Johns Hopkins
Engineering for Professionals

# 605.767 Applied Computer Graphics

Brian Russin

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

# Module 12A
# Animation Techniques

# Animation Topics

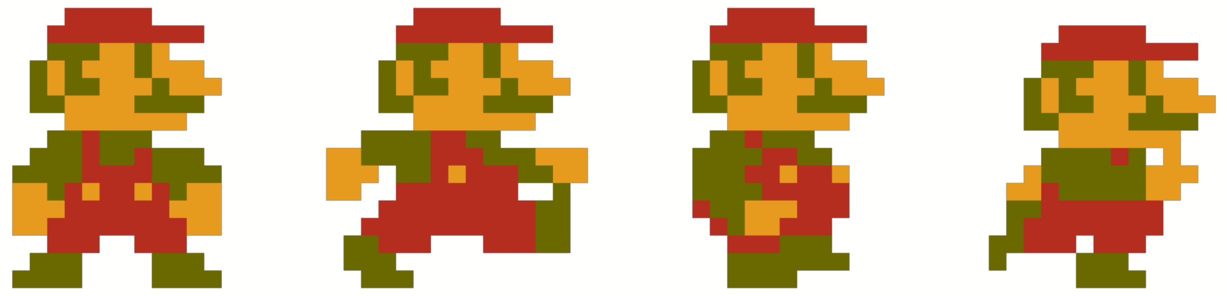
- Animation Types
  - Cel
  - Rigid Hierarchical
  - Per-Vertex, Morph Target
  - Skinned
- Skeleton Principles
  - Bones (Rigs) and Joints
    - Hierarchy
  - Key Frames
- Clips
  - Sequence of frames
  - Local vs. Global Time
    - Synchronizing clips
- Animation Blending
- Other Techniques
  - Application-driven animation
  - User-defined Animation
- Constraints

## Animation Techniques

- Cel Animation
- Rigid Hierarchical Animation
- Per-Vertex and Morph Target Animation
- Skinned Animation

# Cel Animation

- Cel: transparent sheet of plastic on which image can be drawn
  - Used in hand-drawn cartoon animation
  - Series of static images
  - Overlay a background
- Sprite: texture-mapped quad
  - 2D graphics cel equivalent
- Animation
  - Loop - image sequence repeats
  - Clip - traverses image sequence n-times



