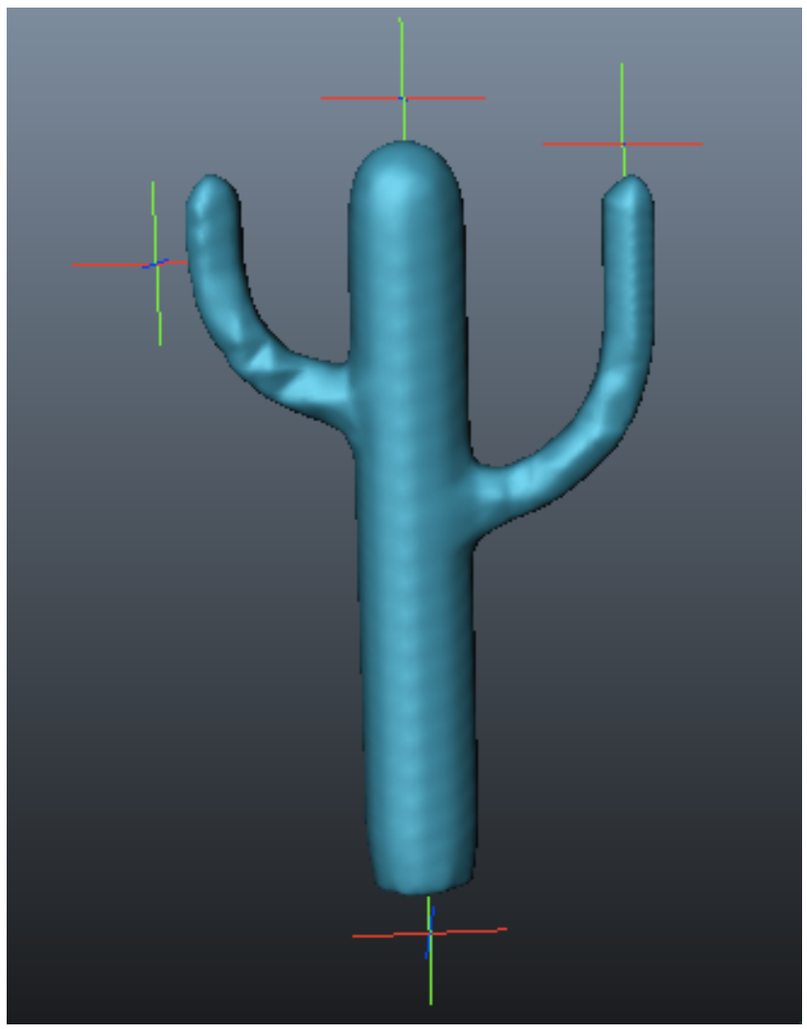
Shape deformation in Computer graphics

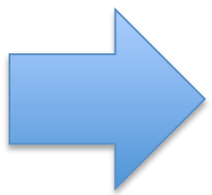
Shizuo KAJI
(Yamaguchi Univ. Japan / U. of Southampton UK)

