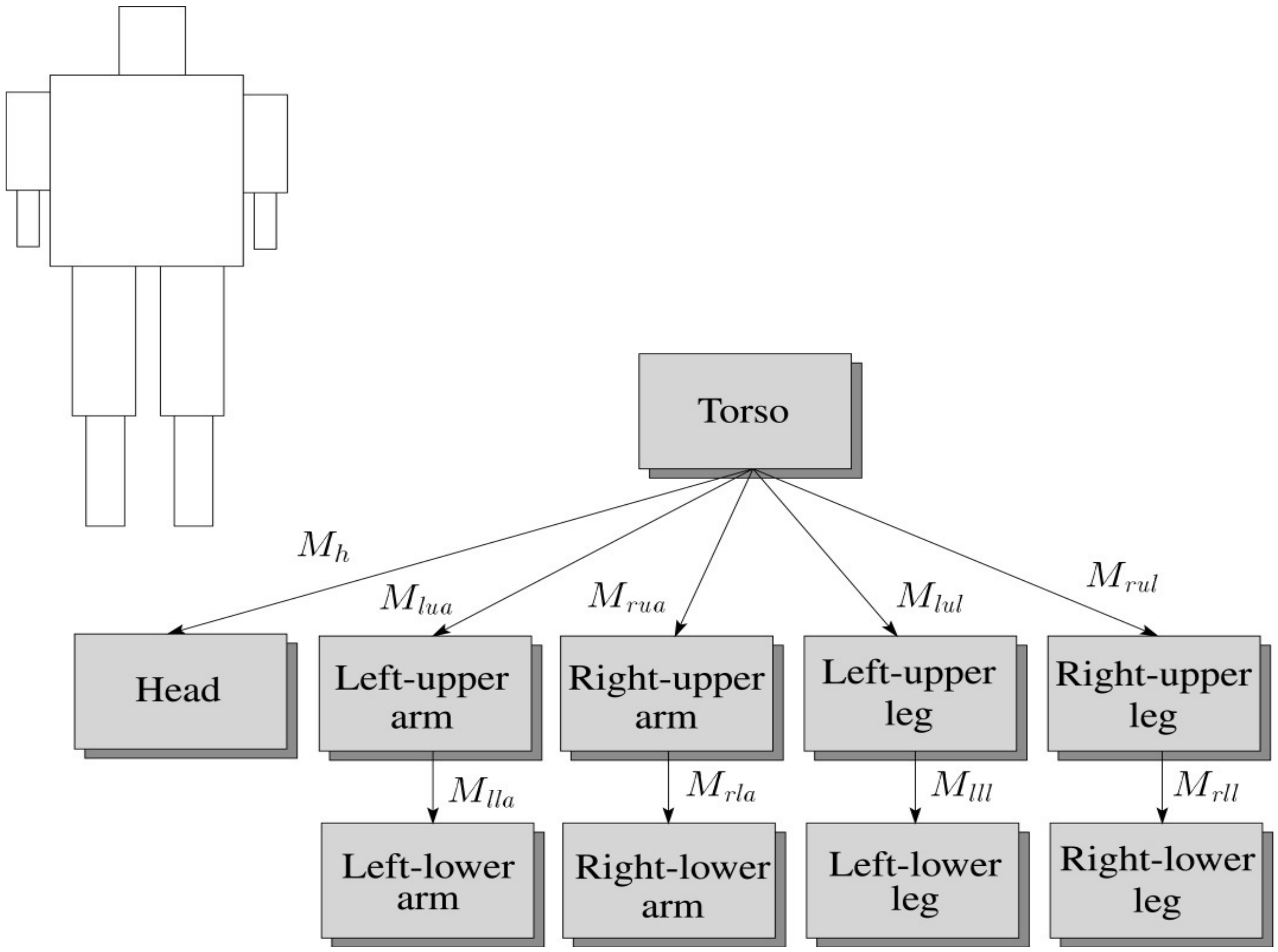
CS354R

DR SARAH ABRAHAM

INTERACTIVE ANIMATIONS

REMEMBER SCENE HIERARCHIES?



MANY TYPES OF HIERARCHIES

ANIMATION

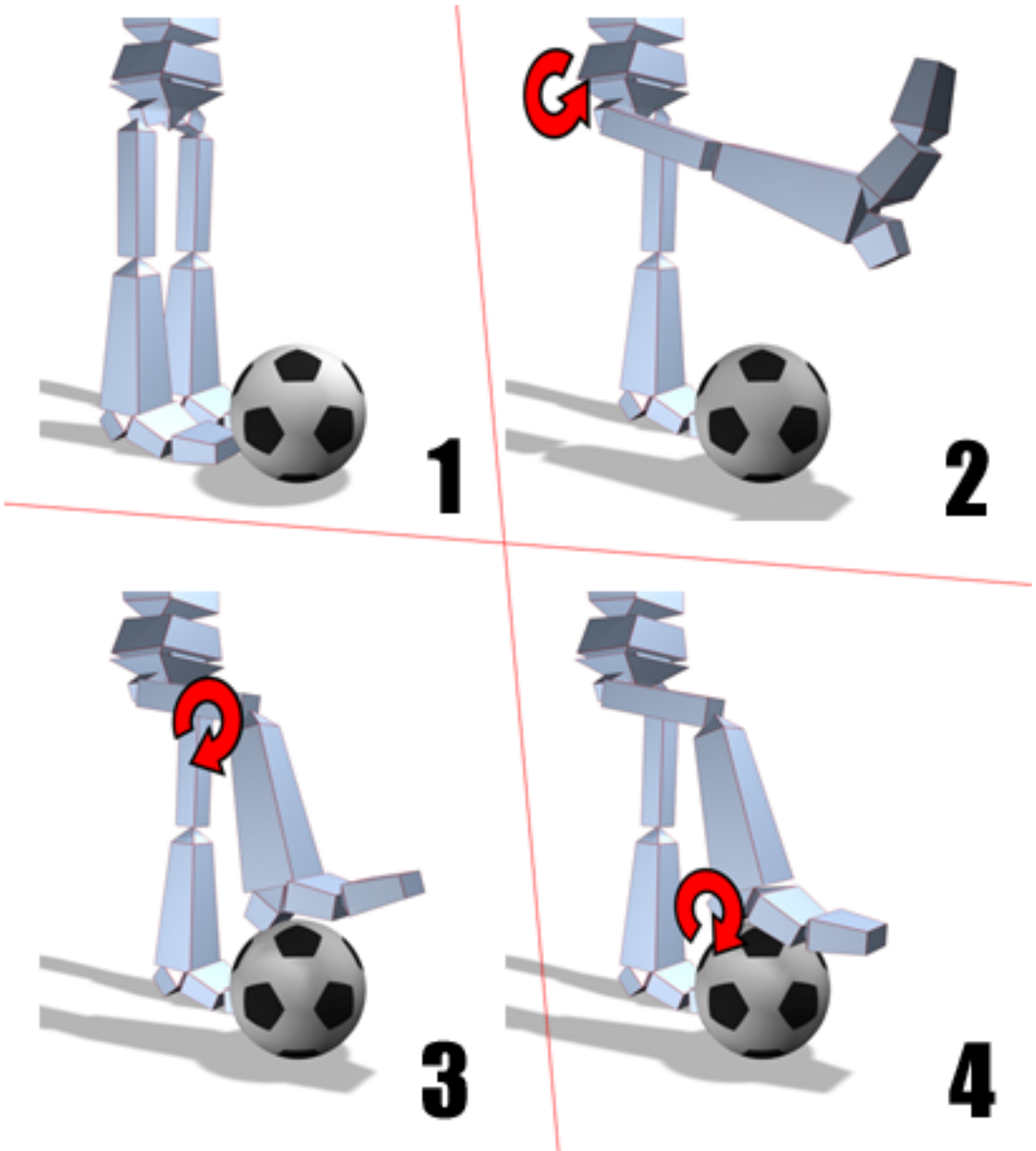
- The above examples are all **articulated models:**
  - Rigid parts (bones)
  - Connected by joints
- They can be animated by specifying the joint angles (or other display parameters) as functions of time.
  - Direct control of joints
  - Inverse kinematics (IK)

DIRECT CONTROL OF JOINTS

- Model has “T” or “Y” or “A” pose
  - Defines rest orientation of bones/joints
  - Includes skin weights specifying joint influence on each polygon
- Specify target pose by changing bone orientation and position in world space
- Map between default pose and target pose to update position of polygons

https://sketchfab.com/salmonax

FORWARD KINEMATICS



INVERSE KINEMATICS

- Inverse kinematics take the **target pose** and compute all necessary joints along the chain to reach that pose
- Forward kinematics are easier to compute but harder to reason about
- Inverse kinematics are more natural to reason about but harder to compute

