**Animation**

- Animate
  - breathe life into, enliven
- Animated cartoon
  - film made from gradually varying sequence of drawings
  - still images shown in sequence appear to move
- Animation makes things move or change
  - according to artist's whims
  - define start and end positions and shapes, define how to interpolate
- Dynamic simulation that makes things move or change
  - based on starting situation
  - a set of rules, such as approximations of laws of physics (kinematics, dynamics), collisions, etc.

**Ways to animate**

- Apply a gradual rigid transformation to an object
  - translate, rotate whole object or joint angles
- Apply a free-form transformation to the space
  - an arbitrary affine or projective transformation
- Modify the definition of an object
  - change the vertex locations
- Animate the camera
  - move it, turn it, zoom in/out
- Change material properties, lighting
- In general:
  - change something (anything) as a function of time

**Keyframing**

- Traditional cartoon process a'la Disney
  - director decides how the story goes
  - master animators draw the main characters (and sketches of background) in **keyframes**
  - junior animators draw the frames between the keyframes (in-betweens), the process in called **tweening**
  - yet others draw the backgrounds
  - a combined image is photographed by stacking the characters in front of the background
- Animation packages borrow the same idea
  - you model the components of the scene
  - you place them at important times, or keyframes
  - the computer interpolates the keyframes (tweens) and renders the images

**What can you keyframe?**

- Object related things (global)
  - size
  - location
  - orientation, …
  - material properties
    (hit 'I' key with mouse in material buttons window (F5))
  - textures (material key, 'I', 'All Mapping')
- Object definition (local)
  - morph vertices between various shapes (vertex keys)
  - morph the relative position of vertices with respect to a base shape
    (relative/weighted vertex keys)
Keyframing in Blender

- Create an object, place it
  - hit ‘TAB’ to leave edit mode
- Insert a key for location
  - hit ‘I’, select ‘Loc’
- Advance the current frame to 50
  - use cursor keys
- Move the object (‘G’ to grab), insert key
- When you change the frame (<50)
  - the object location is interpolated
- Quick preview
  - set end of animation to 50 (F10, Lclick-SHFT)
  - animate in the window mouse (ALT-A)
  - in all windows SHFT-ALT-A

Rendering the animation

- Store the result in /render/sph0001_0050.avi (F10)
- Display buttons
- Enable sequence editor
- Current frame
- Create animation
- Play it back
- Size of the rendering
- Start and end frames
- Type of output
  - AVI Jpeg @ 25 frm/sec, RGB
  - No antialiasing (OSA = OverSampling)
  - or motion blur or shadow calculations

Interpolation

- Animation is mostly about keyframe interpolation
- We have already seen several types of interpolation
  - constant: \( x(t) = x_0, 0 < t < 1; x_1, 1 < t < 2; \ldots \)
  - linear: \( x(t) = x_0(1-t) + x_1t \)
  - we could fit polynomials
    - high degree polynomials => too wiggly, hard to control
    - low degree polynomials => must concatenate segments => beziers or splines
- Positions are easier to interpolate than orientations
  - positions: just interpolate…
  - orientations: can be ambiguous usually best: there is a unique rotation axis that rotates one orientation to another, take the shorter of the two rotations, interpolate the rotation angle
  - \( R, R' = R, \Rightarrow R = R, \theta R' \)
    - convert \( R \) to quaternion \( q \), get axis \( a \) and angle \( \theta \)
    - interpolate \( \theta \), get new quaternion \( q' \), convert to \( R' \)
    - new interpolated orientation is \( R, R' \)

Interpolation curves (IPOs)

- Keyframes add vertices to IPOs
  - select object in 3D window
  - change window by SHFT-F6
  - press home to view curves
- Axes
  - horizontal: frames (time)
  - vertical: value
- You can edit them directly
  - select curve (Rclick) and TAB to edit mode, ...
  - constrained to be function of time
  - 3 interpolation types (hit ‘T’)
    - constant, linear, bezier
  - 4 extrapolation types
    - constant, extrapolate, cyclic, cyclic extrapolate
- Back to 3D window with SFHT-F5

Material IPOs

available only if a material is defined
**Vertex keyframes**

- Morph between shapes
- How to do it in Blender
  - add keys by Insert Key 'Mesh' (or Surface or Curve) in 3D window (choose absolute)
  - when not in EDIT MODE (TAB) select the current key (RClick) in the IPO window (SHFT-F6)
  - edit the vertex locations of the selected keys in EDIT MODE

