ARLAB

ME/CprE/ComS 557

Computer Graphics and Geometric Modeling

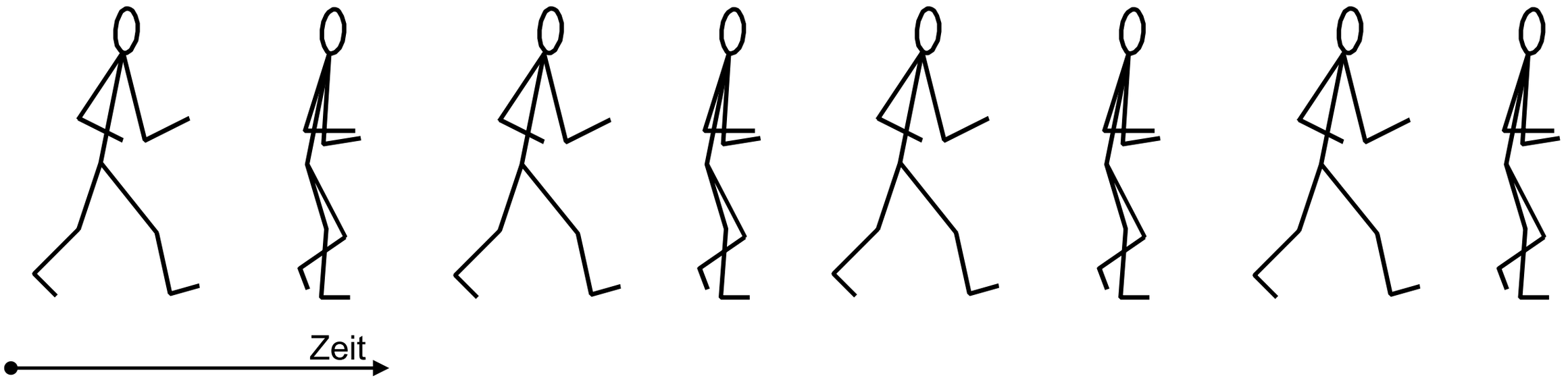
Animations

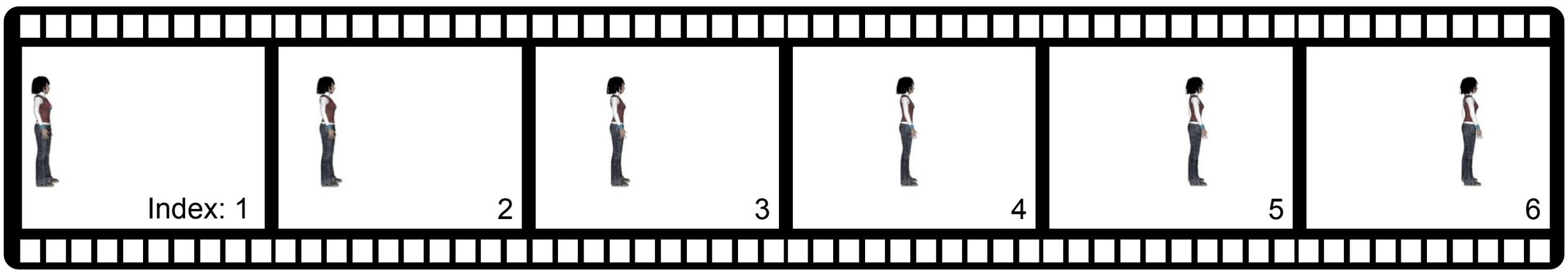
November 17th, 2015

Rafael Radkowski

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Animations





Animation is the process to generate moving objects on screen. The computer graphics rendering pipeline only generates static scenes. Animation techniques alter this scene from frame to frame. Thus, objects appear as moving objects.

Representing the keyframes

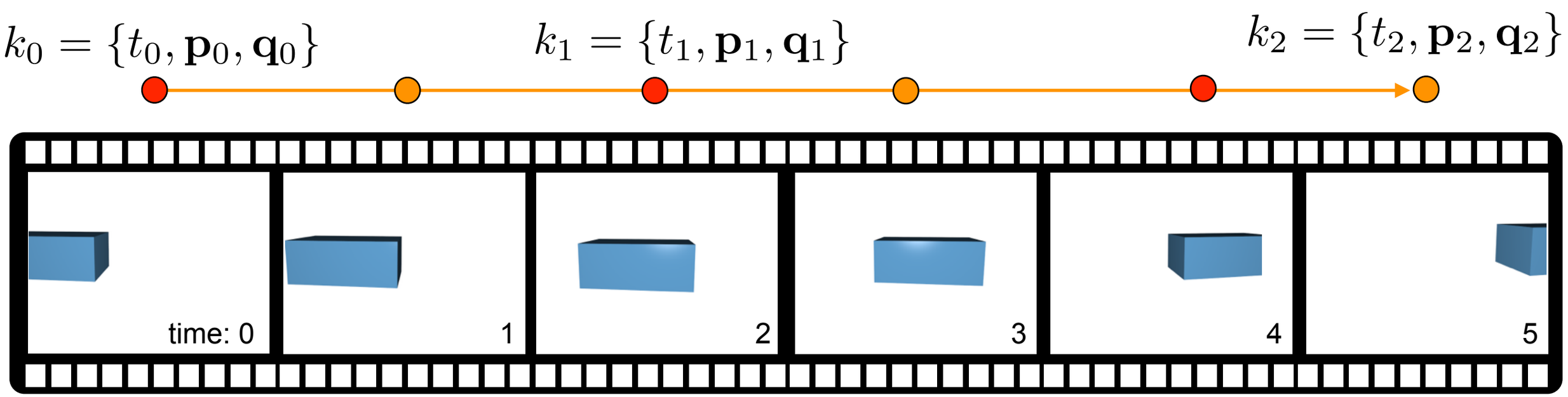
For every keyframe k_i we store

the time fraction in an interval between 0 and 1

the position as vector with {x, y, z} coordinates

the orientation of the object as quaternion q = {x, y, z, w}

$$k_i = \{t_i, \mathbf{p}_i, \mathbf{q}_i\}$$

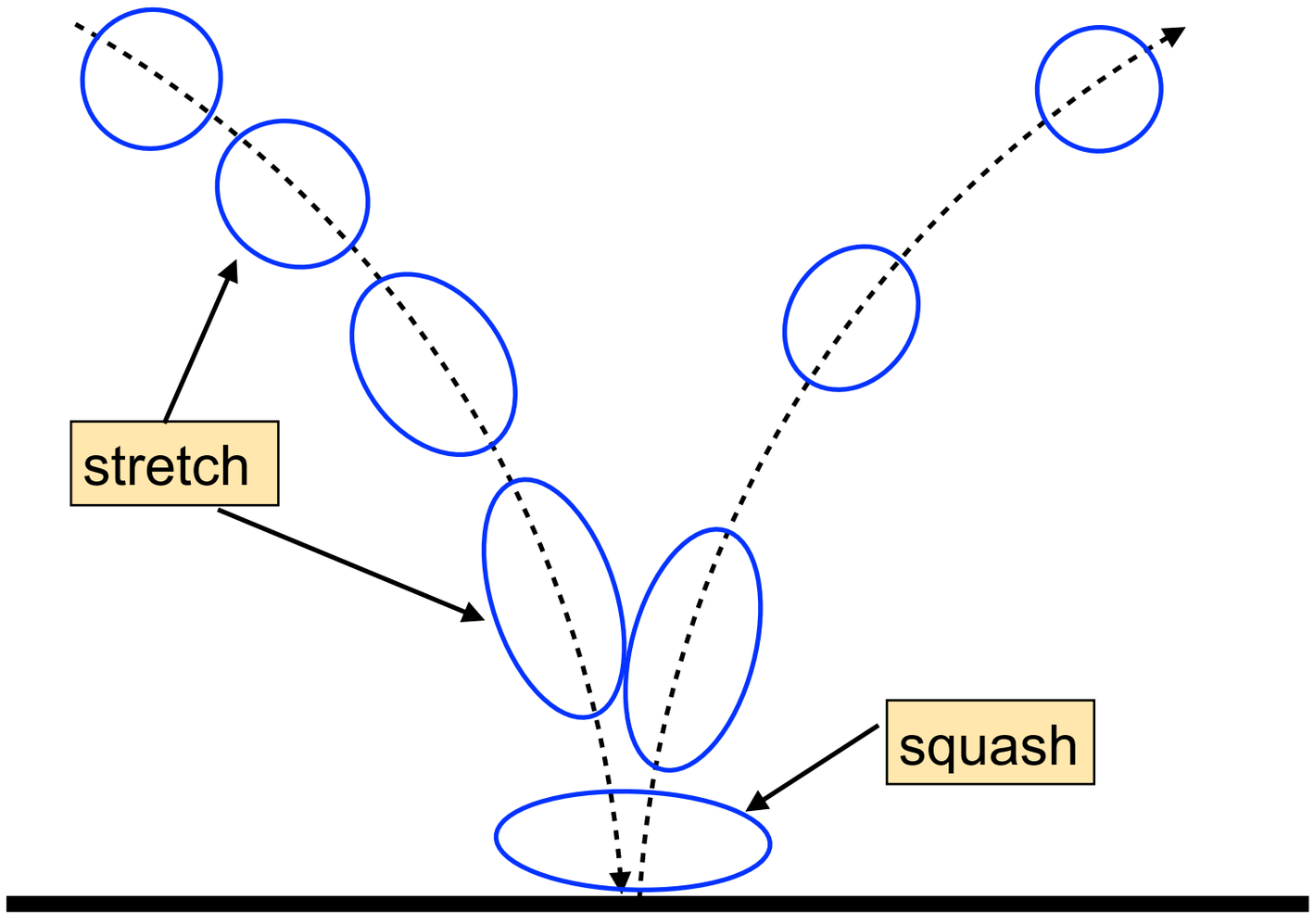


Content

- Simulating Acceleration and Velocity
- Morphing (Blend Shape Morphing)
- Morphing in OpenGL