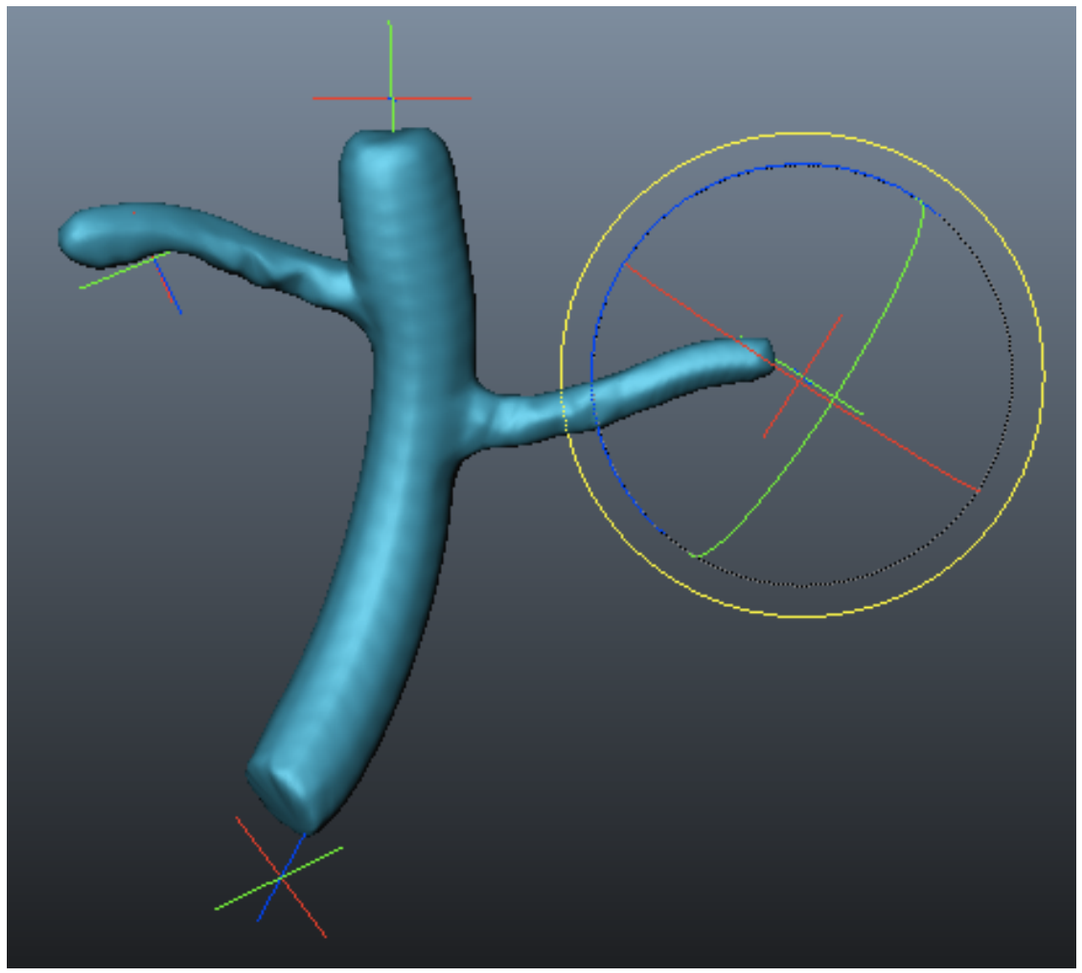
Preprints and Codes are available at: http://www.skaji.org/home/publications

Shape deformation: The problem

Produce animations/variations of a given shape







original shape + interactive user input => deformed shape

Physical vs non-physical methods

Physically-based algorithms have been successfully used (e.g. mass-spring model)

- Real and convincing outputs
- But, computationally expensive
  and difficult to direct outputs
- (bound by physics, little freedom in devising new methods...)

