15-466
Computer Game Programming

Movement: Basic Movement

Maxim Likhachev
Robotics Institute
Carnegie Mellon University
Overall Movement Hierarchy

Character
- Position (velocity)
- Other state

Movement request

Movement algorithm

Game
- Other characters
- Level geometry
- Special locations
- Paths
- Other game state

Movement request
- New velocity
- Forces to apply
Overall Movement Hierarchy

- Based on the state of the game, decides action and computes a path.
- Computes the desired direction of the motion at every frame.
- Computes the actual direction of the motion at every frame.
- Simulates the motion and updates the position/velocity of the character.

(from “Artificial Intelligence for Games” by I. Millington & J. Funge)
Overall Movement Hierarchy

- **Movement request**
  - **Movement algorithm**
    - **Game**
      - Other characters
      - Level geometry
      - Special locations
      - Paths
      - Other game state
  - **Position (velocity)**
  - **Character**

- Computes the desired direction of the motion at every frame
- Simulates the motion and updates the position/velocity of the character
- Computes the actual direction of the motion at every frame

*Based on the state of the game decides action and computes a path*

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
Overall Movement Hierarchy

- **state of the game**
  - decide on action
    - compute path
      - compute motion velocity
        - simulate motion and update position/velocity
Overall Movement Hierarchy

- **state of the game**
- **decide on action**
  - **compute path**
  - **compute motion velocity**
    - **simulate motion and update position/velocity**
  - **compute motion direction**
    - **compute motion velocity**
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. Funge
Definition of the System

- Coordinate system for orientation $\Psi$ \textit{(in rads)}

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

character is reduced to a point

Character is at
$x = 2.2$
$z = 2$
orientation $= 1.5$

compute motion direction

compute motion velocity

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Definition of the System

• Kinematic movement

State of system: position $P = [x,z]$
Definition of the System

• Kinematic movement

Output of system:
velocity $V = [V_x, V_y] = [\text{magnitude } v, \text{ direction } \theta]$, orientation $\Psi$

compute motion direction
compute motion velocity
Definition of the System

- Kinematic movement

Output of system:
velocity $V = [V_x, V_y] = [\text{magnitude } v, \text{ direction } \theta]$, orientation $\Psi$

What $\Psi$ should be set to?
Definition of the System

• Kinematic movement

Output of system:
velocity $V = [V_x, V_y] = [\text{magnitude } v, \text{ direction } \theta], \text{ orientation } \Psi$

Any problems with this approach?
Definition of the System

- Kinematic movement

Output of system:
velocity $V = [Vx,Vy] = [\text{magnitude } v, \text{ direction } \theta], \text{ orientation } \Psi$

Any solutions?

- compute motion direction
- compute motion velocity
Definition of the System

• Kinematic movement

Output of system:
velocity $V = [Vx, Vy] = [\text{magnitude } v, \text{ direction } \theta], \text{ orientation } \Psi$

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Definition of the System

- Kinematic movement

Output of system:
velocity \( V = [V_x, V_y] = [\text{magnitude } v, \text{ direction } \theta] \),
orientation \( \Psi \)

Output of the overall system: \( V, \Psi \)
Definition of the System

- Dynamic movement

State of system:
position \( P = [x, z] \),
velocity \( V = [V_x, V_y] = [\text{magnitude } v, \text{ direction } \theta] \),
orientation \( \Psi \),
angular speed \( d\Psi \)
Definition of the System

• Dynamic movement

Output of system: acceleration $A = [Ax, Az]$, angular acceleration $dd\Psi$
Definition of the System

• Dynamic movement

Output of system: acceleration $A = [Ax, Az]$, angular acceleration $dd\Psi$

Also limit $V$, $d\Psi$ to their max values

Output of the overall system: $V$, $\Psi$
Position Update According to Motion Equation

- Continuous formulation for constant acceleration:

\[
\begin{align*}
    a &= \frac{dv}{dt} \\
    \int_0^t \! a \, dt &= \int_0^t \! \frac{dv}{dt} \, dt \\
    at &= v(t) - v_0 \\
    v(t) &= v_0 + at
\end{align*}
\]

compute motion velocity

\[
\begin{align*}
    v &= \frac{dP}{dt} \\
    \int_0^t \! v \, dt &= \int_0^t \! \frac{dP}{dt} \, dt \\
    \int_0^t (v_0 + at) \, dt &= P(t) - P_0 \\
    v_0 t + \frac{1}{2} at^2 &= P(t) - P_0 \\
    P(t) &= P_0 + v_0 t + \frac{1}{2} at^2
\end{align*}
\]

simulate motion and update position/velocity
Position Update According to Motion Equation

- Continuous formulation for constant acceleration:

\[ a = \frac{dv}{dt} \]

\[ \int_{0}^{t} at \, dt = \int_{0}^{t} \frac{dv}{dt} \, dt \]

\[ at = v(t) - v_0 \]

\[ v(t) = v_0 + at \]

\[ v = \frac{dP}{dt} \]

same derivation for orientation \( \Psi \) and angular speed \( d\Psi \)

\[ \int_{0}^{t} (v_0 + at) \, dt = P(t) - P_0 \]

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

\[ P(t) = P_0 + v_0 t + \frac{1}{2} at^2 \]
Position Update According to Motion Equation

• Discrete formulation:

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

\[ d\Psi[t+1] = d\Psi[t] + dd\Psi[t] \Delta t \]
\[ \Psi[t+1] = \Psi[t] + d\Psi[t] \Delta t + 0.5 dd\Psi[t] \Delta t^2 \]
Position Update According to Motion Equation