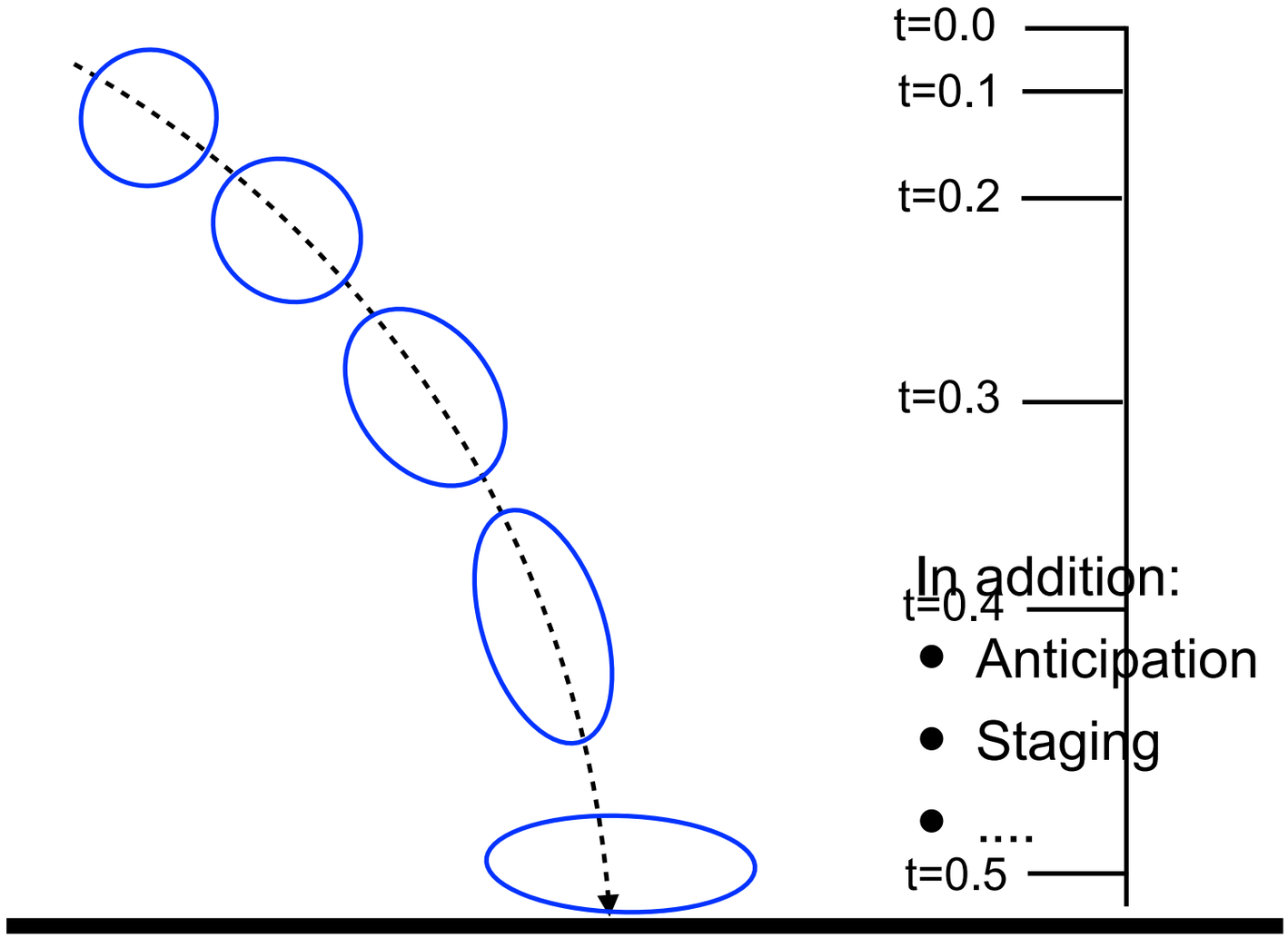
Squash and Stretch

An important technique to simulate acceleration, velocity, and changes in velocity, especially for non-rigid objects, is the squash and stretch technique



A bouncing ball

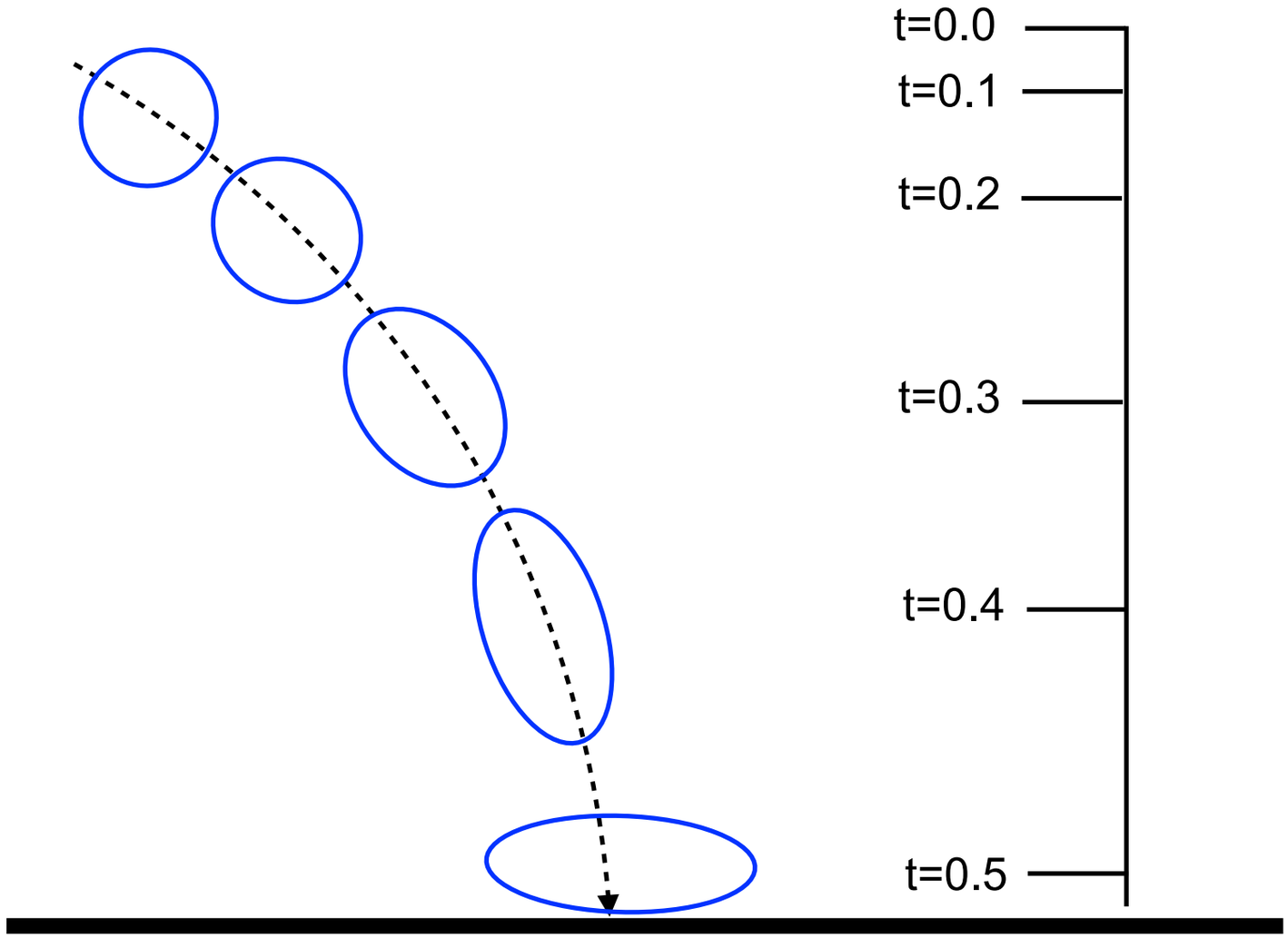
Timing



A bouncing ball

- A slower moving object is represented with more closely keyframes
- A fast moving object is represented with more keyframes

Timing



In addition:
- Anticipation
- Staging
- ....

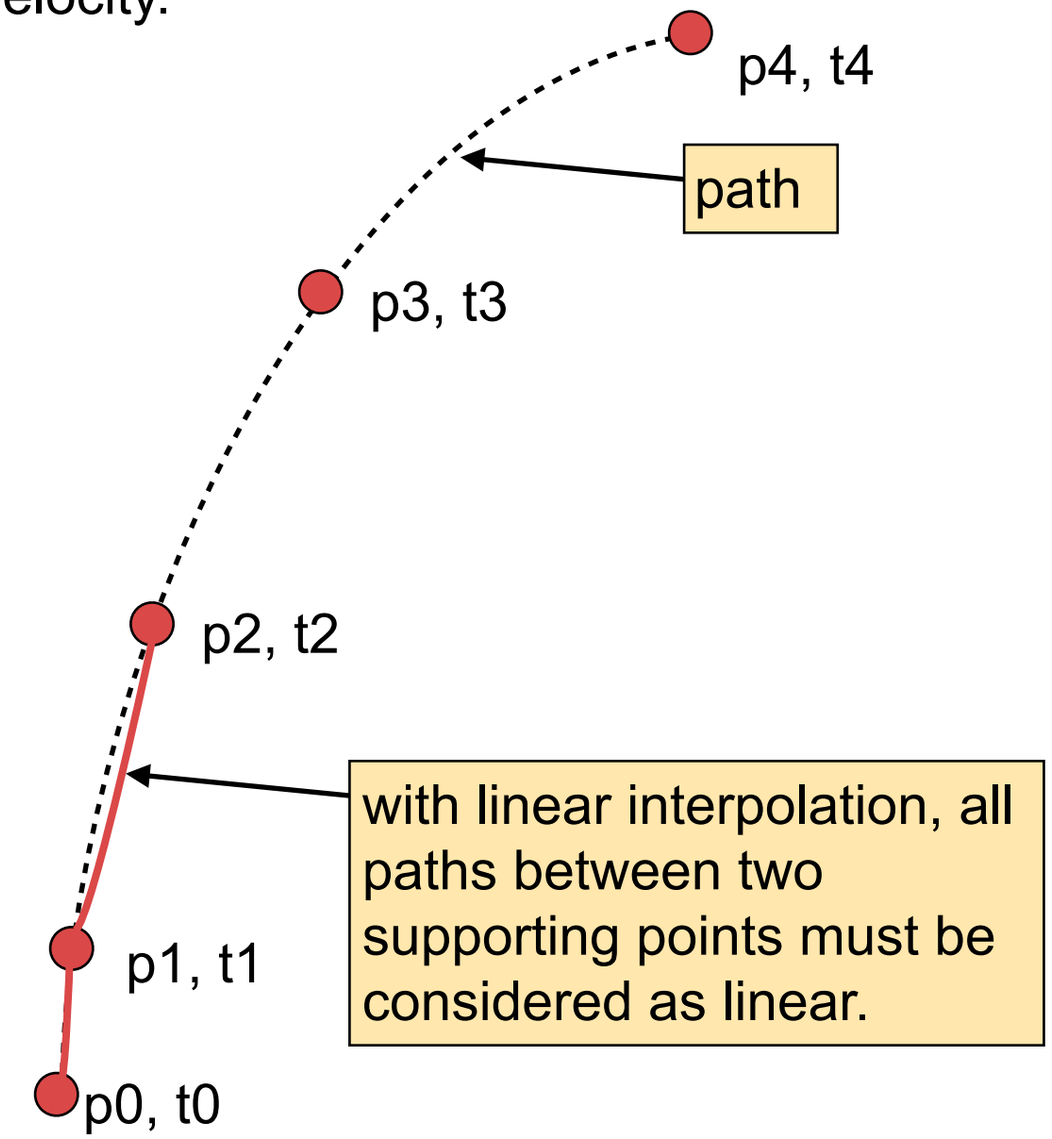
A bouncing ball

- A slower moving object is represented with more closely keyframes
- A fast moving object is represented with more keyframes

Constant Motion

Curve fitting techniques are often used to specify the animation paths between key frames.

A smooth 3D model motion requires equidistant supporting points in order to keep a constant velocity.