DEFINING TARGET POSES

- Skeleton must include **kinematic chains** along parts of hierarchy with joint dependencies
  - e.g. Dependency extends from shoulder to wrist
- Each kinematic chain has an **end effector** to target different positions (or orientations)
  - New position of end effector updates all joints along kinematic chain
- May be possible to solve analytically but usually solved approximately and iteratively
  - Minimization problem: get end effector as close to target position as possible

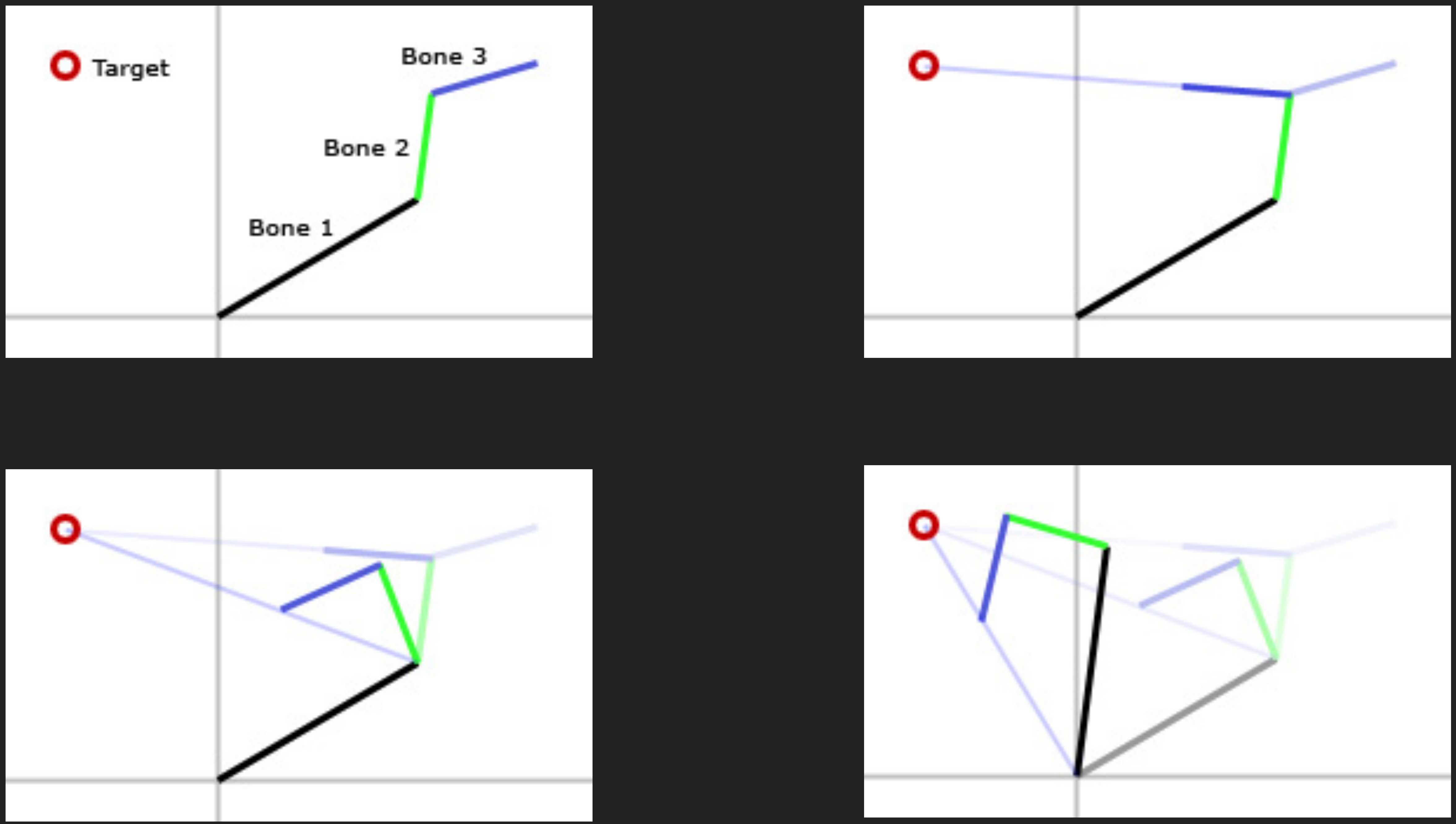
INVERSE KINEMATICS EXAMPLE

JACOBIAN TECHNIQUE

- A Jacobian matrix is a matrix of first-order partial derivatives of system
  - Describes changes in end effector position based on changes in joint angle
- Jacobian inverse allows computation of joint angles from changes in end effectors
- Goal is to perform small, iterative changes to joint angles using Jacobian to reach target position
- Must calculate Jacobian
  - Numerically or analytically
- Must approximate inverse
  - Pseudo-inverse or transpose

CYCLIC COORDINATE DESCENT

- Iterative optimization algorithm used to reduce error or find minimum of function
  - Minimize distance between end effector and target position
- Individually adjusts joint angles starting at last link and working backward
- Determine joint-to-end-effector vector and target-to-end effector vectors
  - Can determine angle between them (or amount to rotate) using dot product
  - Can determine direction to rotate using cross product

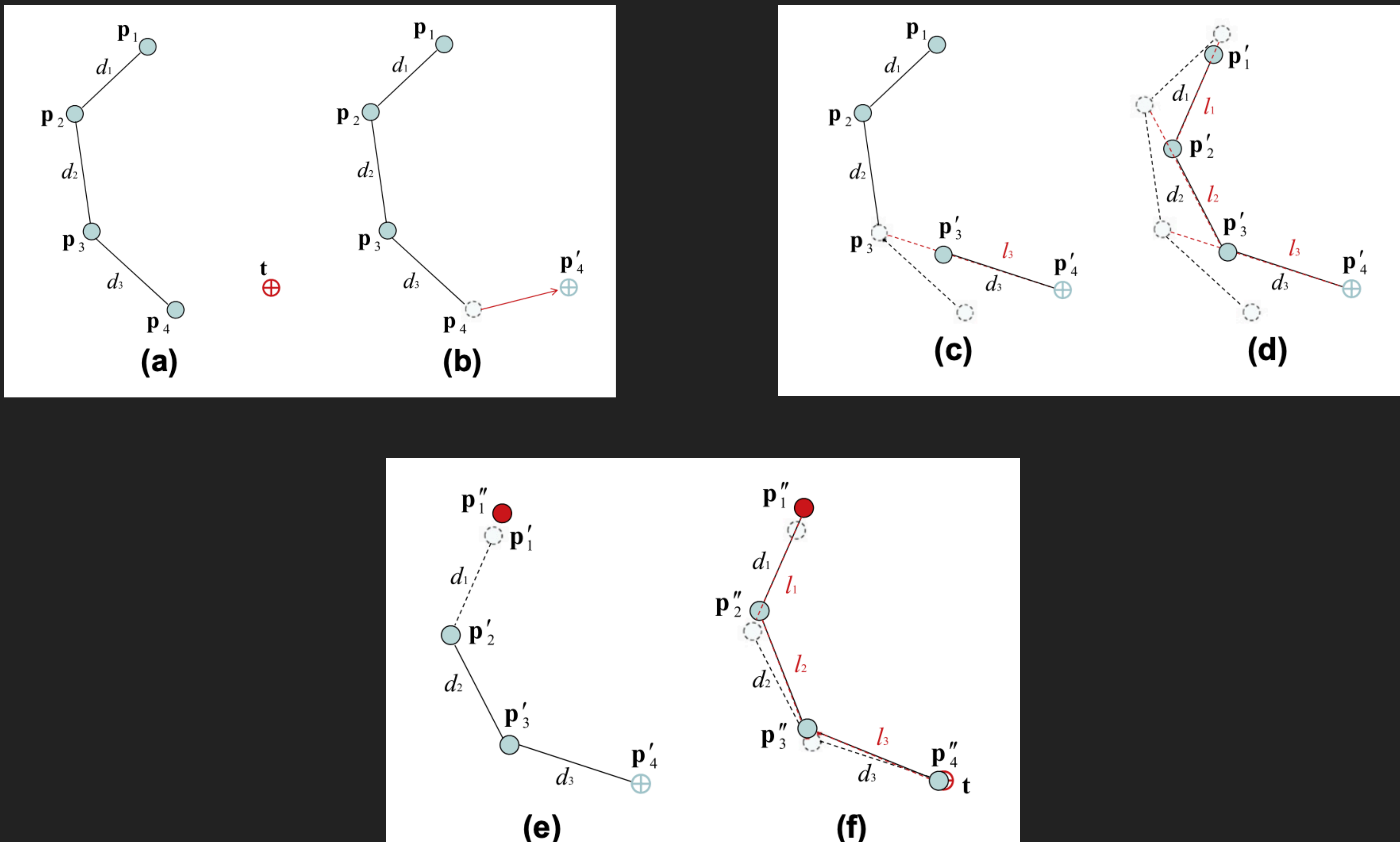

Target
Bone 3
Bone 2
Bone 1

LOCAL MINIMA

- Issue with all optimization algorithms
- Solution finds a local (but incorrect) minima and is unable to continue descent toward global minima
- In CCD, bones do not consider other bones – only distance to optimal position
    - Leads to “tangling” (bones placed in optimal positions but do not respect chain)
    - Placing rotational constraints on joints reduces these errors but cannot fully correct for it

FABRIK

- Forward and Backward Reaching Inverse Kinematics
- Iterative method to find joint position on a line rather than considering joint angles
- Determine if target is reachable (i.e. distance from root to target is less than total length of chain)
- Move end effector to target position (within tolerance) and recalculate positions of previous joints
- Move root joint back to its initial position and recalculate positions of all child joints



p1
p2
p3
p4
d1
d2
d3
t
(a)
p'4
(b)
l1
l2
l3
p'1
p'2
p'3
(c)
(d)
p''1
p''2
p''3
p''4
(e)
(f)

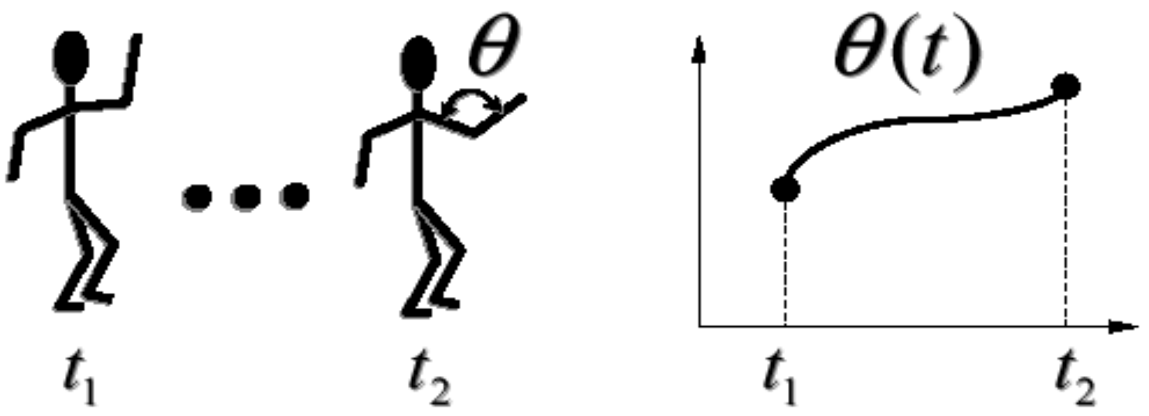
IK IN GAMES

- Emphasis on stability and speed
- Inverse Jacobian common in film but too slow for games
- CCD has unwanted pathologies
- FABRIK is extremely efficient, simple to implement, and looks good
  - Can also apply FABRIK with joint constraints and to hierarchies with multiple end effectors

HOW DO WE ANIMATE?

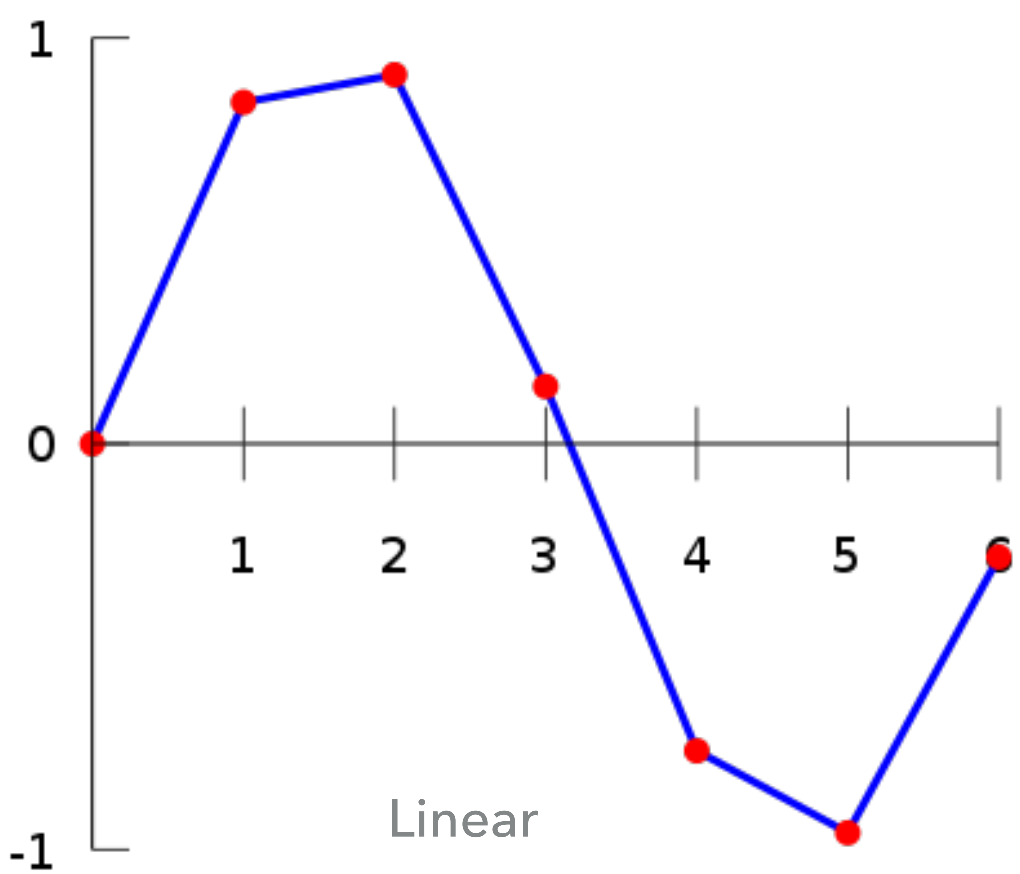
KEY-FRAME ANIMATION

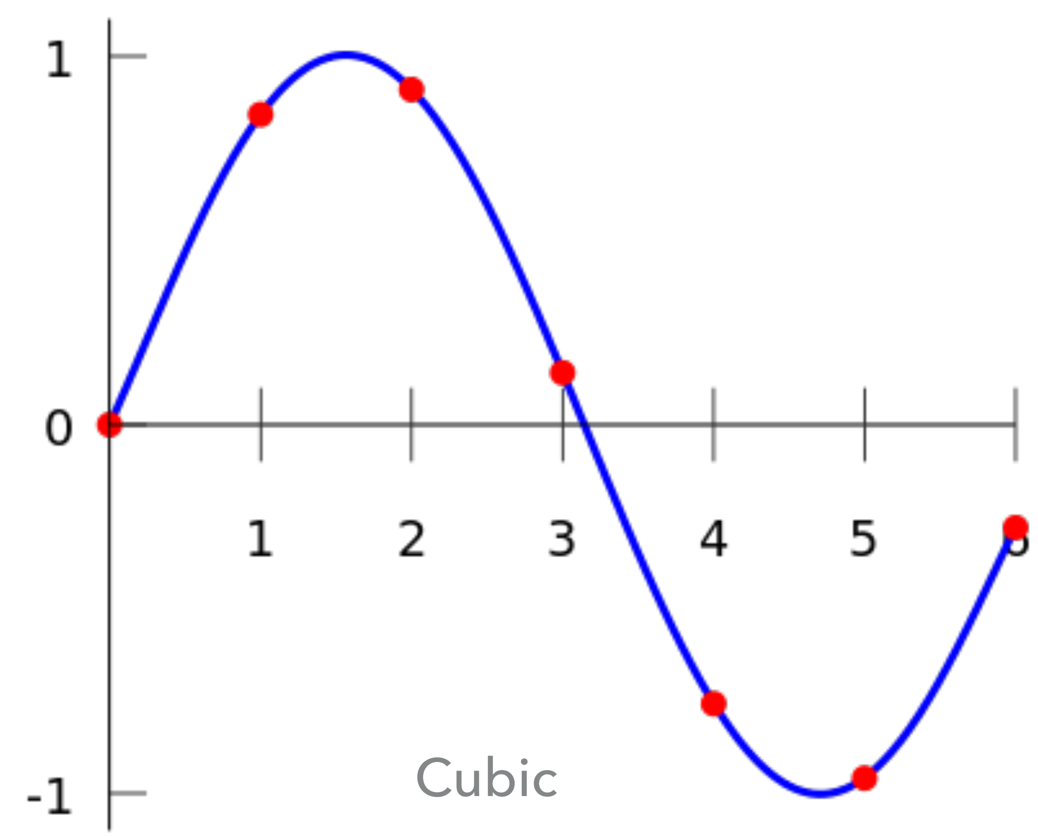
- Use of **key-frame animation.**
  - Each joint specified at various **key frames**
  - System does interpolation or **in-betweening**
- In addition to joint control, we must have:
  - Key frame interpolation (e.g. splines)
  - A good interactive system
  - Animator skill



INTERPOLATION

- Fills in positions/angles between the start and end positions given in the key frame
- Linear and cubic interpolations commonly used
    - Simple to calculate and small number of points required





SPLINES

- Compose complex, high-degree curves out of simpler, low-degree piecewise curves
  - Easier to reason about mathematically
  - More controllable for artists
  - Faster to calculate
  - Can control curve's smoothness
- Smoothness describes how many of a function's derivatives are continuous
  - Usually only need C^2 continuity, or continuity in the second derivative
  - May want lower continuity for artistic purposes

PUTTING IT TOGETHER

- Humble ideas…
  - https://www.youtube.com/watch?v=DRYhorZDVyw
- Lead to much bigger ideas…
  - https://www.youtube.com/watch?v=D9dC6-nivyk

WHAT ABOUT THESE THINGS?

SOFT BODIES

- Treats objects as deformable
  - Shape of object can change
- Accurate simulation is more computationally intensive
  - **Finite element simulation** breaks system into smaller, solvable subdomains that can be reassembled
- Mostly faked in games
  - Rigid bodies in a lattice can fake slime, cloth, etc
- Godot Phyiscs does support soft bodies but more commonly used in pre-simulated objects

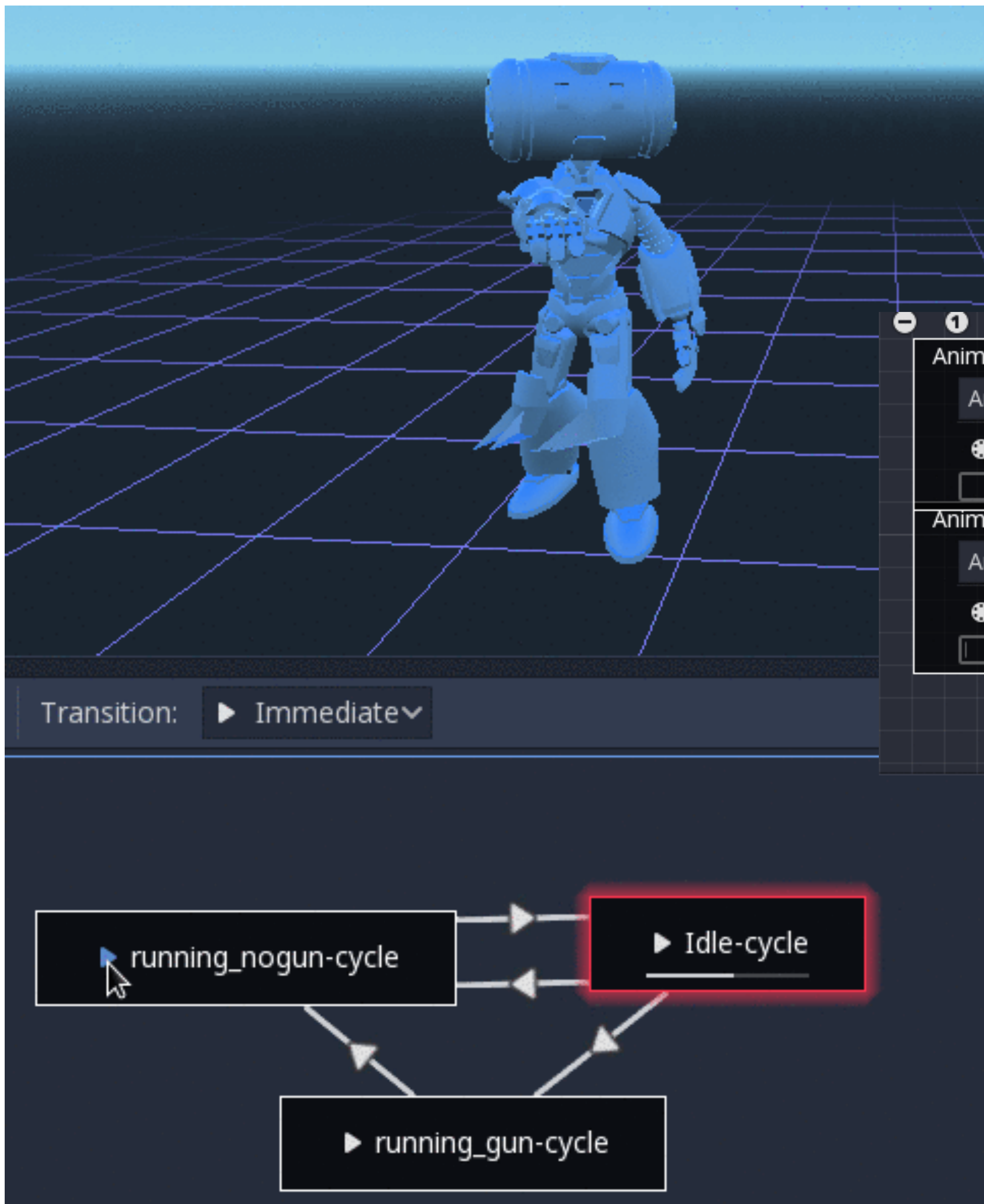
DEFORMATION AND FRACTURE IN GAMES

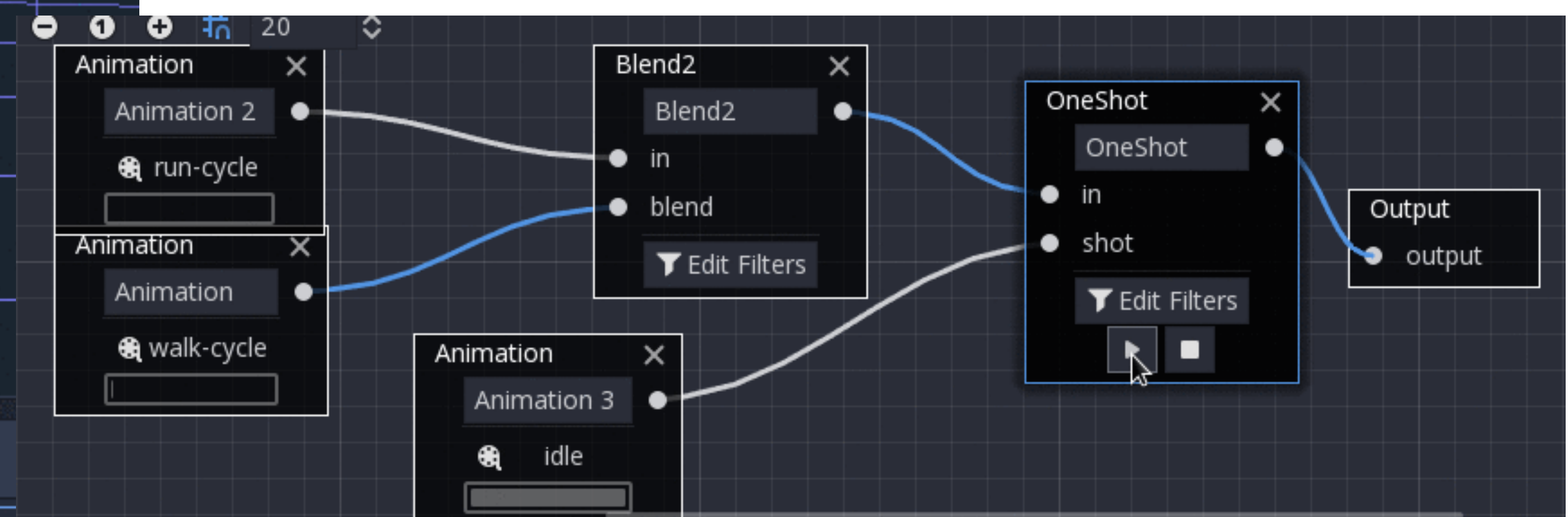
https://www.youtube.com/watch?v=Iy64-Bn7i4k

CONNECTING ANIMATION TO CODE

- Animations must transition based on player actions
- Use of animation **state machines**
  - Track what animation states transition into other animation states
  - Track player interactions and state that can trigger these transitions
- Artists can blend between animations for a more seamless transition

ANIMATION STATE MACHINES IN GODOT





REFERENCES

- [https://medium.com/unity3danimation/analytical-jacobian-ik-cb3df86edf00]
- [http://www.ryanjuckett.com/programming/cyclic-coordinate-descent-in-2d/]
- [http://www.andreasaristidou.com/publications/papers/FABRIK.pdf]
- [http://graphics.berkeley.edu/papers/Parker-RTD-2009-08/Parker-RTD-2009-08.pdf]
- [https://godotengine.org/article/godot-gets-new-animation-tree-state-machine]