- Discrete formulation simplified:

\[
\begin{align*}
V[t+1] &= V[t] + A[t] \Delta t \\
P[t+1] &= P[t] + V[t] \Delta t \\
d\Psi[t+1] &= d\Psi[t] + dd\Psi[t] \Delta t \\
\Psi[t+1] &= \Psi[t] + d\Psi[t] \Delta t
\end{align*}
\]

Why can we do it?
Kinematic Seek Behavior

• Move towards a target point $T$
  
  \[ V = [v, \theta] = [\text{max. speed}, \lambda] \]
  
  \[ \Psi = \lambda \]
Kinematic Flee Behavior

- Move away from a target point $T$
  \[ V = [v, \theta] = [\text{max. speed}, \lambda + \pi] \]
  \[ \Psi = \lambda + \pi \]
Kinematic Seek Behavior

• Move towards a target point $T$
  \[ V = [v, \theta] = [\text{max. speed}, \lambda] \]
  \[ \Psi = \lambda \]
Kinematic Arrival Behavior

- Approach a target point $T$
  
  \[ V = [v, \theta] = [0, \lambda] \]
  \[ \Psi = \lambda \]

  else
  
  \[ V = [v, \theta] = [K*d, \lambda] \text{ for some constant } K \]
  \[ \Psi = \lambda \]

\[ P \] $\rightarrow$ $T$

$d$

$\lambda$
Kinematic Wander Behavior

• Random wandering
  
  \[ d\Psi = \text{random with bias towards} \ 0 \]
  
  \[ \Psi = \Psi + d\Psi \]

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

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
Dynamic Seek Behavior

• Move towards a target point $T$

$$A = \text{max. accel} \times \text{normalize}([dx,dz])$$

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

Take care for angle wrapping

compute motion direction
Dynamic Seek Behavior

- Move towards a target point $T$
  
  $$A = \text{max. accel} \times \text{normalize}([dx, dz])$$

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

  **Compute motion direction**

  **How do we do flee?**

  **Take care for angle wrapping**
Dynamic Flee Behavior

• Move away from a target point \( T \)
  \[
  A = -\text{max. accel} \times \text{normalize}([dx, dz])
  \]
  \[
  dd\Psi = K(\Psi - \lambda - \pi) \text{ limited by max. angular acceleration}
  \]
Dynamic Flee Behavior

• Move away from a target point $T$

\[ A = -\text{max. accel} \times \text{normalize}([dx,dz]) \]

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

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Dynamic Flee Behavior

• Move away from a target point $T$
  
  $A = -\text{max. accel} \times \text{normalize}([dx,dz])$

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

http://www.red3d.com/cwr/steer/SeekFlee.html
Dynamic Arrive Behavior

• Approach a target point $T$
  
  $\text{if } d < \text{arrival radius then } v_{\text{des}} = 0$
  
  $\text{else } v_{\text{des}} = K \cdot d \text{ for some constant } K$
  
  $A = K_1 \cdot (v_{\text{des}} - v_{\text{current}}) \cdot \text{normalize}([dx, dz])$
  
  $dd\psi = K(\psi - \lambda - \pi) \text{ limited by max. angular acceleration}$

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

Meaning of $T$?
Dynamic Arrive Behavior

• Approach a target point $T$

  if $d < \text{arrival radius}$ then $v_{des} = 0$
  else $v_{des} = K \cdot d$ for some constant $K$

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

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

from “Artificial Intelligence for Games” by I. Millington & J. Funge
Dynamic Arrive Behavior

- Approach a target point $T$

  \[
  \text{if } d < \text{arrival radius then } v_{des} = 0 \\
  \text{else } v_{des} = K \cdot d \text{ for some constant } K \\
  \]

  \[
  A = K_1 \cdot (v_{des} - v_{current}) \cdot \text{normalize}([dx, dz]) \\
  \]

  \[
  d\ddot{\Psi} = K(\Psi - \lambda - \pi) \text{ limited by max. angular acceleration}
  \]

\[http://www.red3d.com/cwr/steer/Arrival.html\]
Dynamic Arrive Behavior

- Approach a target point $T$

  \[
  \text{if } d < \text{arrival radius then } v_{\text{des}} = 0 \\
  \text{else } v_{\text{des}} = K \cdot d \text{ for some constant } K \\
  A = K_1 \cdot (v_{\text{des}} - v_{\text{current}}) \cdot \text{normalize}([dx, dz]) \\
  \dd \Psi = K(\Psi - \lambda - \pi) \text{ limited by max. angular acceleration}
  \]

How would you implement velocity matching behavior?
Dynamic Pursue Behavior

- Pursue a moving target $T$ that isn’t close

Where will seek direct the character?

What’s the problem?

Solutions?

compute motion direction

Where will seek direct the character?

What’s the problem?

Solutions?

\[
\begin{align*}
&P \hspace{1cm} T \\
&dx \hspace{1cm} dz \\
&\lambda
\end{align*}
\]
Dynamic Pursue Behavior

• Pursue a moving target $T$ that isn’t close

seek with target position at:

$$\frac{d}{v_{\text{current}}} * V_T$$

Where will pursue direct the character?
Dynamic Evade Behavior

- Evade a moving target $T$ that isn’t close

  - *flee with target position at:*

  $$d/v_{current} * V_T$$
Dynamic Evade Behavior

• Evade a moving target $T$ that isn’t close
  
  *flee with target position at:*
  
  $d/v_{\text{current}} * V_T$

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