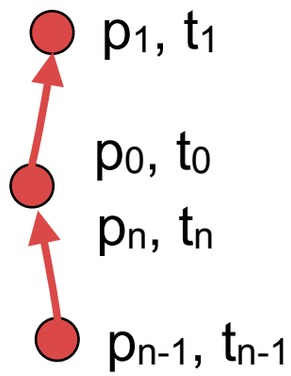


For a smooth motion:

$$\frac{|\mathbf{p}_1 - \mathbf{p}_0|}{t_1 - t_0} = \frac{|\mathbf{p}_2 - \mathbf{p}_1|}{t_2 - t_1} = \cdots = \frac{|\mathbf{p}_n - \mathbf{p}_{n-1}|}{t_n - t_{n-1}}$$

The velocity for all sections must be equal

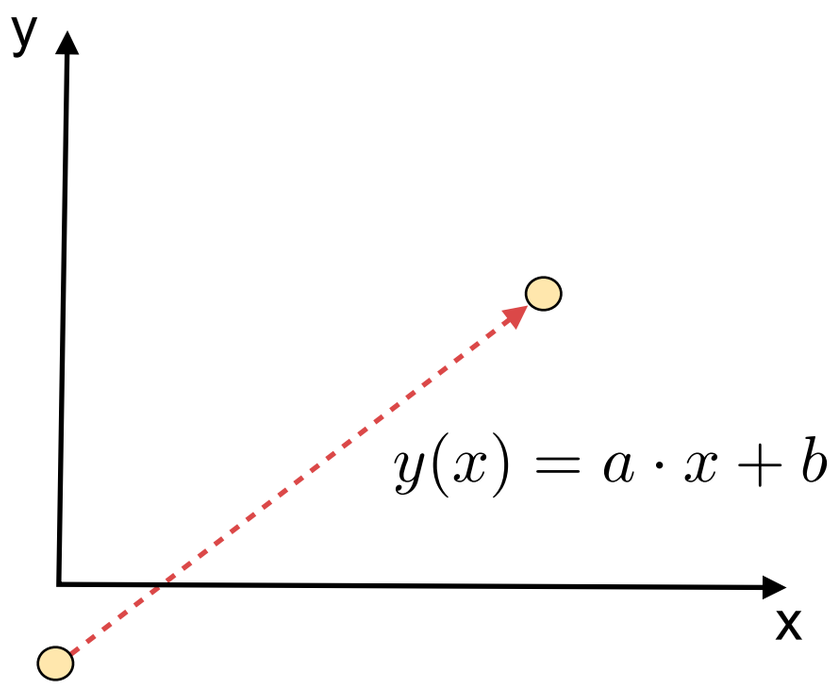
For a smooth loop:



- The first supporting point should also be the last point.
- The direction vector should not change suddenly.

px, tx: supporting point p for keyframe t

Calculating Supporting Points



For a smooth motion:

$$\frac{|\mathbf{p}_1 - \mathbf{p}_0|}{t_1 - t_0} = \frac{|\mathbf{p}_2 - \mathbf{p}_1|}{t_2 - t_1} = \ldots = \frac{|\mathbf{p}_n - \mathbf{p}_{n-1}|}{t_n - t_{n-1}}$$

with

$$\mathbf{p} = (x, y)$$

Try to use a rational function or its parametric representation to describe the path

Given:

- path with $y(x) = a \cdot x + b$, a and b are known.
- velocity of the object v
- distance between two points: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

$$\Rightarrow \frac{|\mathbf{p}_1 - \mathbf{p}_0|}{t_1 - t_0} = \frac{d}{t_1 - t_0} = v$$

Calculating Supporting Points

$$\Rightarrow \frac{d}{t_1 - t_0} = v$$

$$\Rightarrow \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{t_1 - t_0} = v$$

$$\Rightarrow \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = v(t_1 - t_0)$$

$$\Rightarrow (x_2 - x_1)^2 + (y_2 - y_1)^2 = v(t_1 - t_0)^2$$

$$\Rightarrow (x_2 - x_1)^2 + (y_2 - y_1)^2 = v^2\,\Delta t^2 \qquad \text{with} \quad \Delta t^2 = t_1 - t_0$$

$$\Rightarrow x_2^2 - 2x_2x_1 + x_1^2 + y_2^2 - 2y_2y_1 + y_1^2 = v^2\,\Delta t^2$$

with $(a - b)^2 = a^2 - 2ab + b^2$ (binomial series)

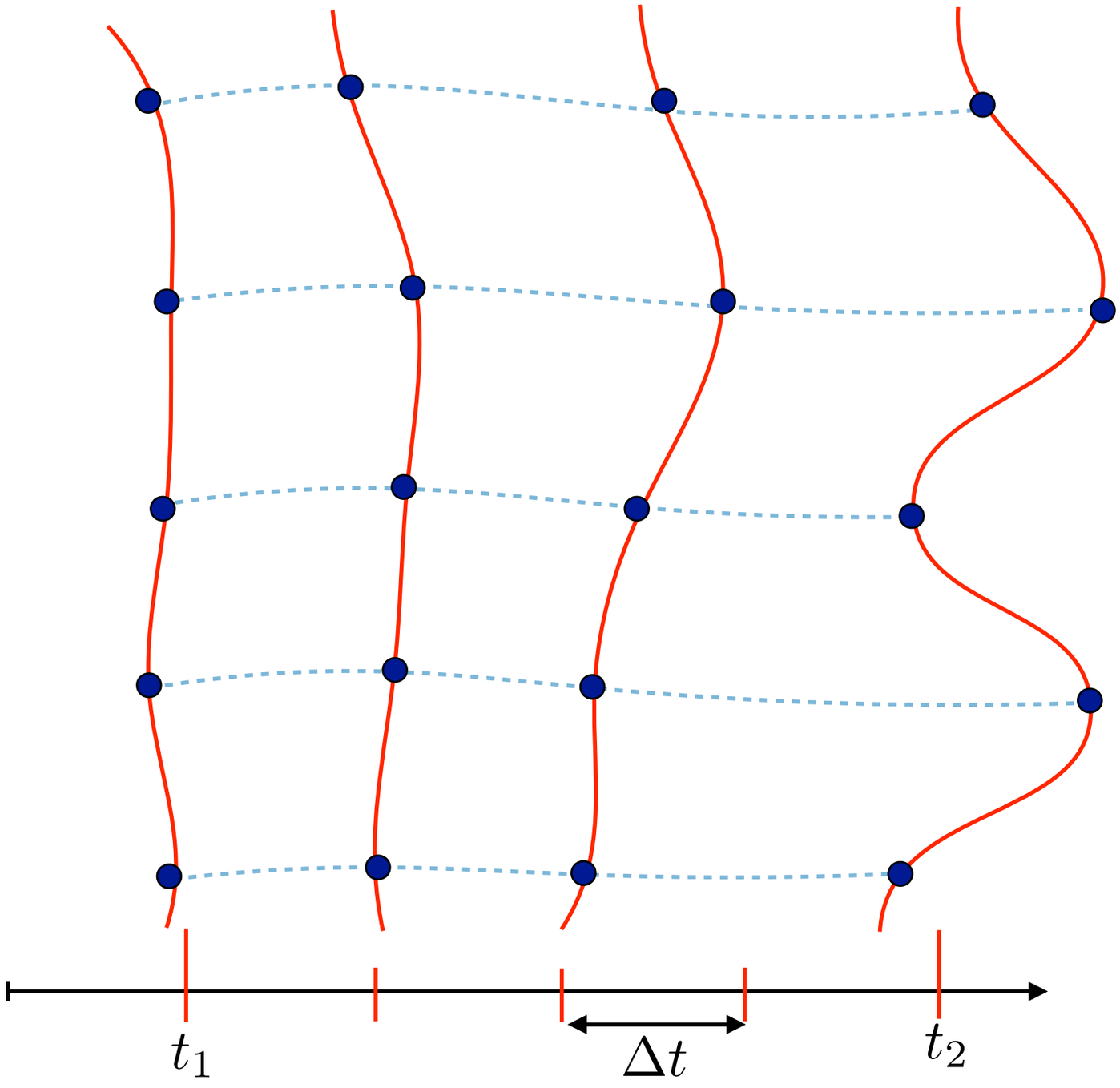
substitute y_2 with $y_2 = a \cdot x_2 + b$

....

Notes

- The path should be described by an explicit rational function.
- non-rational functions are also possible; it can be more difficult (impossible) to find a solution,
- If a path cannot be described with one equation, decompose the entire path in sub-paths.
- One animation vs. switching between animations:
  - one animation spare you from programing the “switching” code
  - multiple animations simplify replacing of a particular sub-path.
- The approach also works for 3D paths.

Constant Motion



For constant motion, divide the time interval between the keyframes in n+1 subintervals

$$\Delta t = \frac{t_2 - t_1}{n + 1}$$

The time for the *j*th frame is:

$$t = t_1 + j\ \Delta t$$

Acceleration

At some point, speed changes are necessary.

The start-up and slow-down portion of an animation path are often modeled with spline or trigonometric functions. Cubic and parabolic functions have also been used.

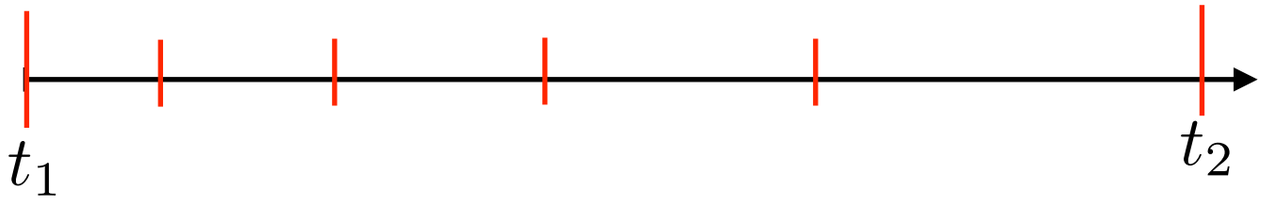
To model **increasing speed**, we want the time spacing between frames to increase.

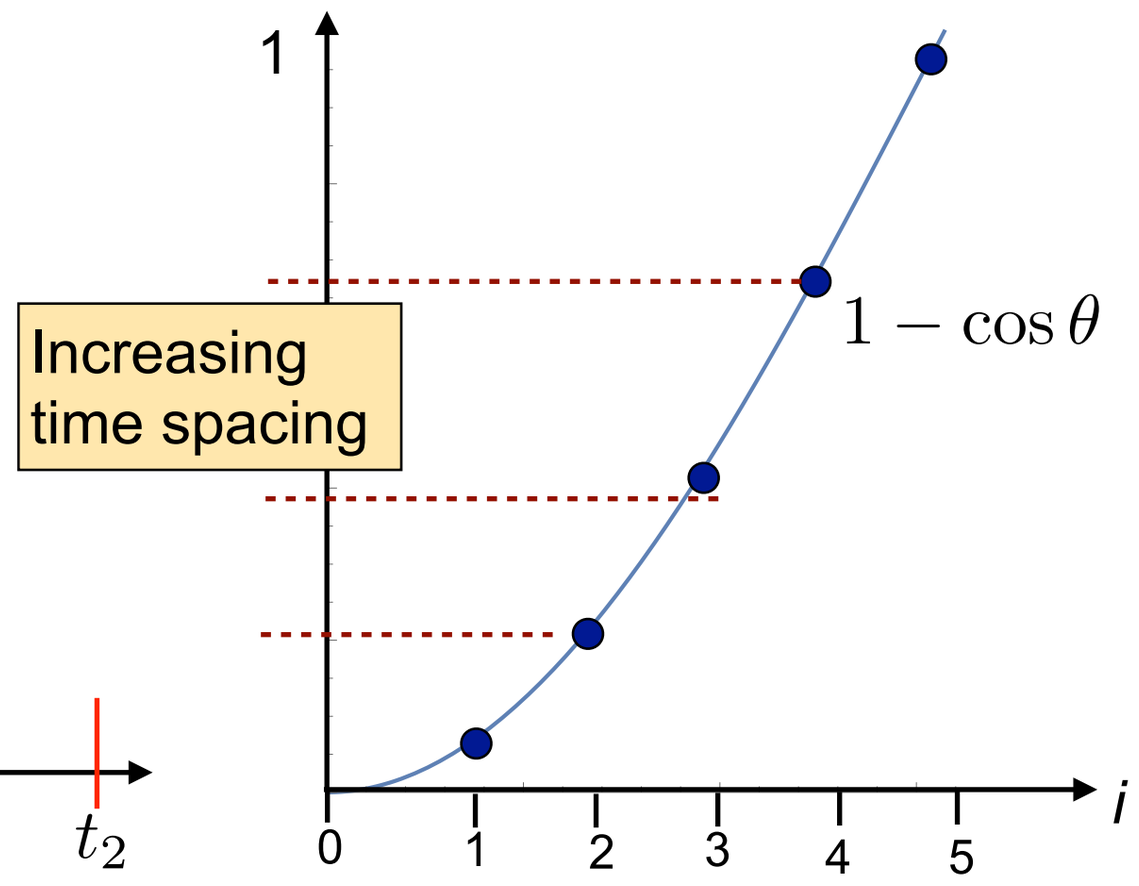
We can obtain an increasing size for the interval with the function:

$$1 - \cos\theta, \quad 0 < \theta < \pi/2$$

for the *i*th frame

$$t = t_1 + \Delta t \left[1 - \cos \frac{i\pi}{2(n+1)}\right],$$
$$i = 1, 2, 3, ..., n$$





Negative Acceleration

At some point, speed changes are necessary.