![Diagram showing vertex keyframes](image)

**IPO for vertex keys**

- There are three keys
  - horizontal lines at 0.0, .5, 1.0
  - at first they appear on top of each other, but you can move them up and down ('G')
  - the orange one is the 1st key (yellow when selected)
  - all others are blue (light blue when selected)
- Current frame is 36
  - vertical green line
  - The red curve is the speed curve
    - a Bezier curve, modifiable
    - in this picture it's been made cyclically continuous (repeats)
    - intersection with green line tells that we are roughly halfway between the 2nd and 3rd key

![Diagram showing IPO for vertex keys](image)

**Relative vertex keys**

- If we have many independent parts that we want to animate in sync, vertex key morphing is difficult to use
- Take face animation
  - basic neutral face
  - want to move eyes, mouth, etc., relative to the neutral face
  - move parts independently but in a coordinated fashion
- Sequence of operations
  - define the basic shape, insert key
  - starting from it, modify a part of the shape, insert a key
  - repeat for the other parts
  - each key gets a weight curve that is used to tell how much of the relative change is applied at a given point of time
  - weight > 1.0 extrapolates, weight < 0.0 does the opposite motion

**Relative vertex key example**

- "Smile -> surprise -> laugh" reaction
- Faces: smile, eye brows raised, neutral
- Each has a key (horizontal) and a weight
- Mouth: half-smile -> neutral -> big smile
- Brows: up -> down -> wiggles up

![Diagram showing relative vertex key example](image)
Recipe for the previous slide

- Create the neutral face
  - Use Mesh->Plane to start
  - SHFT-D to duplicate
  - CTRL-J to join into single mesh in the end
- Add three vertex keys
  - 'I' 'Mesh' Relative
  - Move them apart ('G') for selection, their location is irrelevant
- Edit two of the (blue) keys
  - Select a blue key in IPO window by right clicking
  - Edit the mouth (in 3D, edit mode (TAB))
  - Exit edit mode (TAB), select the other blue key in IPO, edit brow
  - Now if you select the 3 keys, you get either neutral face, smile, or raised brows
- Add weight curves
  - When the ‘----’ above Key 1, etc., is selected, you see the keys
  - Select mouth key, then Left-click Key 1, CTRL Left-click to add curve (now flat)
  - Select ‘----’, select brow key, select Key 2, add another curve
  - SHIFT Left-click Key 1 and Key 2 to see both lines
- Modify the weight curves
  - Select a curve, add new control points with CTRL Left-click
  - You can modify the curve points in edit mode
- Review animation
  - by ALT-A in 3D window
  - or if you move the green bar that selects current frame

Kinematics

- Given this robot arm
  - the shapes and dimensions of its parts (base, lower arm, upper arm)
  - the current configuration (rotations by $\theta$, $\phi$, $\psi$)
  - Where is the end effector?
- Kinematics gives the solution
  - go to the end of one limb at a time, starting from the base,

Blender example

- Create a simple robot
  - a cube for the base
  - exit edit mode (TAB) so next one is a separate object
  - combine a cylinder and a box to make an arm
  - new parts in edit mode extend the same object
  - place rotation center at center of cylinder
  - select cylinder center, move cursor there with SHFT-S
  - 4, exit edit mode (TAB), “Center cursor” at edit buttons
  - duplicate arm (SHFT-D) twice and shrink (S)
  - after shrinking, CTRL-A to “apply scale to vertices”
- Forward kinematics is simple
  - make the base arm the parent of the middle arm
  - Rclick middle arm, SHFT-Rclick base arm, CTRL-P, OK
  - repeat for the other segments, and arms to base
  - now if you rotate (R) base or one of the arms, the rest will follow
- Add weight curves
  - When the ‘----’ above Key 1, etc., is selected, you see the keys
  - Select mouth key, then Left-click Key 1, CTRL Left-click to add curve (now flat)
  - Select ‘----’, select brow key, select Key 2, add another curve
  - SHIFT Left-click Key 1 and Key 2 to see both lines
- Modify the weight curves
  - Select a curve, add new control points with CTRL Left-click
  - You can modify the curve points in edit mode
- Review animation
  - by ALT-A in 3D window
  - or if you move the green bar that selects current frame

