A Conceptual Model for Computer Animation

William Joel, Director
Center for Graphics Research
Western Connecticut State University
Danbury, CT USA

Outline
- Traditional Animation
- Scenes vs. Frames
- Model
- System Design

Traditional Animation
- How was it done?
- How does it differ from computer animation?

Scenes vs. Frames
- Common animation software most often uses a frame-based model
- But traditional animation, and modern animation production, focus on scenes or “shots”
- Therefore, a scene-based model would be more appropriate

Model
- Elements
- Environment
- States
- Frames
- Scenes
- Animations

Elements
- $e_i$
- An “object” in an animation
- Can be represented by a finite set of state variables
Environment

- $E = \{ e_i | 1 \leq i \leq n \}$
- Set of all identified elements
- Generally non-empty, but could be an empty set, $\emptyset$

States

- For each $e \in E$ we associate a non-empty set $S_e$, possibly infinite

Frames

- Let $\{S_e\}_{e \in E}$ be the family of state sets
- Frame $f$ on an environment $E$ is defined as $f \in \prod_{e \in E} S_e$
- and is written $f = \langle s_e | e \in E \rangle$
- or as a function, $f : E \rightarrow \bigcup_{e \in E} S_e$
- where $f(e) \in S_e$
- $F_E$ is the set of all frames for $E$

Scenes

- A scene for an environment $E$ consists of an ascending sequence of reals $I$ and a function $c$ that maps $I$ to $F_E$
- $\langle c, I \rangle$
- When frames are of interest $c(I) = (f_1, ..., f_m)$, where $I = (i_1, ..., i_m)$

Animations

- A sequence of scenes, written $a = (c_1, ..., c_p)$

Object Subdivisions

- Subscene
- Subanimation
- Semienvironment
- Semiframe
- Semiscene
- Coscene
Subscene
- Given a scene \( <c,I> \) based on environment \( E \), a subscene of this scene is the pair \( <c, I_{x,y}> \) where \( I_{x,y} \) is a contiguous subsequence of indices from \( I, (i_x, ..., i_y) \), and \( x \) and \( y \) are frame numbers.
- \( <c, I_{x,y}> \) is also based on \( E \).

Subanimation
- A subanimation \( a_{x,y} \) of an animation \( a = (c_1, ..., c_p) \), is either empty or is a subsequence \( a_{x,y} = (c_{i_x}, ..., c_{i_y}) \) where \( 1 \leq x \leq y \leq p \).

Semienvironment
- A semienvironment \( E' \) of environment \( E \) is a non-empty subset \( E' \subseteq E \).

Semiframe
- A semiframe \( f' \) of frame \( f \) is a restriction of \( f \) to a subset of its environment \( f' = f \mid E' \).

Semiscene
- A semiscene of \( <c,I> \) is a scene \( <c',I> \) based on semienvironment \( E' \subseteq E \), such that 
  \[ c'(i_k) = c(i_k) \mid E' \] for each \( i_k \in I \).

Coscene
- A semiscene that can be computed independently of the remainder of its parent scene.
### Functions
- Restriction
- Composition
- Access
- Iteration
- Re-indexing
- Interpolation
- Concatenation

### Restriction
- Represented by the function $\text{restrict}(\ )$
- For example, a restricted function is written $f \mid E'$

### Composition
- Given frames $f'$ and $f''$, corresponding to environments $E'$ and $E''$ respectively
- When $E' \cap E'' = \emptyset$, $f$ is a frame defined on $E' \cup E''$ such that $f = f' \cup f''$

### Iteration
- To iterate a frame function $o$ on two or more scenes that share a common index sequence
  - $<c, l> = \text{iterate}(o, s_1, \ldots, s_k)$
  - where $s_i = <c_i, l>$

### Access
- **Environment**
  - $E(f) = \text{Domain}(f)$
  - $E(<c, l>) = E(c(l))$
- **Scene function**
  - $\langle c, l \rangle = c$
- **Index sequence**
  - $l(c, l) = l$
- **Repeat a frame**
  - $<c', l> = \text{copy}(c(j), l)$
Re-indexing

- Given \( <c, I> \) and \( I' \), where \( |I| = |I'| = n \)
  \( <c', I'> = \text{reindex}(c, I), I' \)
- and
  \( c'(i') = c(i), 1 \leq j \leq n \)

Interpolation

- Given \( <c, I> \) and \( I' \), where \( |I| \neq |I'| \)
  \( <c', I'> = \text{inter}(c, I), I'; F_n \)
- where
  \( c'(i') = F_n(c, I), I' \)
- It is assumed that
  \( c(i) = c'(i), \text{ when } i = j \)

Composition

- Given
  \( \text{compose}(f_1, f_2) = \text{compose}(f_1, f_2, \text{resolve}) \)
- then
  \( <c_3, I> = \text{iterate}(\text{compose}, <c_1, I>, <c_2, I>) \)

Concatenation

- Given \( <c_1, I_1> \) and \( <c_2, I_2> \), where \( i < j \) for all \( i \in I_1 \) and \( j \in I_2 \)
- \( <c, I> = <c_1, I_1> <c_2, I_2> \)
- \( = <c_1 \cup c_2, I_1 \cup I_2> \)

Functions (continued)

- Locate
  \( c(i) = \text{locate}(c, I), \ell, \text{order} \)
- \( \ell(f) \) takes as input a frame and determines whether a condition is true
Take

- $<c, l_{x,y} > = \text{take}(<c, l>, a, b)$
- Where $a < i_x < i_y < b$

Find

- $c(i) = \text{find}(<c, l>, i)$

Fade-in, Fade-out

- $<c', l> = \text{fadein}(<c, l>, x, y, \text{rule})$
- $<c', l> = \text{fadeout}(<c, l>, x, y, \text{rule})$
- Where the rules is expressed as $\text{rule}(\text{pixel}, t)$

Dissolve

- $\text{cross}(<c', l>, <c'', l>, x, y, \text{rule}) = [<c', l> = \text{fadeout}(<c', l>, x, y, \text{rule})][<c', l> = \text{fadein}(<c'', l>, x, y, \text{rule})]$

Generating Functions

- User defined
- Built on existing primatives
- Re-usable
- $\lambda(c, P)$
- Where $\lambda$ is a set of input scenes, and $P$ is a set of additional parameters

Behaviors

- Generating functions
- Restricted to specific elements
- $\lambda(c, P) = \lambda(c, P) \setminus \{e\}$