The start-up and slow-down portion of an animation path are often modeled with spline or trigonometric functions. Cubic and parabolic functions have also been used.

To model **decreasing speed**, we want the time spacing between frames to decrease.

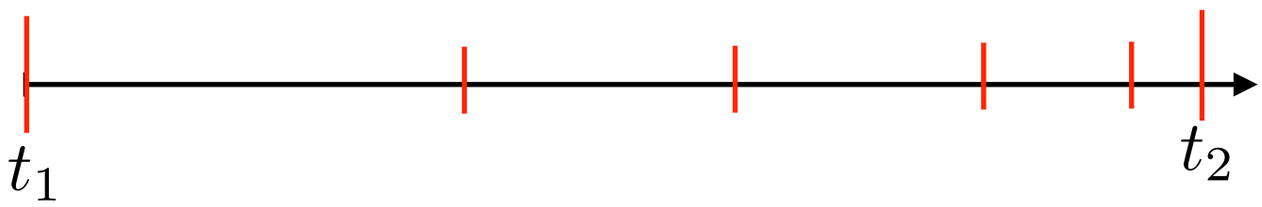
We can obtain an increasing size for the interval with the function:

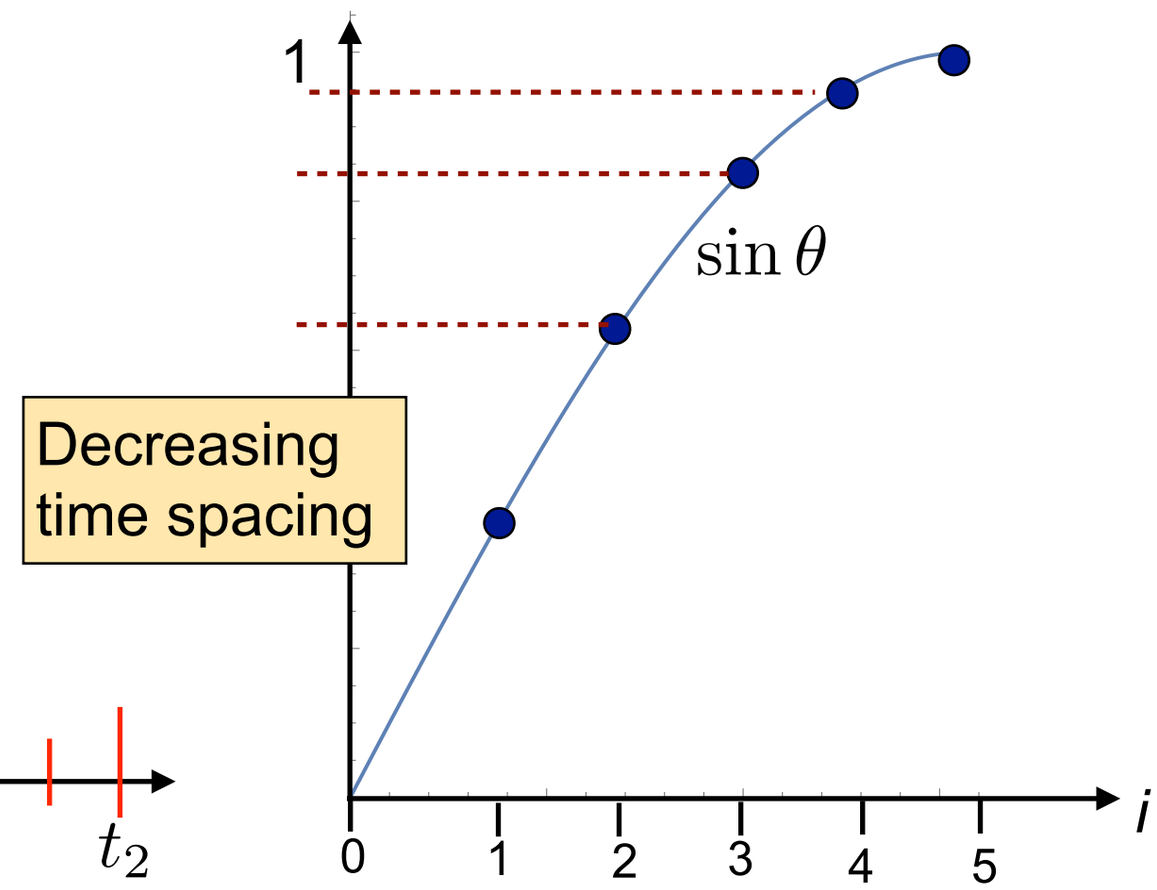
$$\sin\theta, \quad 0 < \theta < \pi/2$$

for the *i*th frame

$$t = t_1 + \Delta t \sin \frac{i\pi}{2(n+1)},$$

$$i = 1, 2, 3, ..., n$$





Both

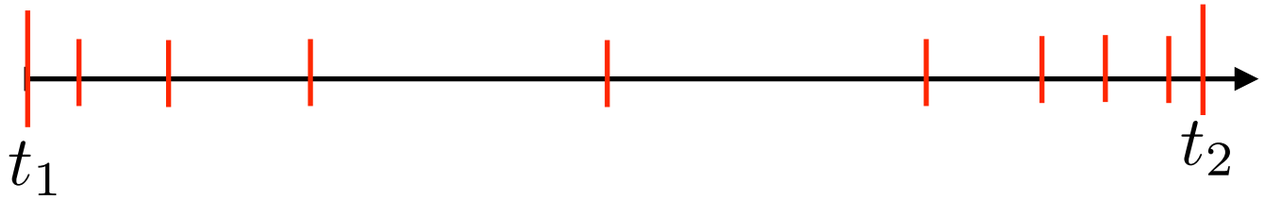
Motion often contain speedups and slowdowns. We can model this using:

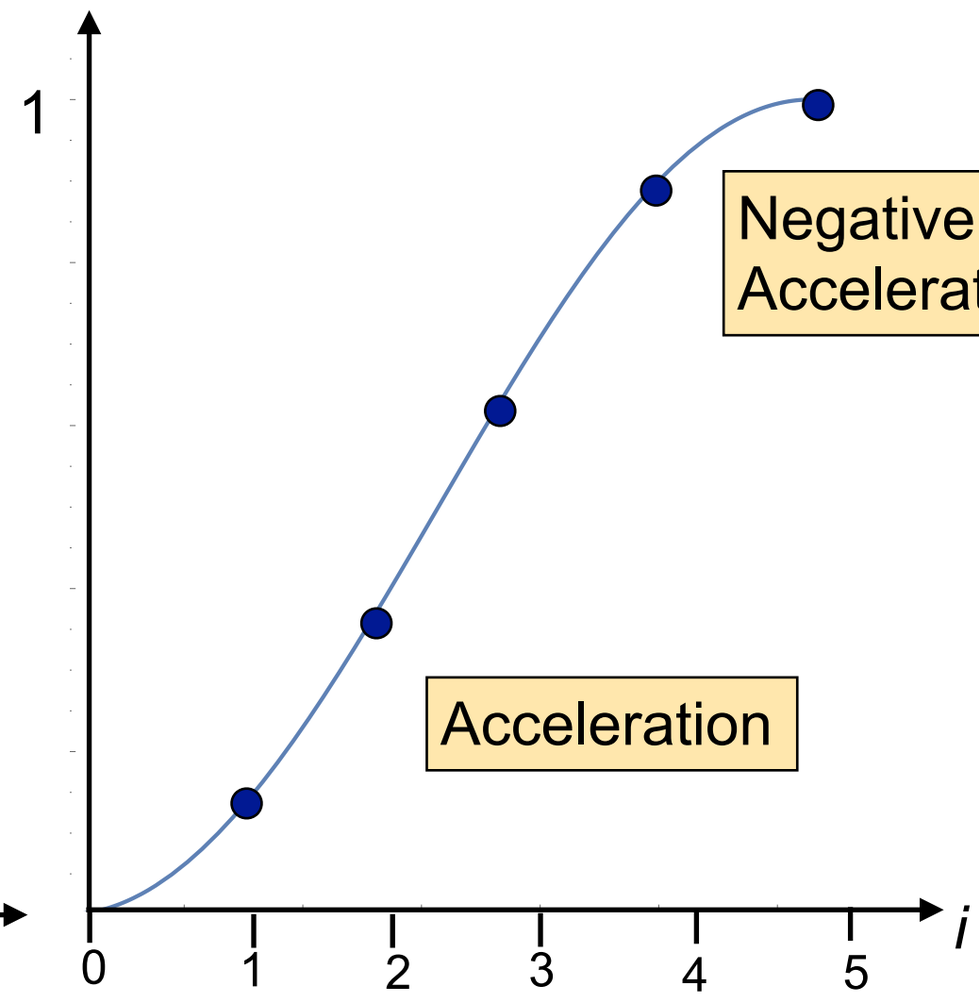
$$\frac{1}{2}(1-\cos\theta), \quad 0<\theta<\pi$$

with the time for frame i:

$$t = t_1 + \Delta t \left\{ \frac{1-\cos[i\pi/(n+1)]}{2} \right\},$$

$$i = 1, 2, 3, ..., n$$

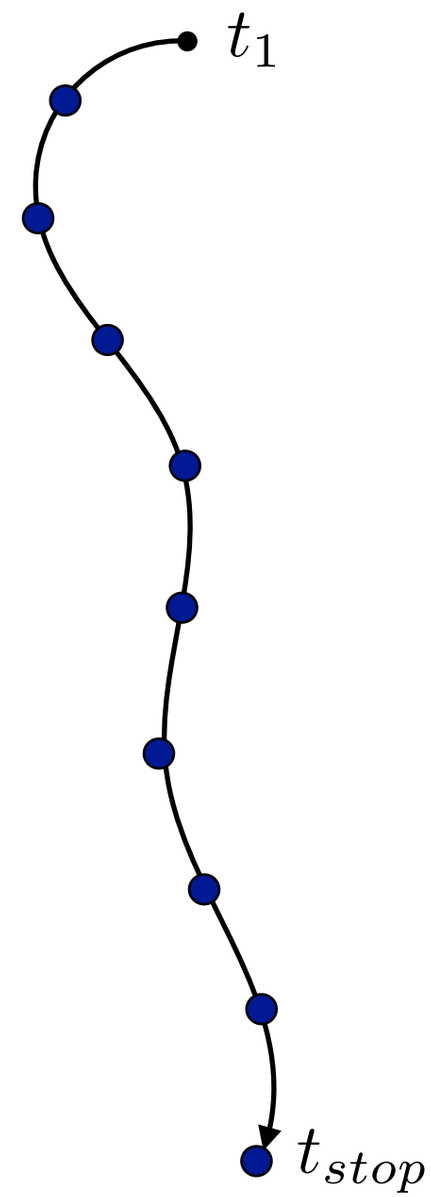




Planing an Animation Path

ARLAB

1. Plan your path for each body
2. Define a function to obtain keyframes (linear, cubic, spline, etc.)
3. Create a uniform distribution of keyframes: **Uniform position / orientation delta!**
4. Calculate the time fraction to meet the speedups and slowdown needs.