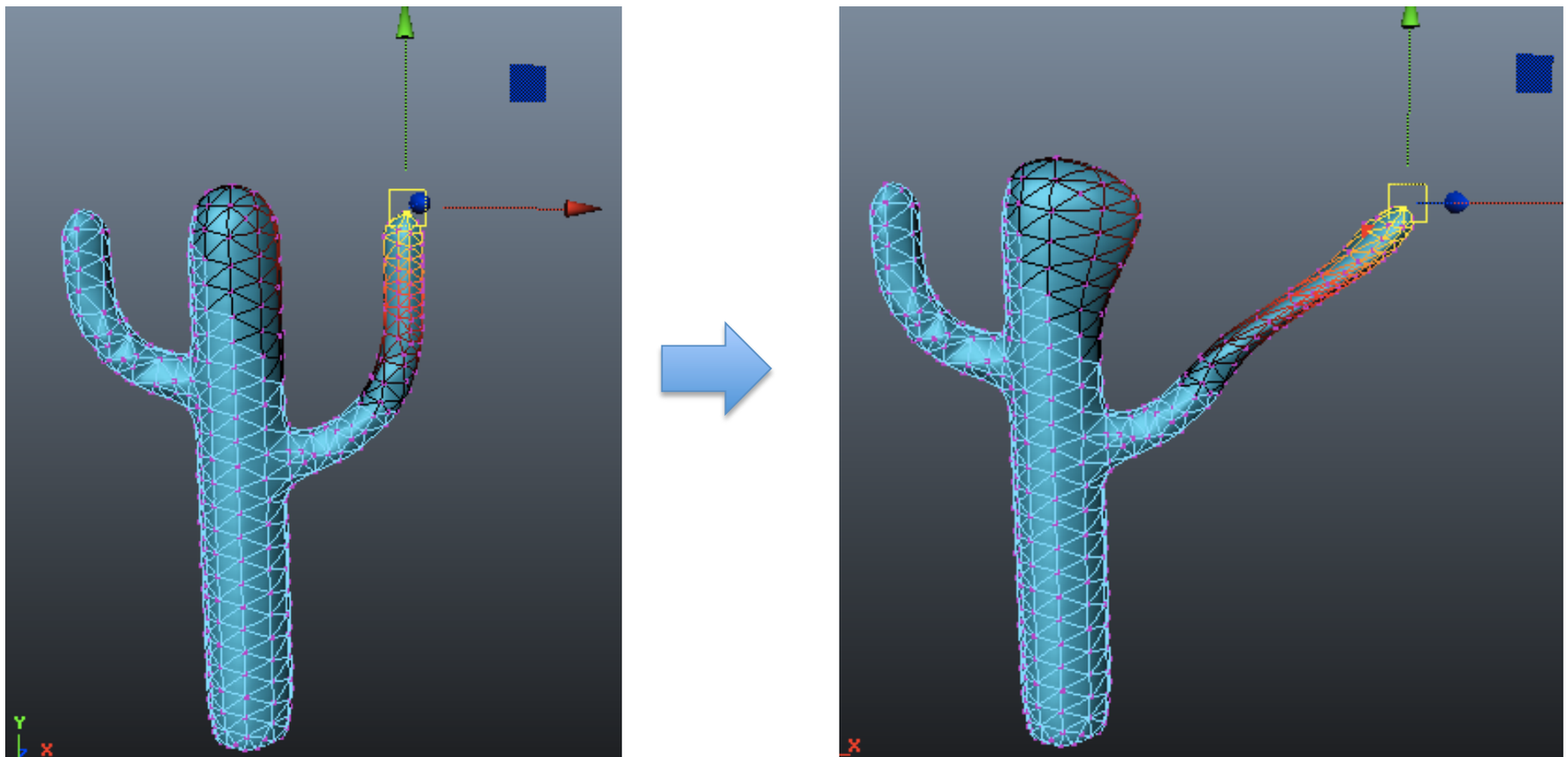
So

We discuss non physically-based methods with

- intuitive and interactive
- computationally cheap

General framework with a simple example

- Shape representation: vertex coordinates
- User interface:
  Pick a vertex and give it a displacement vector
- Amplification/interpolation of user's input:
  distribute the vector to all the vertices (by attenuating wrt distance)
- Reconstruction of the deformed shape from its representation: Add the vectors to the vertex coordinates

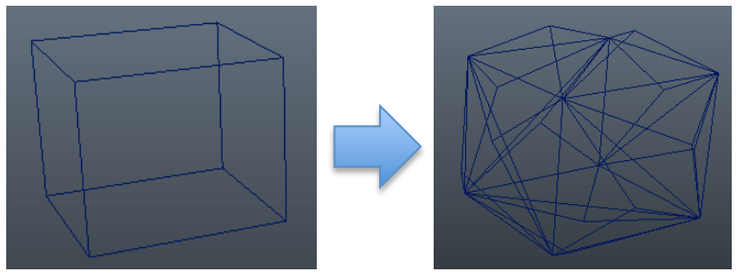


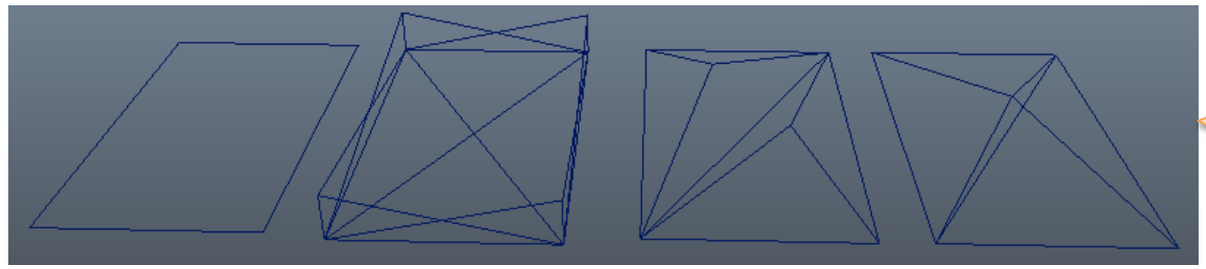
Fast and intuitive but doesn't look nice!

Our method

Input: Polygonal shape (common in CG)

1. Convert the polygonal shape to a 3D simplicial complex by “fattening” it

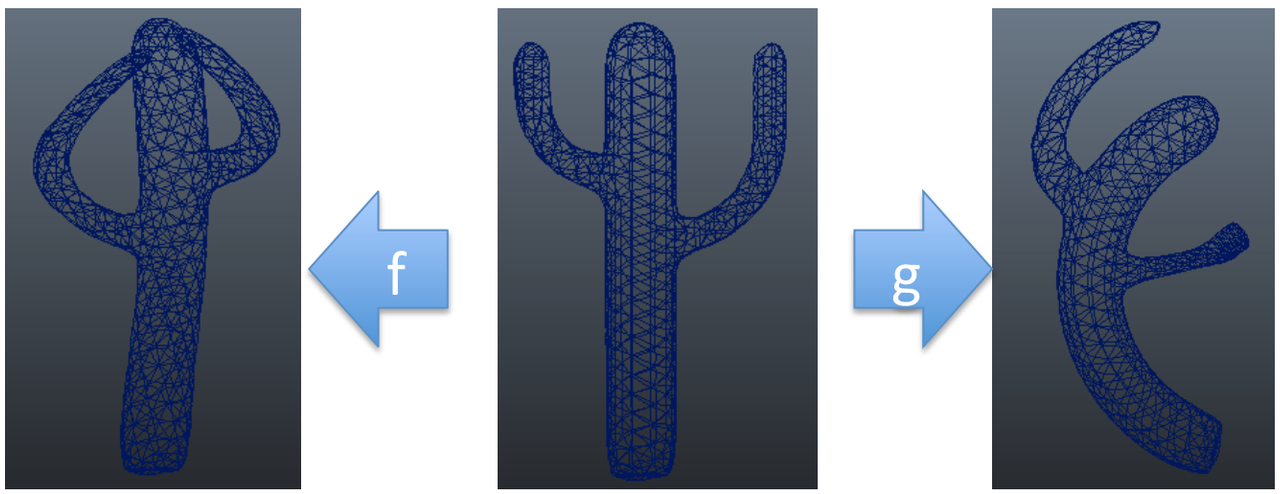




Three different fattening of a square

depending on what kind of geometric features should be encoded, there is a choice for fattening (shape of each face, angle between faces, angle around each vertex(curvature))

2. Deformed shapes are represented by Piecewise-linear maps



Shapes isomorphic to the original one
⇔ images of PL-maps from it
(we have a limitation that we can only produce deformed shapes which are isomorphic to the original)

3. PL maps are mapped into a vector space by applying Cartan decomposition piecewisely

On each simplex, a PL map is nothing but an affine map,
which is represented by a matrix A.
We have a version of Cartan decomposition which associates to A
two matrices R and S with

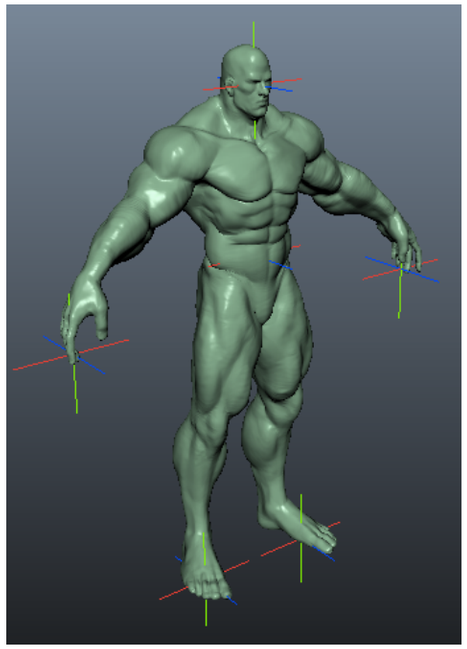
- A = exp(R) exp(S)
- exp(R) is a rigid transformation (i.e., a composition of translation and rotation)
- exp(S) is a symmetric transformation (i.e., anisotropic scaling)

We developed an efficient formula to compute the Cartan decomposition

4. Operate on the vector representation according to user's additional input

Since our data is now in a vector space, we can employ a lot of operations/techniques from linear algebra and calculus (e.g. interpolation)

5. Reconstruct shape by solving an optimisation problem in the vector space



3. The PL map representing the original shape is mapped into a vector space

PL map f

f

f''

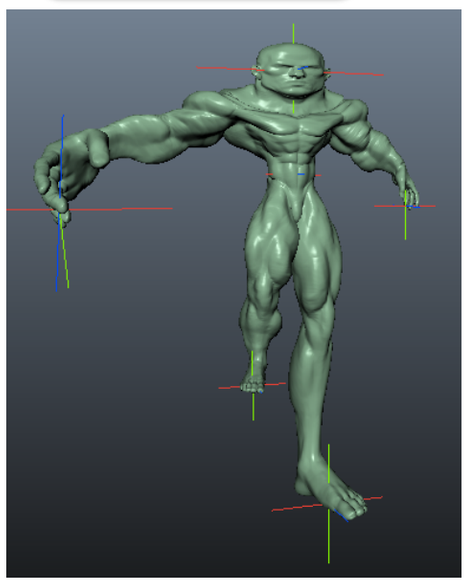
f'

6. The preimage is representing the deformed shape

PL map f''

4. The result of operations may fall out of the image

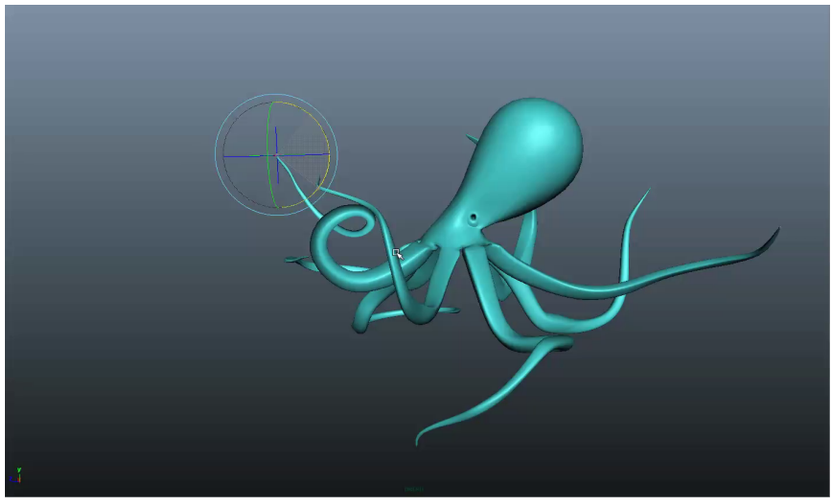
5. Find the closest element in the image



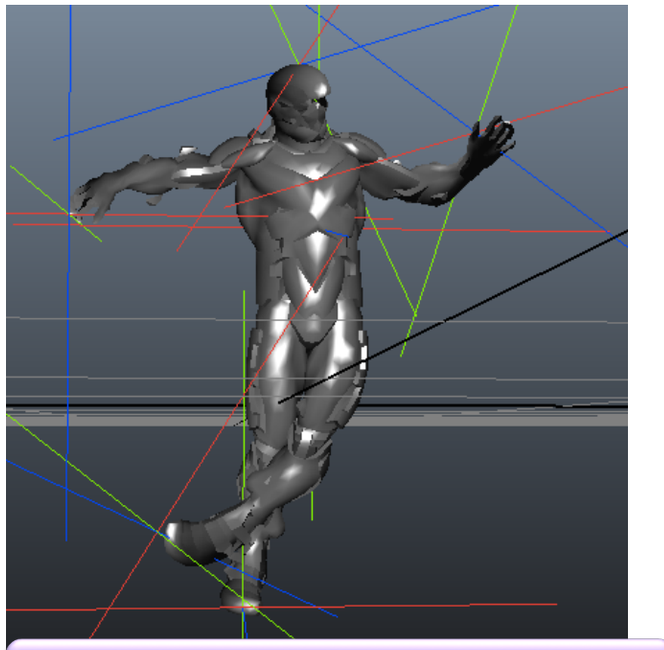
Good definition of "closeness" is essential to obtain visually plausible outputs efficiently

User Interface and applications

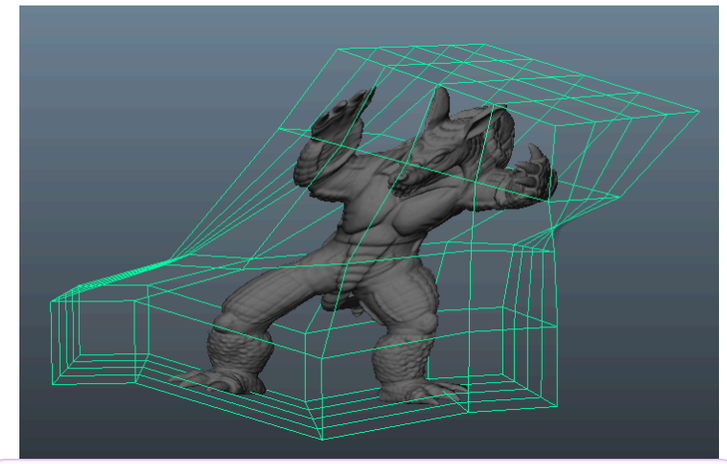
Depending on what kind of additional inputs the user gives, there are a lot of applications



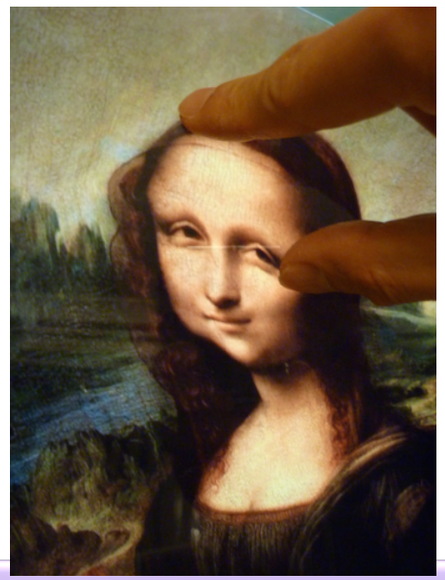
Swirling octopus's leg



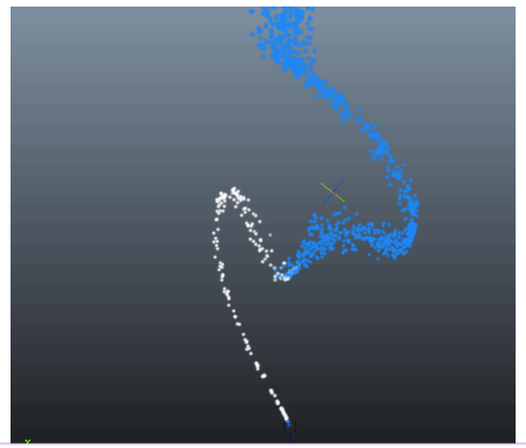
Posing human model



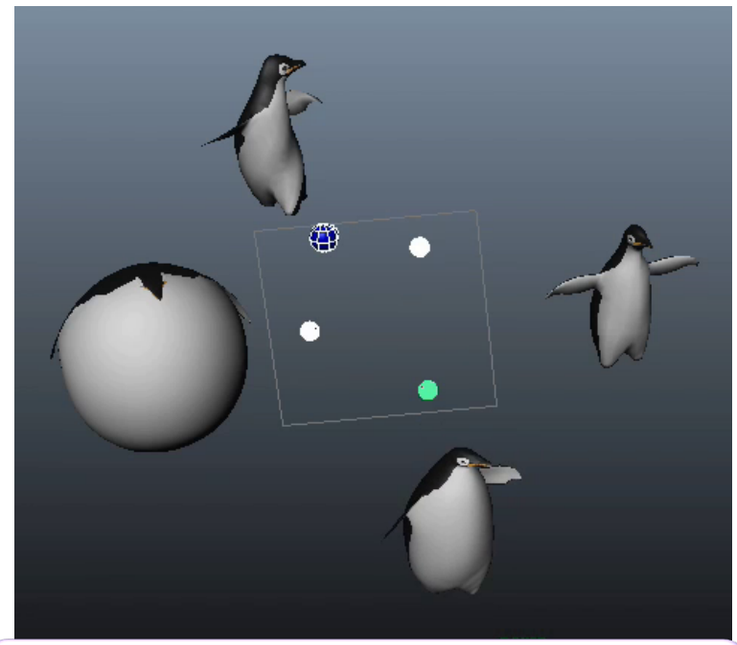
Mesh editing through simpler proxy cage



2D image editing on tablet



Directing particle simulation result



Morphing