Inverse kinematics

- Kinematics (aka forward kinematics) is very easy
  - always a unique solution
  - but usually we want to do the opposite
- Inverse kinematics turns the situation around
  - given the shapes of the robot arm and the location where its end effector should reside
  - Which configuration should the robot be in?
- Not so straightforward
  - there may be multiple, even infinite number of solutions
  - there may be no solution
- In animation: could just try configurations
  - when the constraints are not satisfied, backtrack
  - can be very tedious
Inverse kinematics in Blender

- IK in Blender is called "Armatures" since Blender 2.20
  - before that they were "IKAs", not available any more
  - an armature is like the skeleton, made of bones (arms, legs, spine, ...)
    - separate bones added in edit mode are part of the same armature
- Here are links to a few tutorials
  - http://members.lycos.co.uk/kurppa/blender/ik.html
  - http://members.yourlink.net/gruff/bones/index.html
  - http://blenderchar.weirdhat.com/handtute/

Inverse kinematics

- Need an armature for IK
  - select the first arm segment (Rclick), move cursor to it (SHFT-S 4)
  - add four bones of armature (Lclick), ESC quits
- Name the bones
  - with armature selected, in editmode, select all (A), in edit buttons (F9) hit "Draw Names" and rename bones (in "Selected Bones")
- Add moving bone
  - select sphere btw. B3 & B4, add a single bone by extruding (E)
    - it’s a "handle" for moving the whole set

IK cont.

- Set constraints
  - in pose mode (CTRL-TAB, or click the "smiley face"), select B4
  - select Object panel (F7), constraint buttons, add "IK Solver"
  - fill in the name of the armature (Armature) and the control bone (Bone)
- Now if you select and move Bone, the rest of the armature follows it
  - the orientations and locations of the segments are recalculated

IK cont.

- Attach geometry to armature
  - get out of pose mode (TAB) and edit mode (TAB)
  - select a robot segment (Rclick), and the armature (SHFT-Rclick), make the armature parent (CTRL-P), use bone B1
  - repeat for the other segments
  - note: may need to reposition and orient the parts
- Now in pose mode (select armature, CTRL-TAB) you can move the robot arm
**Skinning**

- If you just attach rigid geometry to bones, they don't behave naturally at the seams
  - they look very stiff, like a robot
  - between segments you can see cracks instead of continuous joints
  - with the robot, we dealt with that by hiding joints with cylinders

- Solution
  - associate a vertex with several bones
  - weight each association
  - when transforming bones, apply a weighted combination of the transforms to each vertex

- In Blender this is done by
  - assigning vertex groups to bones,
  - painting a vertex weight, [link](http://mmaigrot.free.fr/didac-blender/ikas-eng/ik-p5.php)

**Dynamics**

- Definitions:
  - **Kinematics**: how the positions of the parts vary as a function of the joint angles
  - **Dynamics**: how the positions of the parts vary as a function of applied forces

- Requires real physics simulation

- Forward dynamics
  - integrate the effects of the forces as a function of time

- Inverse dynamics
  - given the desired path / behavior, which forces are needed?

- This is already rocket science
  - control theory

**Particle systems**

- Use particle systems to model various natural phenomena
  - dynamic:
    - fire, explosions (flames, sparks)
    - smoke (soot)
    - rain, snow (rain drops, snow flakes)
  - static
    - grass
    - fur

- Properties of particles
  - they are created, grow in age, may breed, get old and die
  - their colors and shapes may evolve
  - they can be affected by forces (wind, gravity, …)

- Flocking behavior
  - if you add an AI component, you can model the behavior of flock of birds, school of fish, pack of dinosaurs, …
  - widely used in movies (Jurassic Park, Quasimodo, …)

**Particles in Blender**

- Create a plane
  - Go to Anim Buttons (F7)
  - New Effect, change Build to Particles
  - Norm = 0.100 to emit particles with initial speed along plane normals
  - Rand = 0.100 to add randomness
  - Force Z = -0.070 to add gravity
  - Total number of particles
  - Duration of particles & their children
  - Randomness of particle age
  - Particle generation start & end (frames)
  - Damping works like friction
**Example: fire**