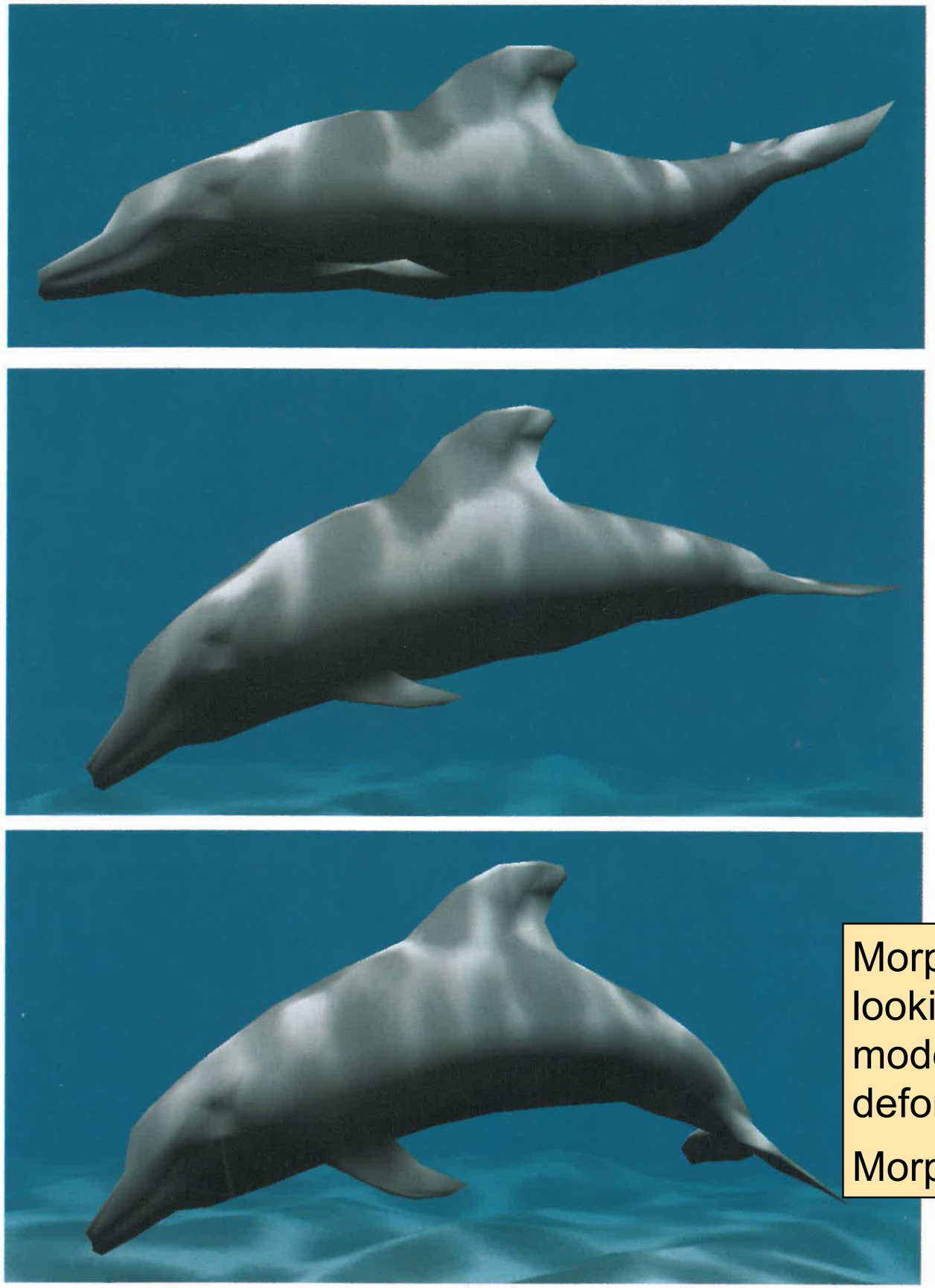


Morphing (Blend Shape Morphing)

ARLAB

VRAC|HCI

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

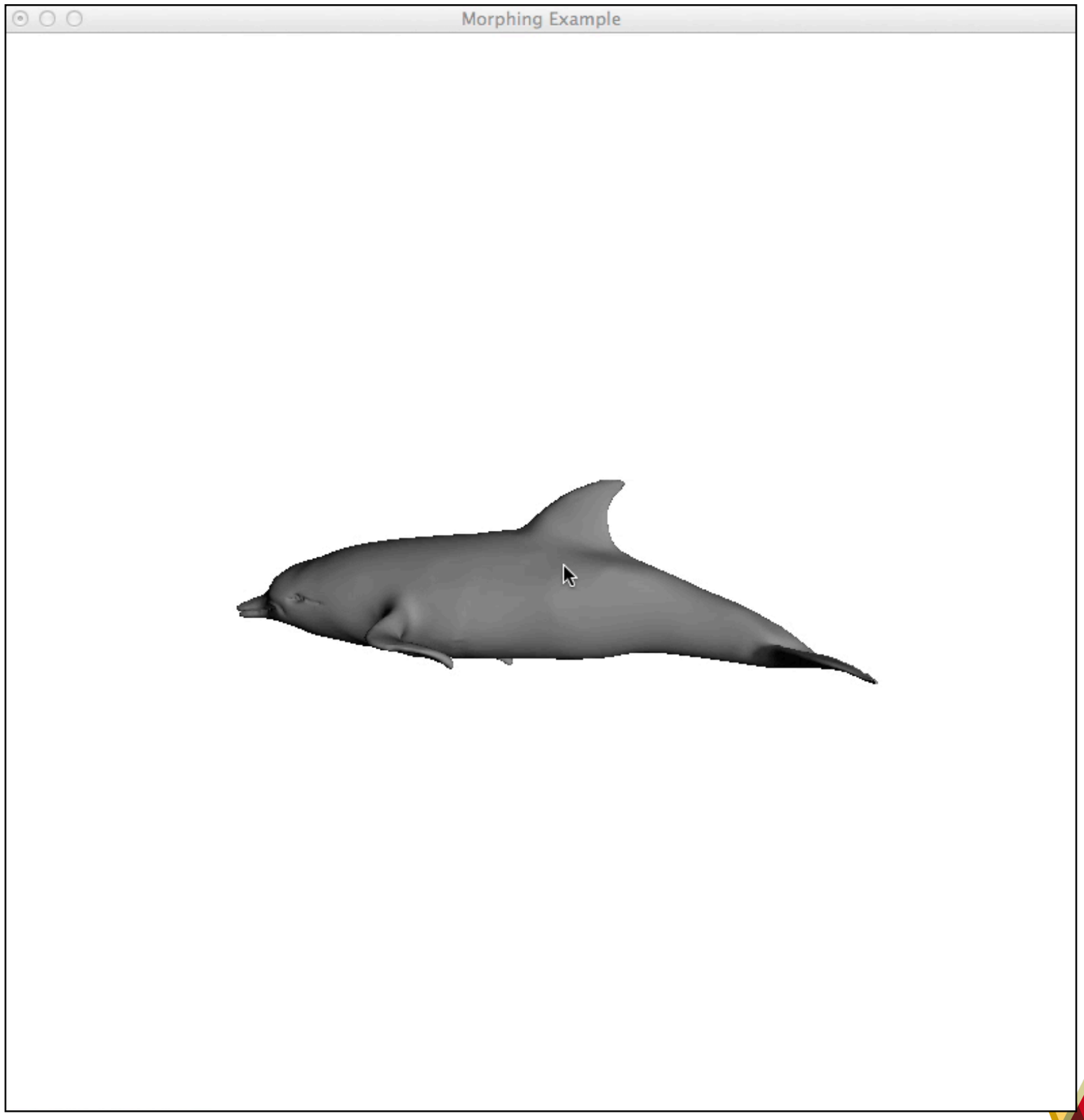


Morphing

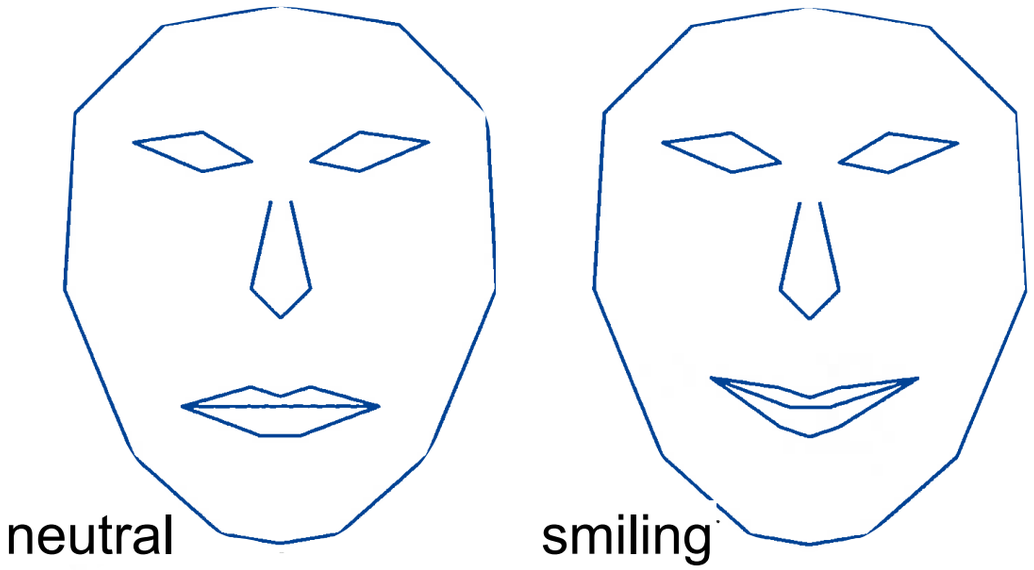
Morphing is a technique that facilitates natural looking motion. It deforms the mesh of a 3D model during runtime using one or multiple deformation models.

