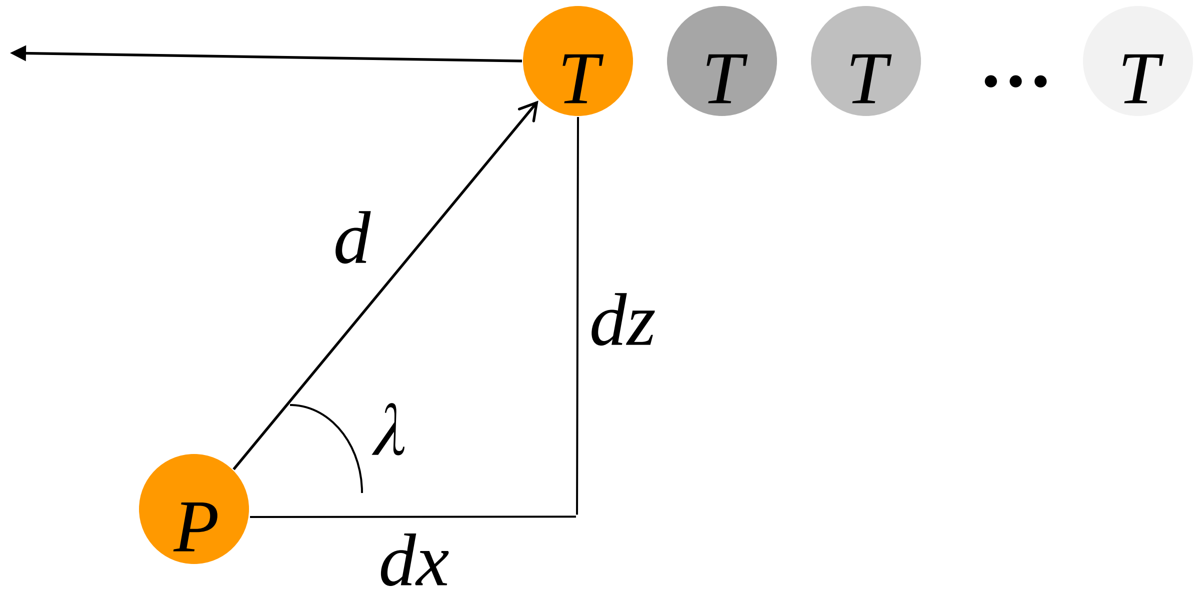
- Pursue a moving target *T* that isn't't close

seek with target position at:

$$d/v_{current} * V_T$$

compute motion direction

Where will pursue direct the character?



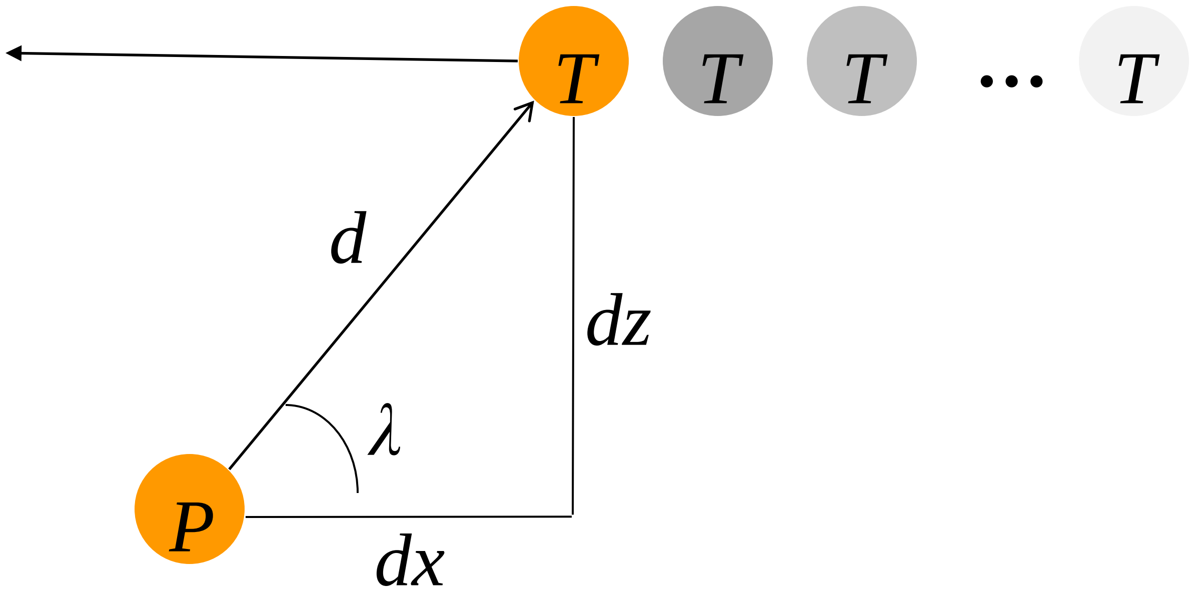
Dynamic Evade Behavior

- Evade a moving target *T* that isn't close

  flee with target position at:

  $d/v_{current} * V_T$

compute motion direction



Dynamic Evade Behavior

compute motion direction

- Evade a moving target *T* that isn't close

flee with target position at:

$$d/v_{current} * V_T$$

http://www.red3d.com/cwr/steer/PursueEvade.html