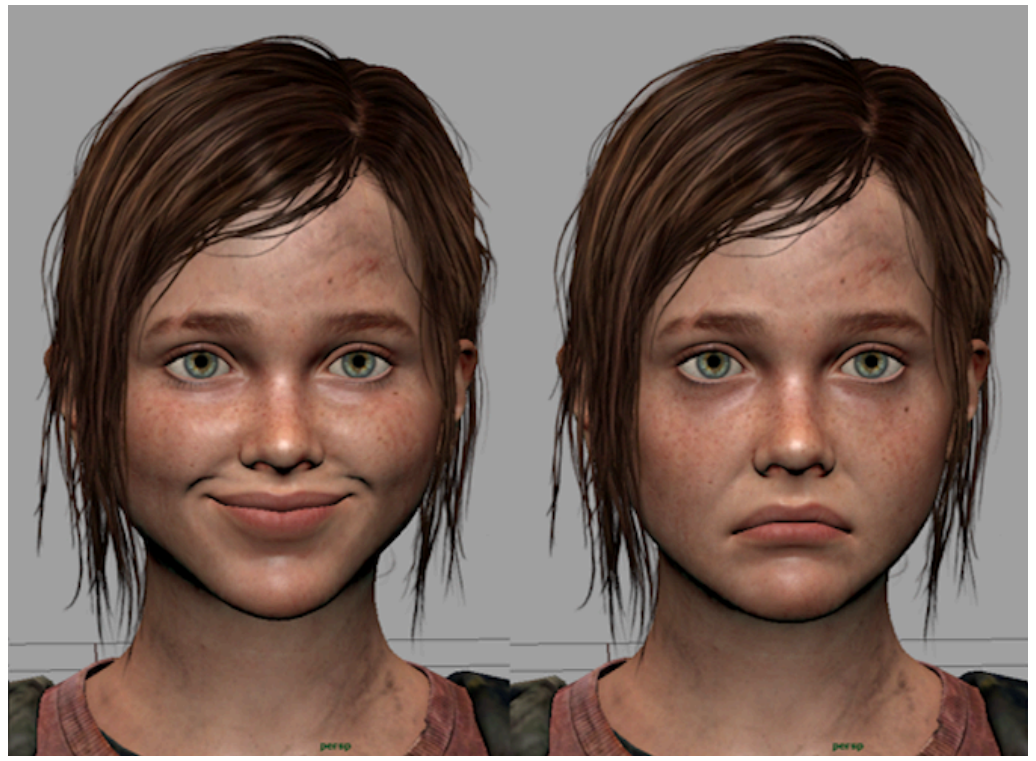
https://www.spriters-resource.com/nes/

## Rigid Hierarchical Animation

- Model as a series of discrete, rigid parts
  - Parts are disjoint
    - Produces cracks between parts
  - Works well for machine-like models
    - Unnatural looking for models of living creatures
- Can be used in 2D and 3D
- Each part has its own transformation
  - Entire model has a transformation hierarchy

# Per-Vertex and Morph Target Animation

- Each vertex has animation data
  - The artist poses each vertex during the animation
  - Most control, but tedious
    - Essential for small, detailed surfaces
      - e.g. facial expressions
    - Impractical for large models
- Discrete poses for vertices called morph targets
  - Extreme or intermediate positions
  - Animation sequence blends or interpolates poses



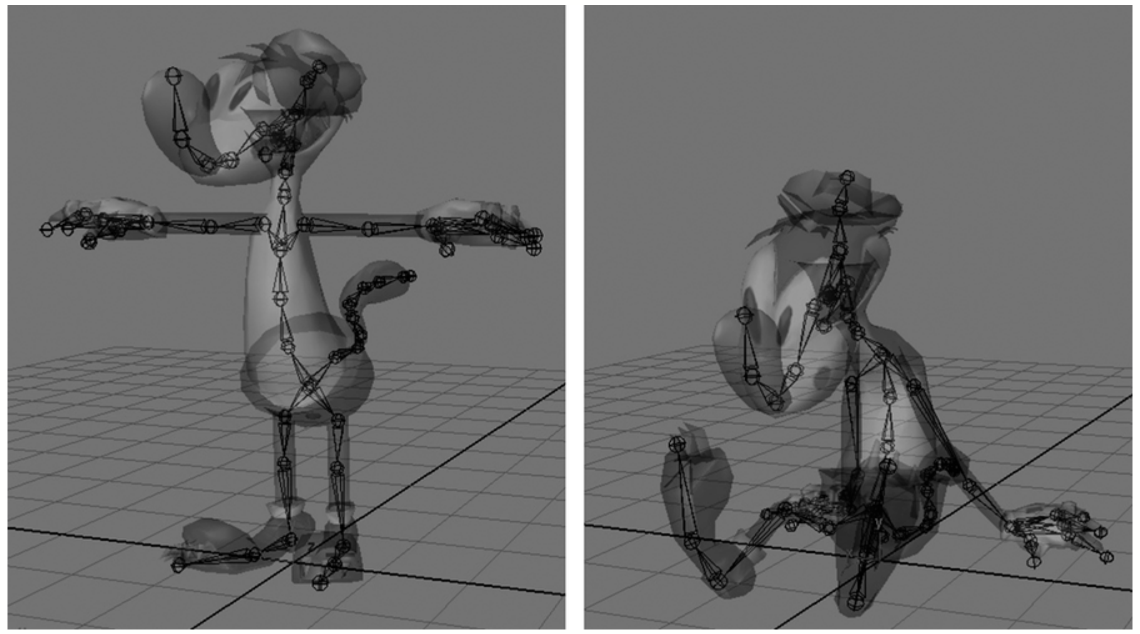
Face targets for The Last of Us
https://www.gameenginebook.com

## Skinned Animation

- Hybrid of Rigid Hierarchy and Per-vertex techniques
  - Hierarchical rigid parts
    - typically called a skeleton or rig
  - Each vertex “assigned” a number of parts
    - Weight per assigned joint
    - Has no absolute position
      - Position is relative to its joints
- Key frames given to underlaying skeleton
- Best balance for flexibility, memory usage, and processing
- Works well with other technologies
  - Motion capture

# Skeleton Rig

- Rigid hierarchy of joints
  - Each joint is a transformation from its parent
    - Has standard SRT data
    - Humanoid primary joint is center of the hips
  - "Bone" is the empty space between joints
- Supported in modeling software
  - Performed by 3D model creators or animators



https://www.gameenginebook.com

## Skeleton Joints

- Can be stored in an array
- Joint information
  - Name
    - string, hash id, index, etc.
  - Index or name of the parent
  - Scale-Rotation-Translation (SRT) data
    - Often stores the inverse transform data
    - Scale
      - Can be constrained to be uniform - represented as single value
      - Or often omitted
    - Rotations should use quaternions for interpolation purposes
- Example: hip->torso->upperArm->lowerArm->hand->finger

## Skeleton Joints (cont.)

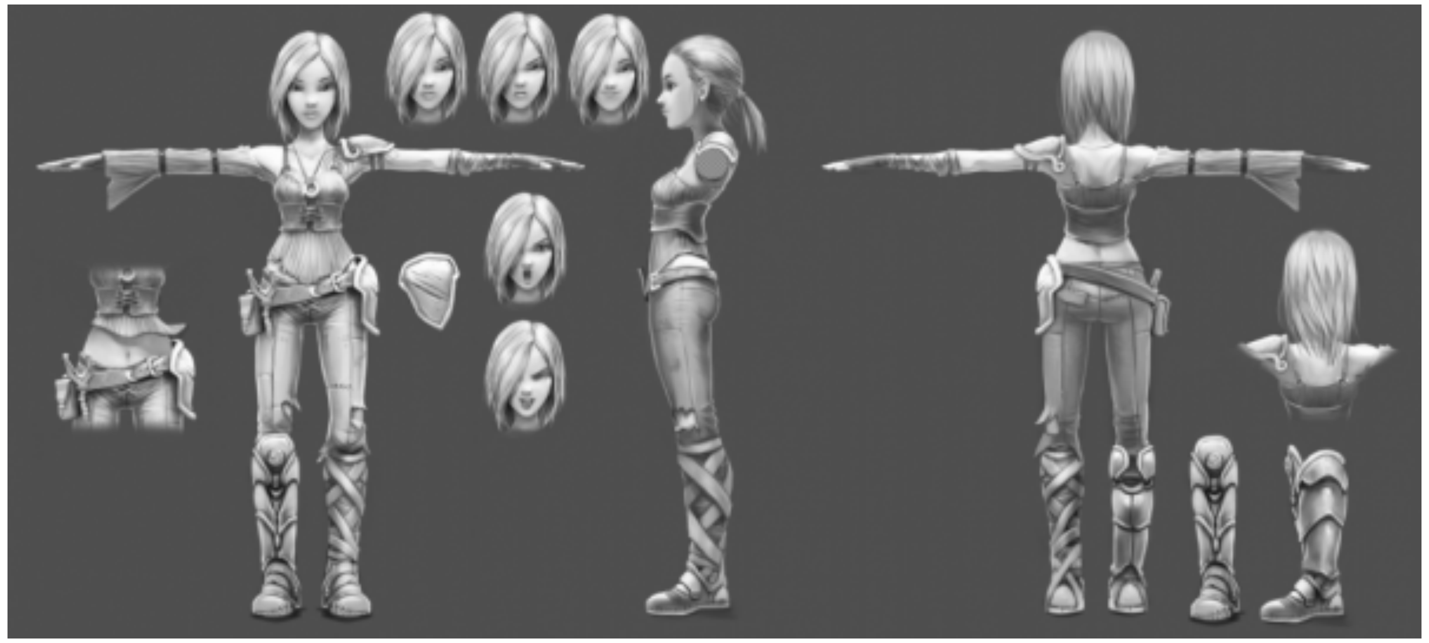
```
struct JointPose { // Uniform Scale
  Quaternion m_rot;   // R
  Vector3    m_trans; // T
  float      m_scale; // S
};

struct JointPose { // Non-uniform Scale
  Quaternion m_rot;   // R
  Vector4    m_trans; // T
  Vector4    m_scale; // S
};

struct SkeletonPose {
  Skeleton  *m_pSkeleton;  // skeleton and num joints
  JointPose *m_aLocalPose; // local joint poses
};
```

## Poses

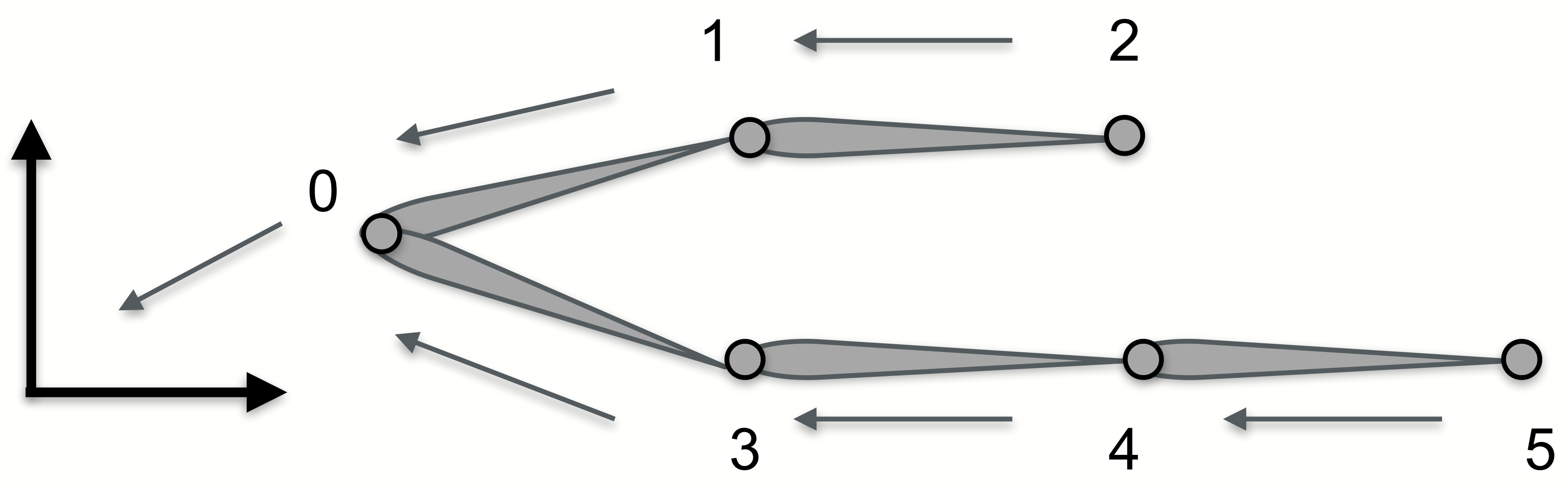
- Bind Pose
  - Primary position
  - Also called the *reference pose*, *rest pose*, or simply the *T-pose*
    - Humanoid character positioned in a *t*-form
  - Used for associating vertices with joints



https://www.animatorisland.com/the-t-pose-all-about-the-mighty-blueprint/

## Poses (cont.)

- Global Pose
  - All joints in world coordinates
- Local Pose
  - Root joint relative to world axes
  - Other joints relative to parent



## Per-Vertex Skin Information

- Each vertex has joint information
  - Associated with 1-to-*n* joints
    - *n*, maximum limit, usually capped at 4
      - 8-bit indices make a 32-bit word
    - Weight for each joints
      - Weights add to 1

```
struct SkinnedVertex {
  float   m_position[3];    // (Px, Py, Pz)
  float   m_normal[3];      // (Nx, Ny, Nz)
  float   m_u, m_v;         // texture coords
  uint8_t m_jointIndex[4];  // joint indices
  float   m_jointWeight[3]; // joint weights (last weight omitted)
};
```

## Rendering Per-Vertex Skin Pose

- Matrix Palette
  - Joint transformations as matrices
    - Can be pre-computed
      - especially if using regular update/draw intervals
  - Passed to GPU as mat4 array
  - In lieu of model matrix
- Vertex attributes
  - Indices and weights for the joints