Morphing is a shorten term of metamorphosing

Example



Blend Shape Technique

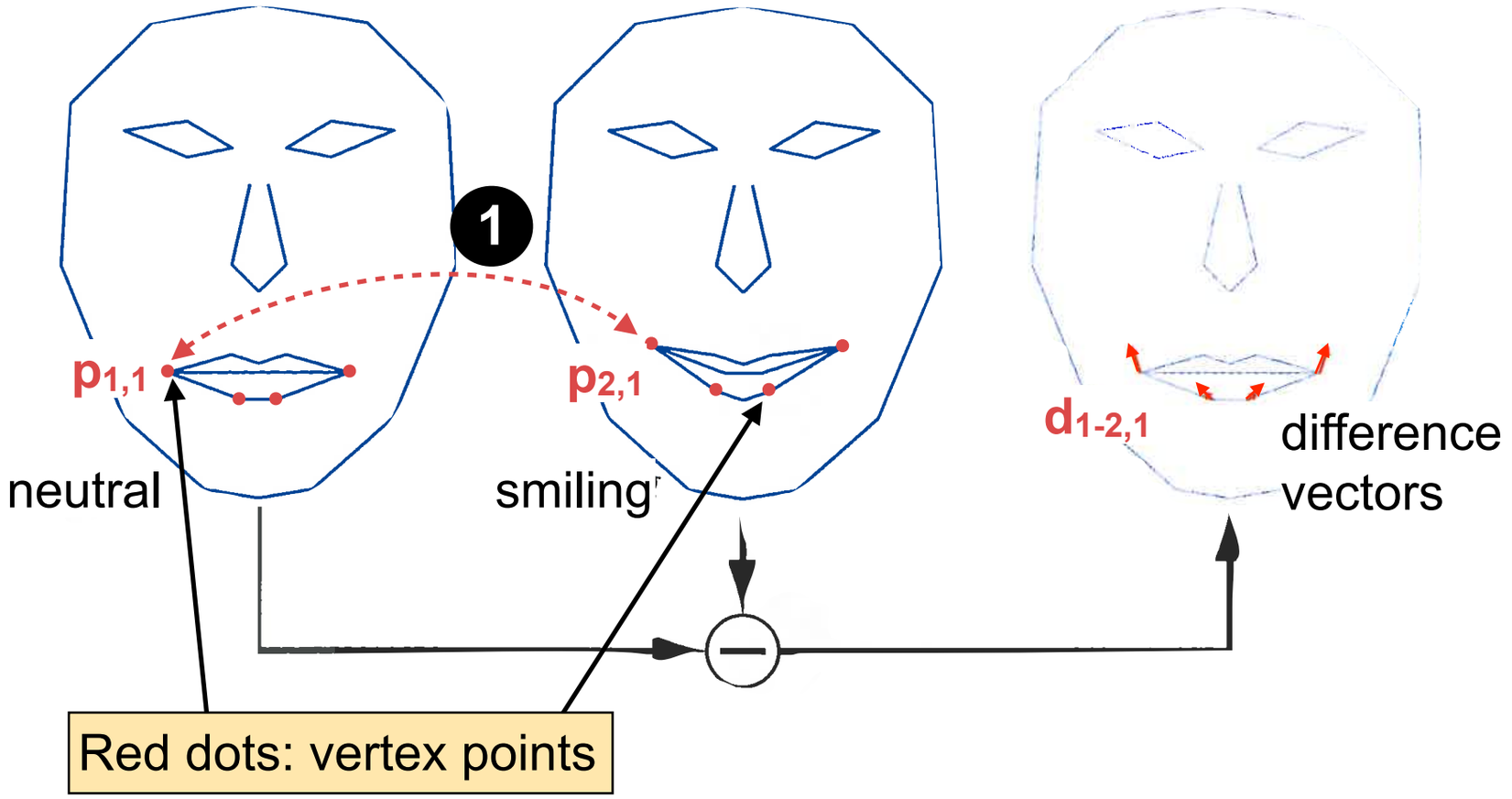


The Blend Shape Technique is a morphing technique that works with two or more 3D models of the object. Each model shows the object in a different pose.

Process:

1. Take (minimum) two 3D models of the object. The number of vertices must be equal. Declare one of these models as your “neutral” model
2. Calculate difference vectors between all associated vertices.
3. During runtime: add the difference vectors to your neutral model.

Calculate the Difference Vectors



The difference vectors store the differences / distance between all associated vertex points of the neutral model and the morphed model, in this case, the smiling face.

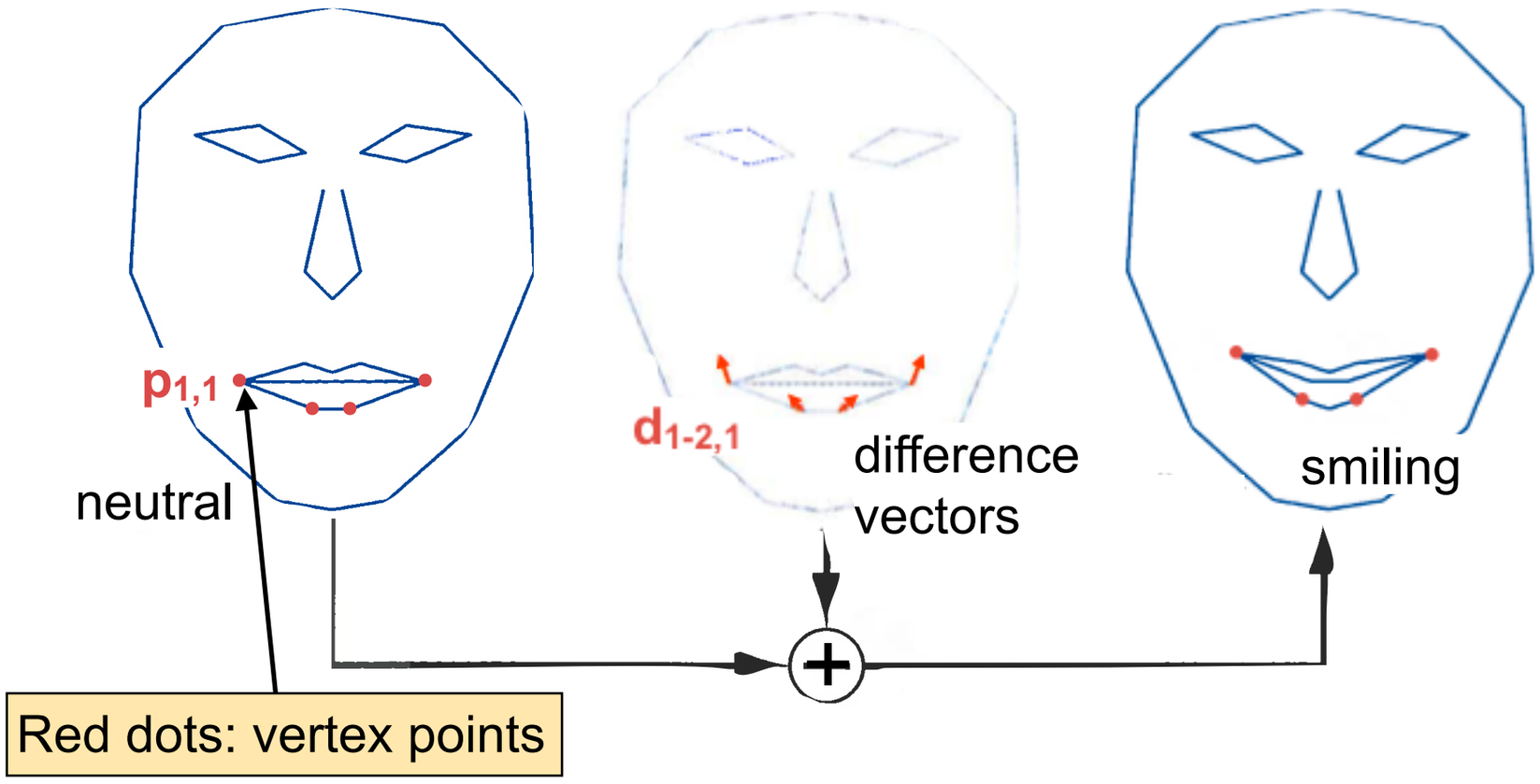
1 the association between the vertex points of two models must be known.

Calculate the difference vectors d_i for all vertex points i:

$$\mathbf{d}_i = \sum_{i=0}^{max} \mathbf{p}_{2,i} - \mathbf{p}_{1,i}$$

$\mathbf{p}_{2,i}$, $\mathbf{p}_{1,i}$: the vertex points i of the neutral model (1) and the smiling model (2)

During Runtime



Take the neutral vector as starting point. Add the difference vectors with a blend value to your neutral model:

$$\mathbf{p}_{out,i} = \sum_{i=0}^{max} \mathbf{p}_{1,i} + f \cdot \mathbf{d}_i$$

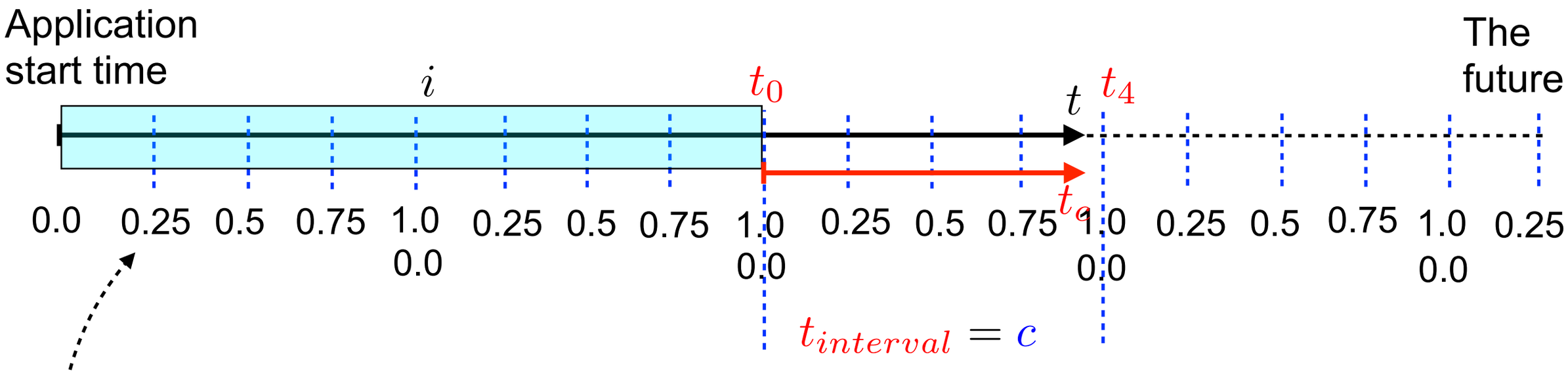
The parameter f is a blend value that runs from 0 to 1. If 0, the neutral model is shown, if 1, the smiling model is shown.

$\mathbf{p}_{2,i}$, $\mathbf{p}_{1,i}$: the vertex points i of the neutral model (1) and the smiling model (2)

$\mathbf{d}_i$: the difference vectors

$\mathbf{p}_{out,i}$: the new model

Calculating f



	Key	Position / Orientation
t_0	0.0	
	0.25	
	0.5	
	0.75	
t_4	1.0	

We have keyframes for five frames.