- Create a plane
  - Subdivide selected vertices (TAB, A, W 1)
  - Add particles, use these values
- Add material (F5) to emitter (plane)
  - select Halo, HaloTex
  - RGB = (1,0,0), Alpha = 0.7, Add = 0.7
  - Cloud texture for more realism
    - NoiseSize = 0.600
    - texture color to yellow (Map To), Ysize = 0.30 (Map input)
- Animate alpha (to fade flame away)
  - when animating materials of particles, frame 1=birth, 100=death
  - add key (I) at frame 1 in material buttons
  - at frame 100, set Alpha = 0.0, add key

**Example: Constrain a sphere above ground**

- Write a script file, open in text edit window (S-F11)
- Create a sphere, rename it to 'Pallo' in edit buttons
- Select script buttons, and object script links
- Add a scriptlink (New), name it, set to Redraw
- Try moving the sphere down, it sticks at z = 0.0 as every redraw calls the script

**Python scripting in Blender**

- Blender supports scripting in Python
  - modeling and animation tool (create and modify objects)
  - import and export scripts to other formats than *.blend
  - click documentation
    - Developer documentation for the Python API (a bit old)
    - Blender 2.31 Python API Reference (new)
- Editing
  - edit window SHFT-F11, R-click to open or load a document
  - editing best done with an external editor (e.g., Emacs)
  - after opening a document and editing in Emacs, reload by CTRL-R
- Run by hitting ALT-P in the edit window

**Example: modeling and animation**

- You can create or modify object geometry, attributes, location, orientation, etc.
- This time the script is not linked to an object, but the whole scene
  - use the scriptlink button with a small globe, Scene scriptlink
  - set to FrameChanged, so it’s called for each frame
- We’ll create a tetrahedron and spin it around on an elliptical path
Modeling and animation

```python
import Blender
from Blender import NMesh
from math import *

def add_face(m, inds):
    f = NMesh.Face()  # create a face
    for i in inds:   f.v.append(m.verts[i])  # add vertices
    m.faces.append(f)                               # add face to mesh

def create_tetra():
    me = NMesh.GetRaw()                             # create a mesh
    for v in [[1,1,1],[-1,-1,1],[-1,1,-1],[1,-1,-1]]:
        me.verts.append(NMesh.Vert(v[0],v[1],v[2])) # add vertices
    for inds in [[0,2,1],[0,1,3],[0,2,3],[1,2,3]]:
        add_face(me, inds)                          # add faces
    obj = NMesh.PutRaw(me, 'tetra', 1)              # add to scene
    obj.name = 'tetra' # change name

if Blender.Get('curframe') == 1:   create_tetra()   # create @ frame 1
tetra = Blender.Object.Get('tetra')                 # get a link
angle = (2*pi* Blender.Get('curframe'))/100         # vary rot angle
tetra.LocX, tetra.LocY = 5*sin(angle), 3*cos(angle) # move around
```

Postprocessing with sequence editor

- You can render things in pieces and then combine them
- You can mix and match video clips and 3D animation
- Example:
  - I have two prerendered animations
  - combine them with fading in
  - add a 3d text on top in the beginning

Add 1st sequence
SHIFT-A in sequencer window, Images, select and get a sequence of images set from frame 1 to 100

Postprocessing...

Add second film (now an avi): SHFT-A, movie, place it down with some overlap

Select first (Rclick), then second (SHFT-Rclick), then add (SHFT-A) Cross, move timeline (SHFT-UP/DOWN) there

Then add some keyframe animated text in 3D view

Finally add a scene, select 1st film and scene, add alphaover effect

Postprocessing

Filename Enable sequence rendering Do it!

1st clip and 3D text

Fading between clips
Motion capture

- Animating humanoid characters is difficult
  - even for artists, and with tools such as inverse kinematics
  - humans are too good in detecting which behavior looks real!

- Solution: track a real person
  - facial features
  - limb joints

- Apply the tracked motions to a CG model

Creating a sketch / plan / movie script

- Always plan a bit first
  - think about a key idea
  - brainstorm with your partner
  - first the story, only then how to implement it

- Write / draw a storyboard
  - What is the story?
  - How will you break it to scenes?
  - Which camera angles are you going to use?

- Implement it with Blender
  - Which elements of Blender do you need to implement your plan?
  - Which tutorials do you need to do?
  - What is the work flow?
  - How will you divide the tasks with your partner?