Crowd simulation
Animating Crowds

- We have been going through methods to simulate individual characters
- What if we want to simulate the movement of crowds?
  - Pedestrians in the streets
  - Flock of birds, school of fishes
  - People in panic
Why need a crowd simulation?

- Shooting scenes with many people
  - Expensive to hire many extra actors
  - We don’t see the details: use simple computer-based models

- Security reasons:
  - Need to evaluate the safety of buildings
Overview

- Agent based Methods
  - Flocking
  - Intention generator
  - Creating scenes with human crowds
  - Patch-based methods
    - Create scenes by building blocks
  - (Local) Collision avoidance
    - Explicit model
    - Velocity obstacle, reciprocal velocity obstacle
- Global methods (Continuum crowds)
Flocking Models – Most basic agent model (Reynolds ’87)

- The agents interact based on simple dynamics
- Good to simulate
  - Flock of birds flying,
  - School of fishes swimming
Forces applied to individuals

- Separation
- Alignment
- Cohesion
- Avoidance
Separation

- Try to avoid running into local flock-mates

\[ F_s = \sum_{i \in S} \left( \frac{x - x_i}{\|x - x_i\|^2} \right) \]

- \( S \): surrounding entities

Separation:
Fly away away from neighbors that are “too close”
Alignment

- Try to fly in the same direction as local flock-mates
  - Gets everyone flying in the same direction

\[
F_A = \sum_{i \in S} (\dot{x}_i - \dot{x}).
\]

Alignment: steer toward average velocity
Cohesion

- Try to move toward the average position of local flock-mates
  - Spaces everyone out evenly,
  - keep boundary members toward the group

\[ F_C = \sum_{i \in S} \frac{x_i}{N} - x \]
Avoidance

- Try to avoid obstacles
  - The closer the character is, the stronger the force

\[ F_o = \frac{N}{D(O, x)} \]

\( N \): the normal vector of the obstacle at the point that is closest to \( x \)
Perception Volume

- Simulating the sight of the entities
Combining Commands

- Consider commands as accelerations
- Give a weight to each desire

$$w_S, w_C, w_A, w_O$$

- Ex. High for avoidance, low for cohesion

Simply summing the weighted sum may cause issues
Combining Commands

- Option: Apply in order of highest weight, until a max (absolute sum of) acceleration is reached
  - Ensures that high priority things happen

\[ w_O \gg w_S, w_C, w_A, \]

http://www.youtube.com/watch?v=4kPSDW7gQHA
Another application of flocking:
Simulating dynamical features of escape panic: Helbing ‘00

- A model to simulate the crowd under panic

\[
m_i \frac{d\vec{v}_i}{dt} = m_i \frac{\vec{v}_i^o(t) \vec{e}_i^o(t) - \vec{v}_i(t)}{\tau_i} + \sum_{j \neq i} \vec{f}_{ij} + \sum \vec{f}_{iw}
\]

Figure 1: An image of Helbing’s Model Simulation.

http://www.youtube.com/watch?v=mUKYtOXg5Zw
Simulating dynamical features of escape panic: Helbing ’00

- $f_{ij}$, the force to stay away from other entities,
- $f_{iw}$ the force to stay away from walls
- $v_i$: velocity of entity $i$
- $v_i^o e_i^o$: the desired velocity of entity $i$

$$m_i \frac{d\vec{v}_i}{dt} = m_i \frac{\vec{v}_i^o(t) \vec{e}_i^o(t) - \vec{v}_i(t)}{\tau_i} + \sum_{j \neq i} f_{ij} + \sum f_{iw}$$

Figure 1: An image of Helbing’s Model Simulation.
Simulating dynamical features of escape panic: Helbing ‘00

- $f_{ij}$, the force to stay away from other entities,
- $f_{iw}$ the force to stay away from walls
- $v_i$: velocity of entity $i$
- $v_i^o e_i^o$: the desired velocity of entity $i$

$$m_i \frac{d\vec{v}_i}{dt} = m_i \frac{\vec{v}_i^o (t) \vec{e}_i^o (t) - \vec{v}_i (t)}{\tau_i} + \sum_{j \neq i} f_{ij} + \sum f_{iw}$$

Figure 1: An image of Helbing’s Model Simulation.
Simulating dynamical features of escape panic: Helbing ‘00

- $f_{ij}$, the force to stay away from other entities,
- $f_{iw}$ the force to stay away from walls
- $v_i$: velocity of entity $i$
- $v_i^o e_i^o$: the desired velocity of entity $i$

$$m_i \frac{d\vec{v}_i}{dt} = m_i \frac{\vec{v}_i^o(t)\vec{e}_i^o(t) - \vec{v}_i(t)}{\tau_i} + \sum_{j \neq i} f_{ij} + \sum f_{iw}$$

Figure 1: An image of Helbing’s Model Simulation.
Flocking Evaluation

- **Advantages:**
  - Complex behaviour and phenomena from simple rules
  - Many types of behaviour can be expressed with different rules and parameters

- **Disadvantages:**
  - Can be difficult to set parameters to achieve desired result
  - Problems regarding strength of forces
Overview

- Agent based Methods
  - Flocking
  - Intention generator
  - Creating scenes with human crowds
  - Patch-based methods
    - Create scenes by building blocks
  - (Local) Collision avoidance
    - Explicit model
    - velocity obstacle, reciprocal velocity obstacle
- Global methods (Continuum crowds)
Making the Agents Smarter

- Designing the details of the behaviours
  - What to do when you see a predator
  - What to do when you find somebody you know?
  - What to do when the traffic lights are red?
- Intention generator
Artificial Fish
- Terzopoulos et al.
SIGGRAPH ‘94

- Considering attributes such as
  - Hunger
  - Libido
  - Fear
for deciding the behaviour of the fish

- Adding sensory perception such as
  - vision
Intention Generator

1. First check the sensory information for collision
2. If any close predator, either
   • School
   • Escape
Otherwise if hungry eat
If full mate

http://www.youtube.com/watch?v=RHt_8ZYQVZw
Intention Generator

- Decision making for the fish
What about humans?

- Can do something similar
- But humans are a bit more high level
  - Not only eating or escaping
  - Have destinations
  - Contexts
  - Socializing
    - Grouping,
    - Talking to somebody you know

A more complex model is needed
Behavior model: Autonomous Pedestrians

- More complex than a fish

http://www.youtube.com/watch?v=cqG7ADSvQ5o
More serious problems for humans: context

- Interactions with the environment
- Collision avoidance
  - More complex than fishes
- Getting to the destination:
  - Use path-finding algorithms such as A* search (shortest route to the destination)
Interactions with the Environment

Examples
- At bus stops, people stop and queue
- At an elevator, people wait for it and ride on it when it comes and travel to different floors
- TV in the living room: turn on the TV and sit on the sofa to watch TV
Interactions with the Environment

■ Examples
- At bus stops, people stop and queue
- At an elevator, people wait for it and ride on it when it comes and travel to different floors
- TV in the living room: turn on the TV and sit on the sofa to watch TV

■ Preparing an intention generator to interact with all these objects is not so easy
Interactions with the Environment

- **Examples**
  - At bus stops, people stop and queue
  - At an elevator, people wait for it and ride on it when it comes and travel to different floors
  - TV in the living room: turn on the TV and sit on the sofa to watch TV

- Preparing an intention generator to interact with all these objects is not so easy

- **Need various object / motion data**
Interactions with the Environment

- **Examples**
  - At bus stops, people stop and queue
  - At an elevator, people wait for it and ride on it when it comes and travel to different floors
  - TV in the living room: turn on the TV and sit on the sofa to watch TV

- Preparing an intention generator to interact with all these objects is not so easy

- Need various motion/object data

- **IDEA:** associate such motions with the objects
  - Once the character comes across such objects, they launch the associated motion
Embedding motions into the environment:
Advantages of the approach

- **Efficient data handling**
  - Each agent holds the data only needed at that moment

- **The control is decentralized**
  - The system is scalable, and large crowds can be simulated
Patch-based approaches

- Pre-compute the patches (building blocks) which include the characters and the environment
- Concatenate them to generate large scale scenes during runtime
- Motion Patches
- Crowd patches
Motion Patches

- Building Blocks for Virtual Environments
  - Embed the motions into the environment
  - The patches are spatially aligned
Crowd Patches

- A patch-based approach to generate scenes of crowded environment
- Crowds avoiding each other in the patches
- The timing and location the characters entering the patches are fixed so that the people can keep on entering / exiting
- The characters can be switched to make more variations
Overview

- Agent based Methods
  - Flocking
  - Intention generator
  - Creating scenes with human crowds
  - Patch-based methods
    - Create scenes by building blocks
  - (Local) Collision avoidance
  - Explicit models,
  - Velocity obstacle, reciprocal velocity obstacle
- Global methods (Continuum crowds)
Collision avoidance

- Humans avoid others in streets in a complex way
- Sometimes wait, sometimes, move aside while walking
- Need to either
  - Model them based on a complex collision avoidance engine
  - Find a good mathematical model
Avoid oncoming pedestrians

- If a cross collision is estimated by H with C
  - If H is arriving slightly earlier, it accelerates and turn away from C
  - If H is arriving slightly later, it decelerates and turn towards C
  - So C does the same
Head-on collision

- If a head-on collision is estimated, the agents turn away from each other
Safe turning

• If an agent needs to make a quick turn, the curvature of the turn is gradually increased until a collision free turn is found

• The velocity is decreased according to the curvature
A Mathematical Model: Velocity Obstacle

The set of all velocities of an agent that will result in a collision, assuming the other maintains its current velocity.

\[ B \oplus -A \]

\[ VO_B^A(v_B) \]

\[ v_A - v_B \]

\[ (v_B) \]

\[ (-v_B) \]
Velocity Obstacle (2)

Strategy: Select a velocity that is closest to the desired velocity but outside the VO
Velocity Obstacle (3)

Good for collision avoidance, but can result in oscillation when both agents use the same strategy.

http://youtu.be/BNTNpYTw3_s
Reciprocal Velocity Obstacle (RVO)

Idea: instead of choosing a new velocity outside the velocity obstacle, take the *average* of a velocity outside the velocity obstacle and the current velocity.
Reciprocal Velocity Obstacle (RVO)

Idea: instead of choosing a new velocity outside the velocity obstacle, take the average of a velocity outside the velocity obstacle and the current velocity

http://youtu.be/9O-YkaiBVXw
Reciprocal Velocity Obstacle (RVO)

Idea: instead of choosing a new velocity outside the velocity obstacle, take the \textit{average} of a velocity outside the velocity obstacle and the current velocity.

\[ \text{RVO}_B^A(v_B, v_A) = \{v'_A | 2v'_A - v_A \in \text{VO}_B^A(v_B)\} \]

http://youtu.be/9O-YkaiBVXw
Reciprocal Velocity Obstacle (RVO) (2)

- No oscillation
- No global communication needed between the agents
- Simple and can handle multiple agents
  - Calculate the RVO with all the agents in the neighbours and select a velocity outside all the RVOs

http://youtu.be/soHH-ocT1V8
http://youtu.be/nZ4mVCZRD0E
Collision Avoidance: Local vs. Global

- Sometimes you need to consider more about the future
  - Taking a shortest path
  - Plan to avoid all the obstacles ahead in advance
  - For all the agents

→ Global Model
Continuum Crowds

- Solving the path-planning and collision avoidance at the same time
  1. Compute the potential field at every time step
     - Based on the other avatars and obstacles
     - The goal location
  1. The character’s movement determined based on the potential field
  2. Update the potential field
Continuum Crowds:

procedure

- Discretize the space into grids
- Decide the start / goal of the characters
Continuum Crowds: procedure 2

- The cost to the goal is going to be computed by the following function:

\[
\alpha \int_P 1 ds + \beta \int_P \frac{1}{f} ds + \gamma \int_P \frac{g}{f} ds
\]

- Path Length
- Time
- Discomfort

\[
\int_P C ds, \quad \text{where} \quad C \equiv \frac{\alpha f + \beta + \gamma g}{f}
\]
Continuum Crowds: procedure 2

- The cost to the goal is going to be computed by the following function:

\[ \alpha \int_P 1 \, ds + \beta \int_P \frac{1}{f} \, ds + \gamma \int_P \frac{g}{f} \, ds \]

\[ \int_P C \, ds, \quad \text{where} \quad C \equiv \frac{\alpha f + \beta + \gamma g}{f} \]
Discomfort Field

- Produced by other obstacles / characters

\[ \alpha \int_P 1 ds + \beta \int_P \frac{1}{f} ds + \gamma \int_P \frac{g}{f} ds \]

- Path Length
- Time
- Discomfort
Continuum Crowds: procedure 1

\[ \alpha \int_P 1 ds + \beta \int_P \frac{1}{f} ds + \gamma \int_P \frac{g}{f} ds \]

Path Length  | Time  | Discomfort

\[ \int_P C ds, \quad \text{where} \quad C \equiv \frac{\alpha f + \beta + \gamma g}{f} \]

- Potential field
- Finding the path $P$ with minimal $\int_P C ds$ from all the grid points
Eikonal function

As $\phi$ is the field based on the optimal cost, the gradient of the field will be the cost at that location.

$$\phi(x) = \int_P C \, ds$$

$$\|\nabla \phi(x)\| = C,$$
Continuum Crowds: procedure 2

- Starting from the goal we expand outwards and accumulate the cost $C$
  - *Fast Marching Method*
  - Similar to Djikstra’s algorithm
Fast Marching Method

Initialize Step: Put the goal in the Alive Points
Alive Points (A): The set of points we know the cost ($T_{i,j}$)
Narrow Band: grid points neighbors of A.
Far Away Points: all others grid points \{i,j\}. Set the cost ($T_{i,j}$) $\infty$ for all points in FarAway.

![Diagram of Fast Marching Method](image)
Marching Forwards

1. Begin Loop: Let $(i_{\text{min}}, j_{\text{min}})$ be the point in NarrowBand with the smallest value for $\emptyset$.

2. Add the point $(i_{\text{min}}, j_{\text{min}})$ to $A$; remove it from NarrowBand.

3. Tag as neighbors any points $(i_{\text{min}}-1, j_{\text{min}})$, $(i_{\text{min}}+1, j_{\text{min}})$, $(i_{\text{min}}, j_{\text{min}}-1)$, $(i_{\text{min}}, j_{\text{min}}+1)$ that are either in NarrowBand or FarAway. If the neighbor is in FarAway, remove it from that list and add it to the set NarrowBand.

4. Recompute the values of $\emptyset$ at all neighbors.

Return to top of Loop.
Computing the values of $\phi$

- Use the Eikonal equation

$$||\nabla \phi(x)|| = C,$$

- Among the adjacent grid cells, we first find those with less cost

$$m_x = \arg \min_{i \in \{W,E\}} \{ \phi_i + C_{M \rightarrow i} \} \quad m_y = \arg \min_{i \in \{N,S\}} \{ \phi_i + C_{M \rightarrow i} \}.$$
Computing the values of $\phi$ (2)

- Compute $\Phi$ by solving the Eikonal function with finite difference

$$\| \nabla \phi(x) \| = C,$$

$$\left( \frac{\phi_M - \phi_{m_x}}{C_{M\rightarrow m_x}} \right)^2 + \left( \frac{\phi_M - \phi_{m_y}}{C_{M\rightarrow m_y}} \right)^2 = 1.$$
If the group of people share the same speed, goal, and discomfort, we can use the same potential field for all these characters.

- Very efficient if there are little number of groups
Can simulate the dynamics of crowds

- A global approach (optimal towards the goal)
- Can simulate phenomena observed in real humans

http://www.youtube.com/watch?v=lqJuVhDFSp8
Summary and Discussions

- **Two streams**
  - Agent-based
  - Global methods

- **Applications**
  - Animation, games
  - Evacuation analysis

- **Collision avoidance is important not only for character animation but also for robotics**
Reference

- Kang Hoon Lee, Myung Geol Choi and Jehee Lee, Motion Patches: Building Blocks for Virtual Environments Annotated with Motion Data, accepted to ACM SIGGRAPH 2006.
- Adrien Treuille, Seth Cooper, Zoran Popovic, Continuum Crowds, SIGGRAPH 2005