1. Calculate the number of past intervals *i*

$$i = \left\lfloor \frac{t}{t_{interval}} \right\rfloor$$

2. Calculate the time in the current interval:

$$t_c = t - t_{interval}\ i$$

3. Return the fraction as values between [0, 1]

$$f = \frac{t_c}{t_{interval}}$$

ARLAB

Morphing in OpenGL

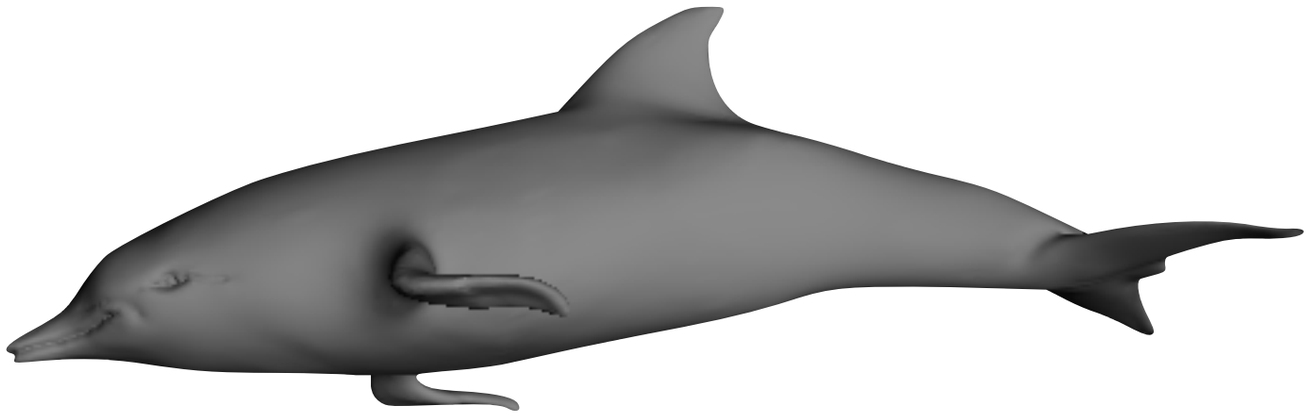
VRAC|HCI

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

3D Models

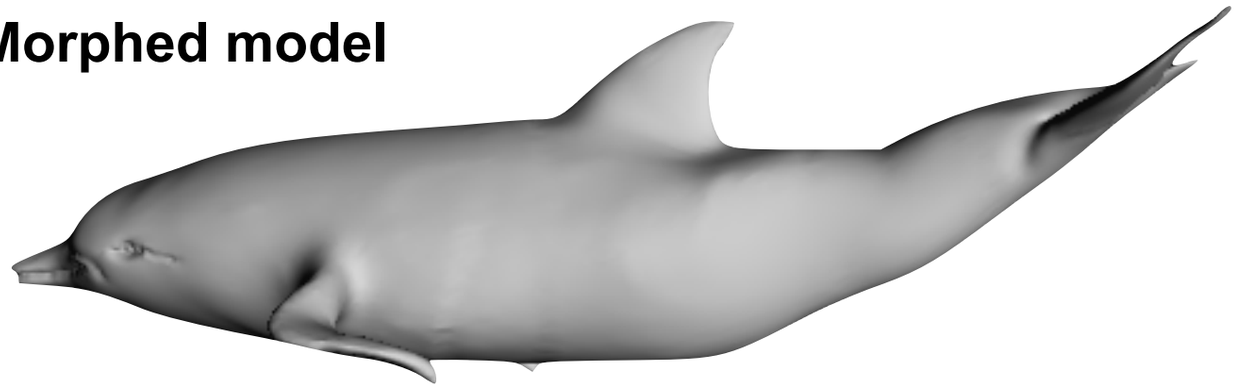
ARLAB

Neutral model



- Two 3D models are the minimum number of models.
- Both models are available as .obj files (or as files in a different file format).
- Both models show a different pose.
- Declare one model as **neutral model.**

Morphed model



VRAC|HCI

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Load the Models

```
void init_scene(void)
{
    // Read an obj file and load it, but not displayed yet
    if (!g_pmodel) {
        g_pmodel = new GLObjectObj("../data/dphn1.obj");
        if (!g_pmodel) exit(0);
```

The model that will be rendered at runtime.

The neutral model

```
    // Read an obj file and load it, but not displayed yet
    if (!g_pmodel2) {
        g_pmodel2 = new GLObjectObj("../data/dphn2.obj");
        if (!g_pmodel2) exit(0);
}
```

The morphed model

The Obj Loader was introduced in 20_Model_Loader

Calculate the Difference Vectors

The example function init_morphing shows how to calculate the difference vectors:

```
void init_morphing(void)
{
1   vertices_diff = (GLfloat*)malloc(sizeof(GLfloat) * 3 * (g_pmodel->numvertices + 1));

    for (int i = 1; i <= g_pmodel->numvertices; i++)
    {
2       vertices_diff[3 * i + 0]  = g_pmodel2->vertices[3*i+0] - g_pmodel1->vertices[3*i+0];
        vertices_diff[3 * i + 1]  = g_pmodel2->vertices[3*i+1] - g_pmodel1->vertices[3*i+1];
        vertices_diff[3 * i + 2]  = g_pmodel2->vertices[3*i+2] - g_pmodel1->vertices[3*i+2];
    }
}
```

1. Allocate memory for your difference vectors
2. Calculate the difference between all vectors

Calculate the Difference Vectors -
Allocate Memory

```
void* malloc (size_t size);
```

Allocates a block of size bytes of memory, returning a pointer to the beginning of the block.

Parameters:

- size: the amount of memory
- return void*: a pointer to the start element of this memory block.

Example:

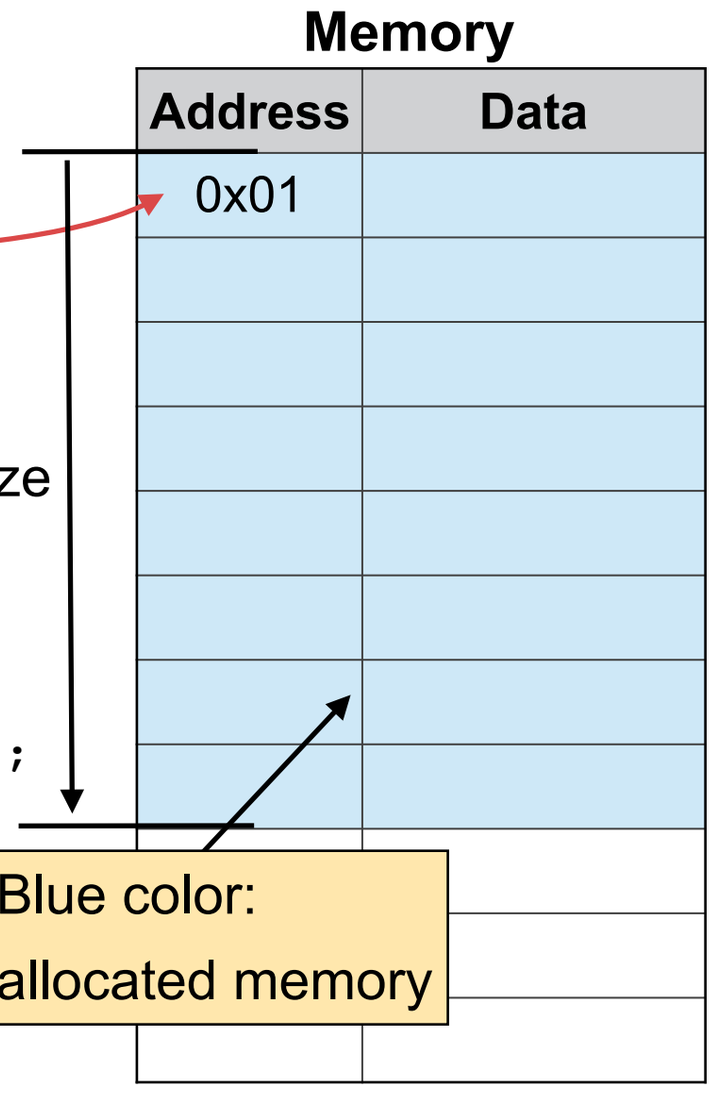
```
GLfloat* vertices1;
```

A float pointer that refers to the start of the allocated memory block

```
vertices1 =
(GLfloat*)malloc(sizeof(GLfloat) * 3 * (g_pmodel->numvertices + 1));
```

Every vertex point has **three** coordinates x, y, z. Every coordinate is a **4-byte float**

The overall number of vertices in this model + 1 buffer memory block



sizeof

`sizeof(`*`type`*`)`

Returns the size of an object in byte

Datatypes:

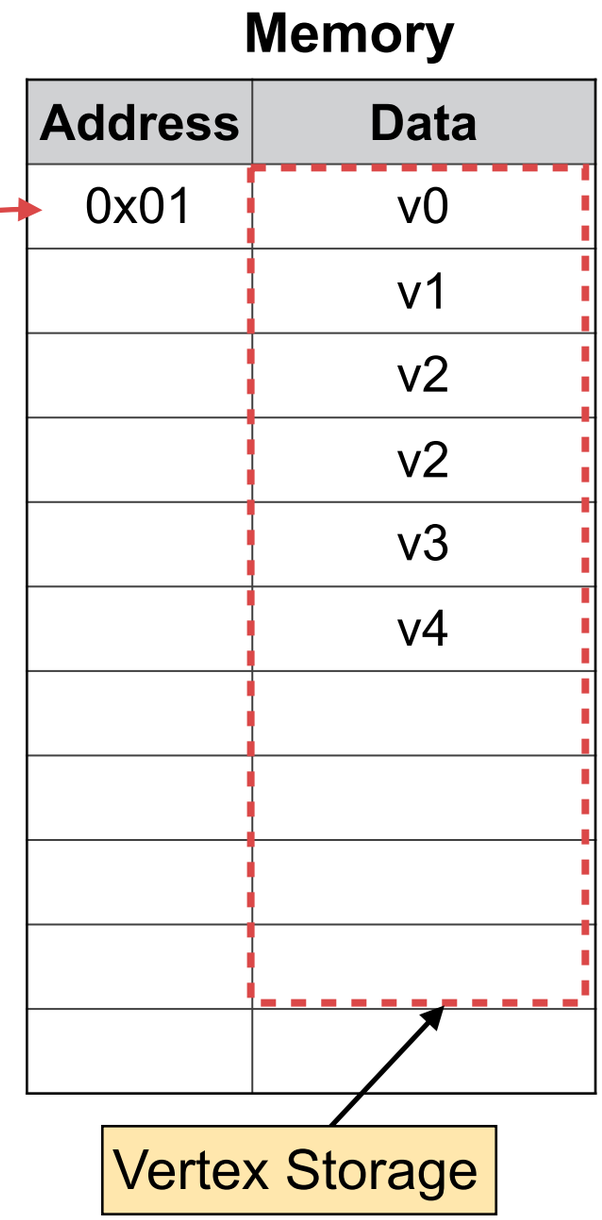
- short int - 2 bytes (16 bits)
- int int - 2 bytes (16 bits)
- long int - 4 bytes (32 bits)
- signed char - 1 byte (range -128 ... +127)
- unsigned char - 1 byte (range   0 ... 255)
- float - 4 bytes
- double - 8 bytes
- long double - 8 bytes

The model

```
//// Model loading variables
GLMmodel* g_pmodel = NULL;
```

Model structure:

```
class GLmodelObj {
  char*    pathname;            /* path to this model */
  char*    mtllibname;          /* name of the material library */

  GLuint   numvertices;         /* number of vertices in model */
  GLfloat* vertices;            /* array of vertices  */

  GLuint   numnormals;          /* number of normals in model */
  GLfloat* normals;             /* array of normals */

  GLuint   numtexcoords;        /* number of texcoords in model */
  GLfloat* texcoords;           /* array of texture coordinates */
```



The Obj Model provides GLfloat pointer *vertices* that refers to the starting point of all vertices in memory.

The variable *numvertices* stores the overall number of vertices of this particular model.

