Modeling Individualities in Groups and Crowds

Crowd Variety

First: meshes, use of various templates

Second: use of textures
Various Textures

Third: Designing Color Variety
- Random colors for each body part not good enough
- Realistic results require constrained randomness

Designing Texture Variety
- HSB Maps
  - HSB space constrained to a 3D color space.
  - Allow to deal with localized constraints.
  - Eyes example:
    - Hue from 20 to 250;
    - Saturation from 30 to 80;
    - Brightness from 40 to 100.

Same texture mapping for different templates; different characters by varying colors
Various Colors

Appearance sets applied to instances of a single human template. Note the different specular effects on the body parts and the varying cloth patterns.

Various colors and textures

Template
Accessories

Accessories: small meshes representing elements that can easily be added to human template original mesh

- From subtle details, like watches, jewelry, or glasses, to larger items, such as hats, wigs, or backpacks.
- Distributing accessories to large crowd of few human templates varies shape of each instance, and thus makes it unique.
- Similarly to deformable meshes, accessories are attached to a skeleton and follow its animation when deformed.

• We assume that accessory vertices all attached to same joint, i.e., mesh not deformed.
• Example: hat would be attached to skull joint.
• Several important steps to model accessory, so that it can later be correctly placed and oriented for all human templates.
• First of all, we identify joint to which accessory should be attached.

Various colors, textures and accessories

Accessory movements (hands in the pocket, phone call, hand on hip...).
By applying variety techniques at 3 levels, instances of same template seem unique.

Movie

Animation Variety
To animate a virtual crowd is it just to animate many Virtual Humans?

No, first, we need spontaneous and flexible movements before emergency situation. After gas leak happened.

Second, we need to define collective behaviors while keeping individualities.

Problems
- Keyframe: too fastidious to use
- Motion Capture: too expensive, no flexibility, hard to do for locomotion
- Motion graphs: too expensive

Model-driven Methods: No general method to model behaviors and actions with flexibility and variety:
- It takes a model for:
  - Walking
  - Grasping
  - Running
  - Jumping
  - Visiting Munich
  - Drinking a beer

And everything is not so simple.
Making Them Walk

- PCA in hierarchical structure of sub-PCA spaces.
- At each level of hierarchy, important parameter (personification, type of motion, speed) extracted and related function elaborated, allowing not only motion interpolation but also extrapolation.

Motion modeling using PCAs and time-warping

Motion correction

Adaptive motion control

Integrated Walking/Running engine

Normalization and time warping

- Murray: all leg relative angles in sagittal plane (hip, knee, ankle) show similar trajectories for all adult men for same value of normalized speed $V$

$$V = \frac{V}{H}$$

$V$: walking velocity  
$H$: hip joint height

- Time warp using Inman law: links normalized speed to cycle frequency

$$f = 0.743 \sqrt{V}$$

Animation engine able to continuously vary speed and compute phase variation $\Delta \phi$ for given duration $\Delta t$: $\Delta \phi = \frac{\lambda}{\text{cycleDuration}} = \Delta t \ f$

Motion Correction

- Footplant enforcement
- Numerical IK with priorities [Baerlocher04, Callenec04]

- 1 $\rightarrow$ Ankle
- 2 $\rightarrow$ Toe

IK used to prevent feet sliding by exploiting predictive capability of model.

- Yellow: current posture
- Green: anticipated posture with IK

Footplant enforcement
- Numerical IK with priorities [Baerlocher04, Callenec04]

- 1 $\rightarrow$ Ankle
- 2 $\rightarrow$ Toe

IK used to prevent feet sliding by exploiting predictive capability of model.

- Yellow: current posture
- Green: anticipated posture with IK
• normalize generated motion
• may animate human with large range of different sizes.
• complete locomotion engine, transitions between different motions handled.

Personification weights: 5 people, different in height and gait captured while walking and running
  • allows user to choose how wishes to parametrize these different styles.
• Speed: the five subjects have been captured at many different speeds.
  • allows to choose at which velocity walk/run cycle should be generated.
• Locomotion weights: defines whether cycle is walk or run animation.

A motion kit holds several items:
  • Name, identifying what sort of animation it represents, e.g., walk_1.5m/s,
  • Type, determined by 4 identifiers: action, subaction, left arm action, and right arm action,
  • Link to skeletal animation,
  • Link to rigid animation,
  • Link to impostor animation.

Example of motion kit structure. On the left, a virtual human instantiated from a human template points to the motion kit it currently uses. In the center, a motion kit with its links identifying the corresponding animations to use for all human templates and LOD.
Varied locomotion cycle created in two passes:
1) generates primary cycle with default upper body animation.
2) adds secondary upper-body motion using prioritized IK solver [Baerlocher-Boulic 2004].
Tool allows to enforce several constraints at same time, with levels of priority if necessary.
To solve IK problem, following inputs needed:
- Original locomotion cycle,
- Hand-designed “first guess” posture of hand and arm, using MotionBuilder [Autodesk, 2009b].
- Set of constraints to apply to hand and/or arm.

E.g. three colored cubes on hand have to be positioned as close as possible to three corresponding colored cubes attached to pelvis.
Set of controlled effectors attached to the hand and corresponding goal positions attached to the pelvis.

locomotion cycle augmented with right hand in pocket.

Crowd Rendering
• dynamic meshes in red,
• static meshes in green
• Impostors in blue.

Apparent levels of detail; in red: the rigid meshes, in green: the impostors

Rendering pipeline (4 steps)
• First step **culling** which determines visibility and rendering
• representation or **fidelity** for each simulated human
• culling not done on each individual but at node level,
  => determine fidelities for set of characters at once.
Second step: rendering of dynamic meshes
- most detailed \textit{fidelity} and can play back interpolated animations based on skeletal postures.
- Animation constructed based on walking style of human and its speed
- Hardware vertex shader and fragment shader used to deform and render human on GPU.

Deformed Meshes
- Algorithm: Skeleton Subspace Deformation (SSD)
- each vertex deformed by weighted transformation of one or several attached bones or joints.
\[ \gamma(t) = \sum_{i=1}^{n} \lambda_i X_{i}^{\text{ref}} v_{i}^{\text{ref}} \]
- where \( \gamma(t) \): deformed vertex at time \( t \),
- \( \lambda_i \): \text{global transform of bone } i \text{ at time } t,
- \( X_{i}^{\text{ref}} \): inverse \text{global transform of bone in reference position}
- \( v_{i}^{\text{ref}} \): vertex in reference position.

LODs
- 5'000 polygons
- 1'000 polygons
- 100 polygons

Third step: Static meshes
- keeps pre-transformed set of animations usually in range of 2 or 3 animations, using lowest resolution mesh of deformed ones in the previous step.
- By pre-computing deformations substantial gains in speed achieved
Forth step: impostors

- Assumption: camera will never be directly above crowd.
- Only sample images at waist level of character needed.
- Each template sampled at 20 different angles, for each of 25 key-frames composing walk animation.

Accessories

- Similarly to deformable meshes, accessories attached to skeleton and follow its animation when deformed.
- We assume that accessory vertices all attached to same joint, i.e., mesh not deformed.
- Example: hat attached to skull joint.
- Several important steps to model accessory, so that it can later be correctly placed and oriented for all human templates.
- First of all, we identify joint to which accessory should be attached.
Second phase: accessory transformed, to perfectly coincide with mesh of each human template.
Changes expressed relatively to attach joint as 4×4 transformation matrix $T_{accessory}$.

$$T = \begin{pmatrix} R & 1 \\ 0 & 1 \end{pmatrix} = T_{char} T_{joint} T_{accessory}.$$ $T_{char}$: character transformation matrix in world coordinates
$T_{joint}$: attach joint deformation matrix, relative to $T_{char}$.

For deformable meshes, Equation directly computed at runtime.
For rigid meshes and impostors, dedicated methods to easily place accessories.

### Accessories for Rigid Meshes
- Main advantage: no dynamic deformation of skeleton at runtime.
- However, implies matrix $T_{joint}$ in Eq. 1 not available to place accessory.
- Naive approach: store accessory animation as for rigid mesh.
- However, we take advantage from assumption that mesh never deformed.
- Every time rigid mesh keyframe stored, matrices $T_{joint}$ to which accessories attached also saved.

### Accessories for Impostors
- Accessories for impostors generated in preprocess: image tiles sampled all around object in orthographic mode and saved in 512 × 512 normal and uv maps.
- With this method, samples are more uniformly distributed.
- Memory storage cost for accessory impostor is constant and independent from number of keyframes generated for human impostor.
- More precisely, we store one uv map and one normal map per accessory.
Runtime pipeline to place accessory impostors

Five steps:
- Step 1: Compute the accessory transformation matrix $T$ with Equation.
- Step 2: Retrieve normal and uv map tile, representing the accessory from correct point of view.
- Step 3: Compute exact position of accessory impostor, i.e., the quad.
- Step 4: Rotate quad around camera $Z$ axis so that it is correctly oriented.
- Step 5: Compute depth of each impostor pixel (or fragment) to avoid visual artifacts

- Since character wears hat inclined backwards, quad has to be rotated to correctly imitate 3D hat orientation.
- Main problem when addressing such an issue with a perspective projection is that depth values in the $Z$ buffer do not vary linearly.
- Our dedicated algorithm solves visibility problems inherent in impostors, while allowing the use of any projection.
Hybrid architecture

- to handle path planning of thousands of pedestrians in real time, while ensuring dynamic collision avoidance.
- scalability of our approach allows to interactively create and distribute regions of varied interest, where motion planning ruled by different algorithms.

Practically, regions of high interest are governed by a long-term potential field-based approach, while other zones exploit graph of environment and short-term avoidance techniques.

- Combination of
  - Potential field approach (Treuille et al. SIGGRAPH 2006, Continuum crowds)

Comparison between our approach and our implementation of purely potential field-based approach of Treuille et al. for varying number of groups. Each group is composed of 100 pedestrians.

Observation of interesting emergent behaviors, e.g., lane formations or panic effects, => crowd motion planning more realistic
**YAQ system**

Virtual population created by assembling such blocks.
- As interactions solved once for each block, and further reused during simulation, we save precious computation time.
- Assembling and disassembling blocks can be achieved at run-time, breaking limitations on environment size: virtual population only generated in front of spectator’s point of view.

---

**Patches**

Illustration of motion patches from [Lee et al., 2006]: patches are building blocks annotated with captured motions that can be combined to generate movements of virtual character.

- Objective: to populate virtual environments by composing small pieces of pre-computed animations:
  - **crowd patches**
  - can contain moving pedestrians, humans executing other tasks, animals, or objects.
  - Inside a patch, animations computed to be cyclic over constant period, and can be seamlessly repeated.
  - Allows animated content to appear in endless motion.
  - Animated objects may cross limits of patch and move to neighbor patch.
  - Trajectories of such objects must then meet common limit conditions to allow going from one patch to another adjacent one.
2 categories of dynamic objects: endogenous and exogenous objects.

- Trajectory of endogenous objects always remains inside geometrical limits of patch for whole period.
- Exogenous objects: trajectory goes out of patch borders at some time, and thus, does not meet periodicity condition.

Creating Worlds using (top) a bottom-up technique, e.g., procedural generation, or (bottom) a top-down approach, starting from a geometrical model of the environment.

A pre-defined city populated with crowd patches, based on a template map.

Frame rate evolution (without rendering) for a varying number of patches to simulate.
Thank you for your attention.

Questions ?…
References (VRlab)