Calculate the Difference Vectors

Use a for-loop to run through all vertices of both and subtract their positions.

```
for (int i = 1; i <= g_pmodel->numvertices; i++)
{
    vertices_diff[3 * i + 0]  = g_pmodel2->vertices[3 * i + 0] - g_pmodel1->vertices[3 * i + 0];
    vertices_diff[3 * i + 1]  = g_pmodel2->vertices[3 * i + 1] - g_pmodel1->vertices[3 * i + 1];
    vertices_diff[3 * i + 2]  = g_pmodel2->vertices[3 * i + 2] - g_pmodel1->vertices[3 * i + 2];
}
```

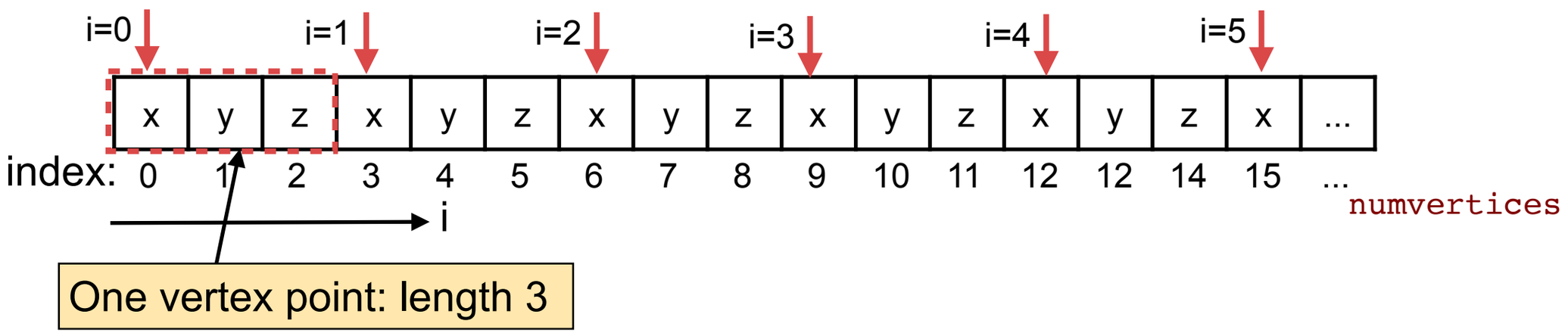
Vertex points of the morphed model

Vertex points of the neutral model

Memory

Addres	Data
0x01	v0
	v1
	v2
	v2
	v3
	v4

Consider the memory as long array:



During Runtime

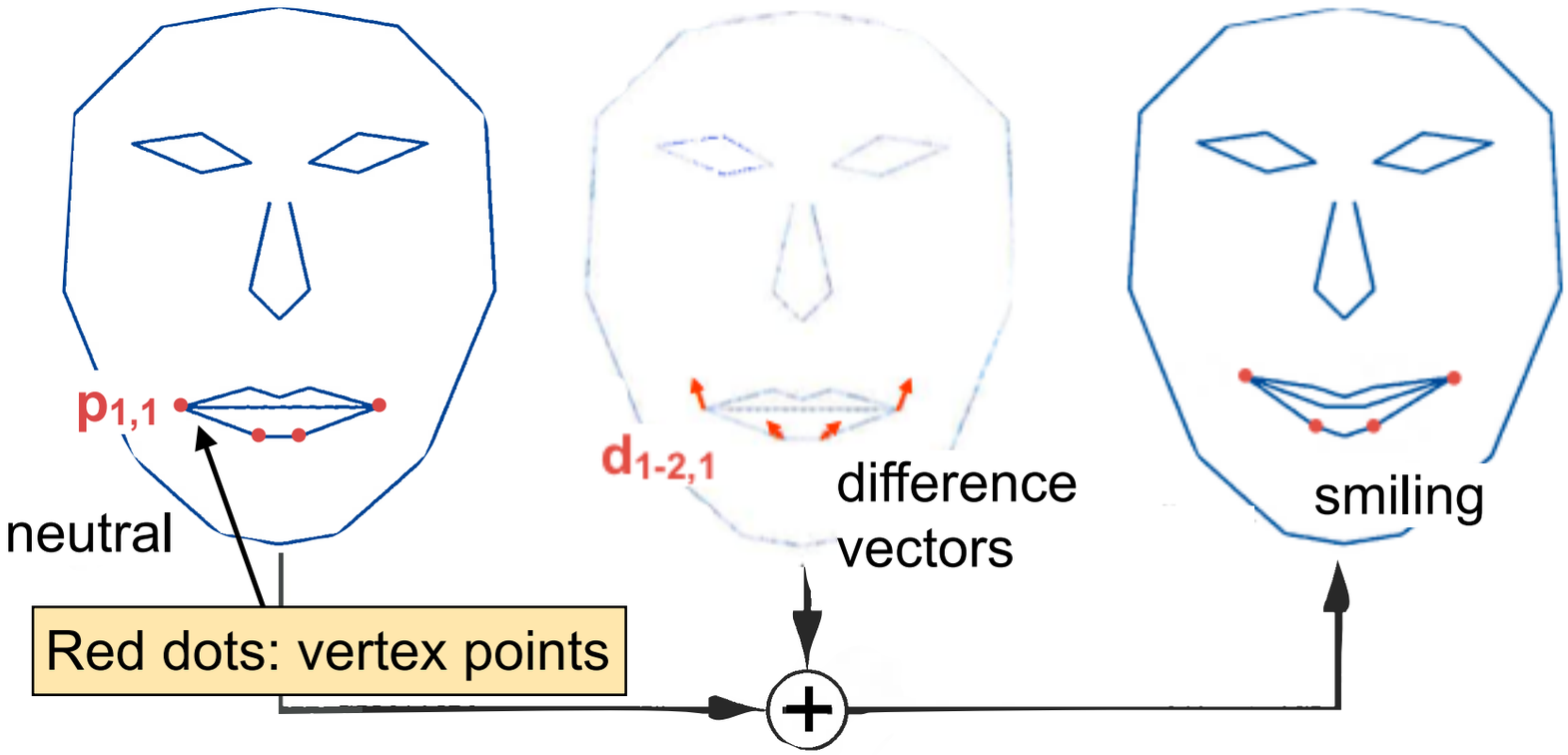
```
void myAnimiationCallback(int value)
{
    // Fetch the current time
    float time_0 = (float)clock()/CLOCKS_PER_SEC;

    // calculate the time offset
    float time_offset = (time_0) /animation_time;

    if((time_0 - g_time_1) >= 0.015)
    {
        // get the fraction
        float integral;
        float fractional = std::modf(time_offset, &integral);

        //--------------------------------------------------------------
        // Morph the object
        for (int i = 1; i <= g_pmodel->numvertices; i++)
        {
            g_pmodel->vertices[3*i+0] = g_pmodel1->vertices[3*i+0] + fractional * vertices_diff[3*i+0];
            g_pmodel->vertices[3*i+1] = g_pmodel1->vertices[3*i+1] + fractional * vertices_diff[3*i+1];
            g_pmodel->vertices[3*i+2] = g_pmodel1->vertices[3*i+2] + fractional * vertices_diff[3*i+2];
        }

        // Update the models vertices
       [....]

        g_time_1 = time_0;
    }
}
```

Calculate the fractional part

Use a for loop to run through all vertices and add a fraction of the difference vector with respect of the animation time.

During Runtime

ARLAB



$$\mathbf{p}_{out,i} = \sum_{i=0}^{max} \mathbf{p}_{1,i} + f \cdot \mathbf{d}_i$$

```
// Morph the object
for (int i = 1; i <= g_pmodel->numvertices; i++)
{
    g_pmodel->vertices[3*i+0] = g_pmodel1->vertices[3*i+0] + fractional * vertices_diff[3*i+0];
    g_pmodel->vertices[3*i+1] = g_pmodel1->vertices[3*i+1] + fractional * vertices_diff[3*i+1];
    g_pmodel->vertices[3*i+2] = g_pmodel1->vertices[3*i+2] + fractional * vertices_diff[3*i+2];
}
```

The vertices of the neutral model

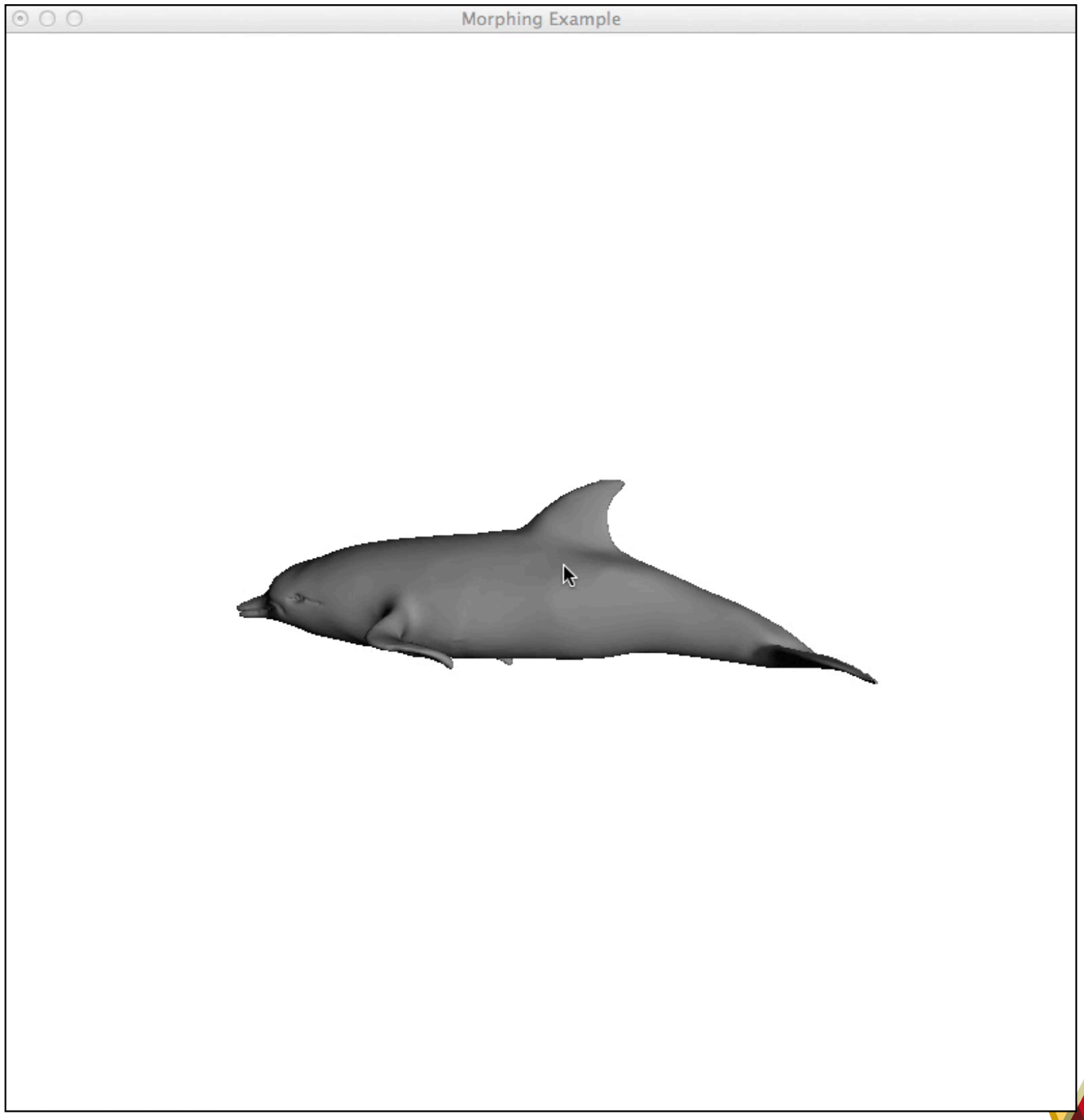
The fractional part

The difference vectors.

VRAC|HCI

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Example



Thank you!

Questions

Rafael Radkowski, Ph.D.
Iowa State University
Virtual Reality Applications Center
1620 Howe Hall
Ames, Iowa 50011, USA

+1 515.294.5580
+1 515.294.5530(fax)

rafael@iastate.edu
http://arlab.me.iastate.edu

www.linkedin.com/in/rradkowski

ARLAB

VRAC|HCI

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY