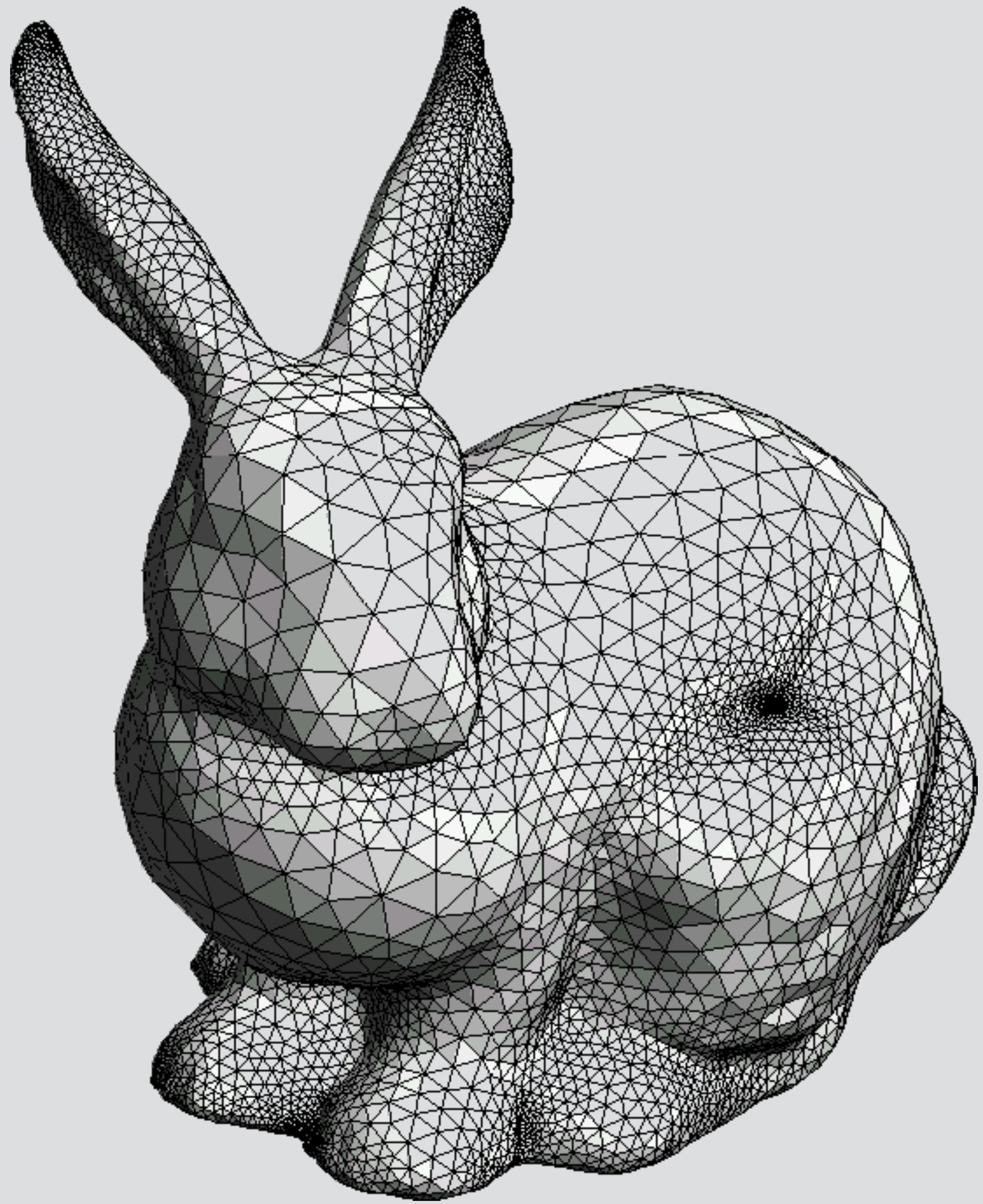
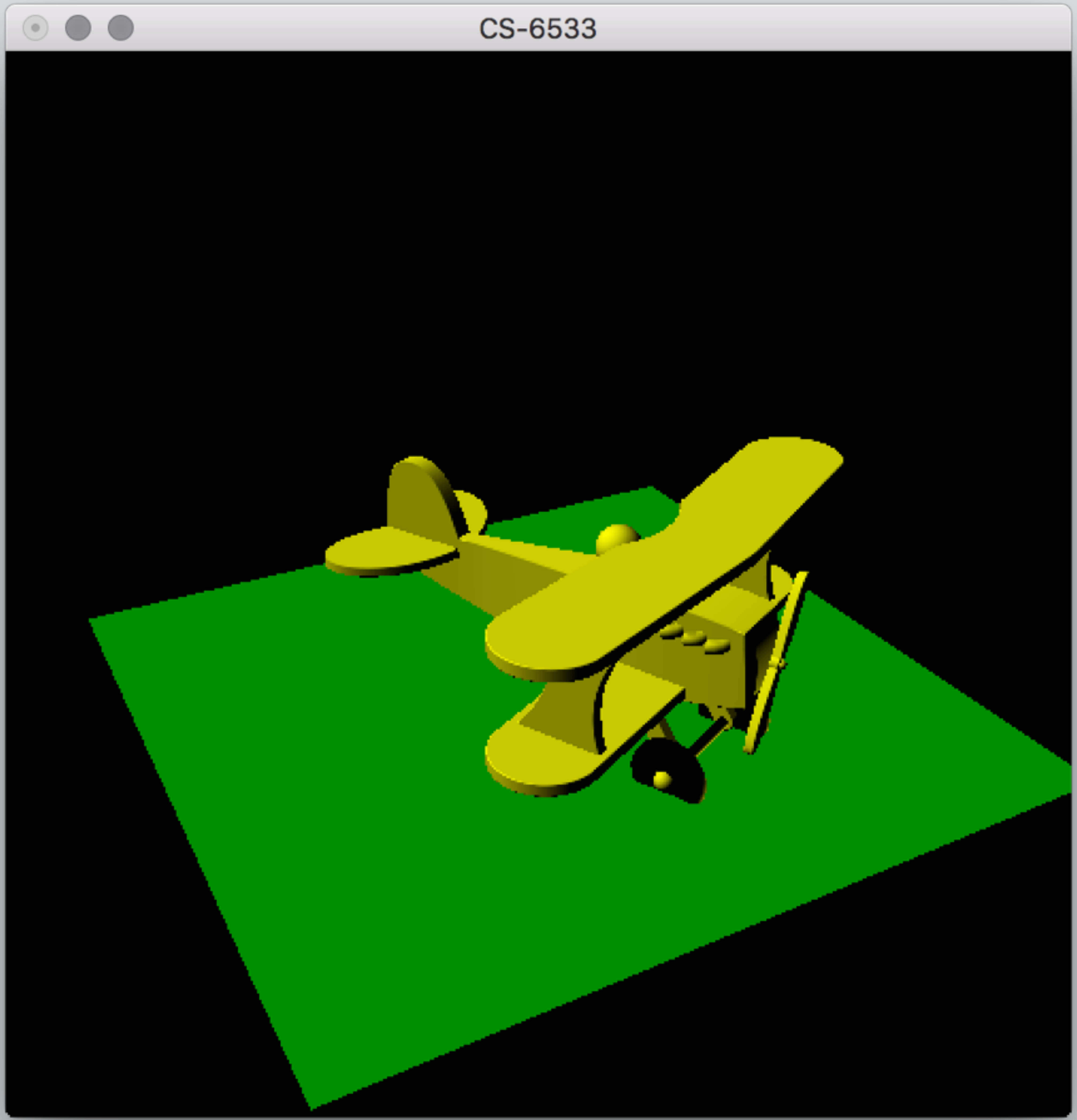
Camera and Animation

CS GY-6533 / UY-4533

Mesh Loading

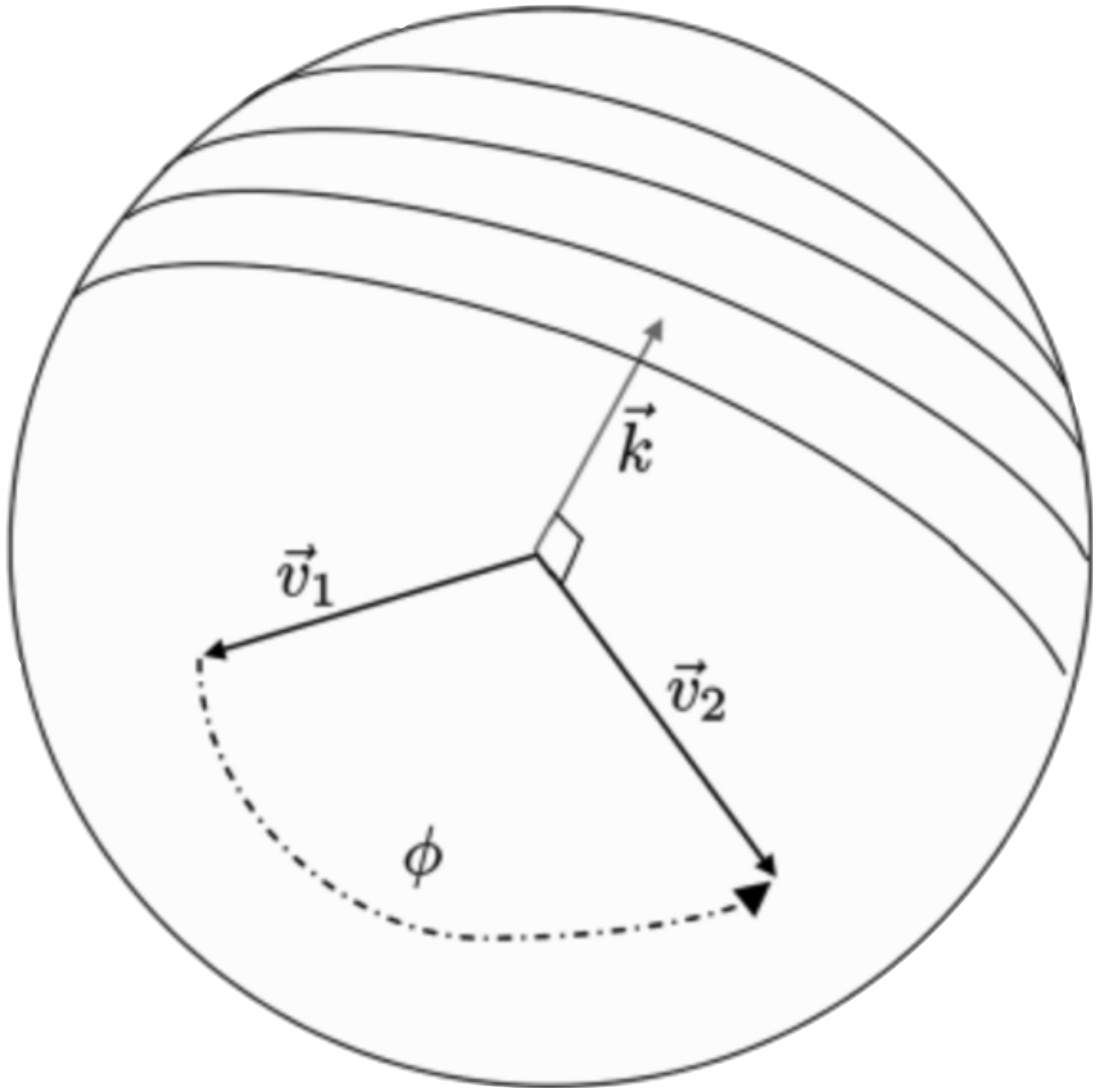



CS-6533

```cpp
#define TINYOBJLOADER_IMPLEMENTATION
#include “tiny_obj_loader.h"

////

void loadObjFile(const std::string &fileName, std::vector<VertexPN> &outVertices, std::vector<unsigned
short> &outIndices) {
    tinyobj::attrib_t attrib;
    std::vector<tinyobj::shape_t> shapes;
    std::vector<tinyobj::material_t> materials;
    std::string err;

    bool ret = tinyobj::LoadObj(&attrib, &shapes, &materials, &err, fileName.c_str(), NULL, true);

    if(ret) {
        for(int i=0; i < attrib.vertices.size(); i+=3) {
            VertexPN v;
            v.p[0] = attrib.vertices[i];
            v.p[1] = attrib.vertices[i+1];
            v.p[2] = attrib.vertices[i+2];
            v.n[0] = attrib.normals[i];
            v.n[1] = attrib.normals[i+1];
            v.n[2] = attrib.normals[i+2];
            outVertices.push_back(v);
        }
        for(int i=0; i < shapes.size(); i++) {
            for(int j=0; j < shapes[i].mesh.indices.size(); j++) {
                outIndices.push_back(shapes[i].mesh.indices[j].vertex_index);
            }
        }
    } else {
        std::cout << err << std::endl;
        assert(false);
    }
}
```

```
std::vector<VertexPN> meshVertices;
std::vector<unsigned short> meshIndices;

loadObjFile("toyplane.obj", meshVertices, meshIndices);
```

Arcball and Trackball



$$\phi = \arccos(\vec{v}_1 \cdot \vec{v}_2)$$

$$\vec{k} = \text{normalize}(\vec{v}_1 \times \vec{v}_2)$$

$$\begin{bmatrix} \cos(\phi) \\ \sin(\phi)\hat{\mathbf{k}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{v}}_1 \cdot \hat{\mathbf{v}}_2 \\ \hat{\mathbf{v}}_1 \times \hat{\mathbf{v}}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \hat{\mathbf{v}}_2 \end{bmatrix} \begin{bmatrix} 0 \\ -\hat{\mathbf{v}}_1 \end{bmatrix},$$

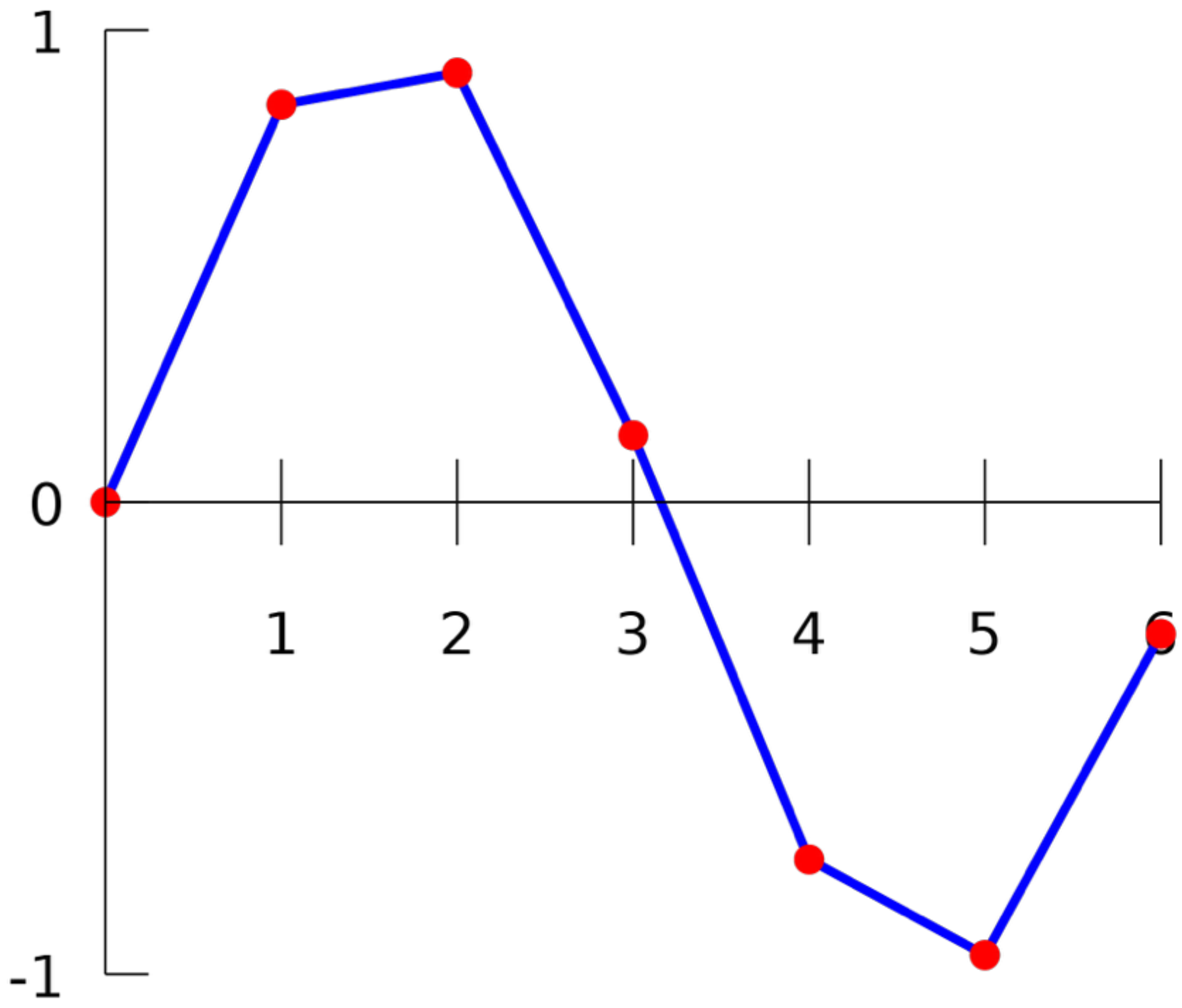
$$(x - c_x)^2 + (y - c_y)^2 + (z - 0)^2 - r^2 = 0,$$

$[c_x, c_y, 0]^t$ - screen center of the sphere and **r** is radius of sphere in pixels

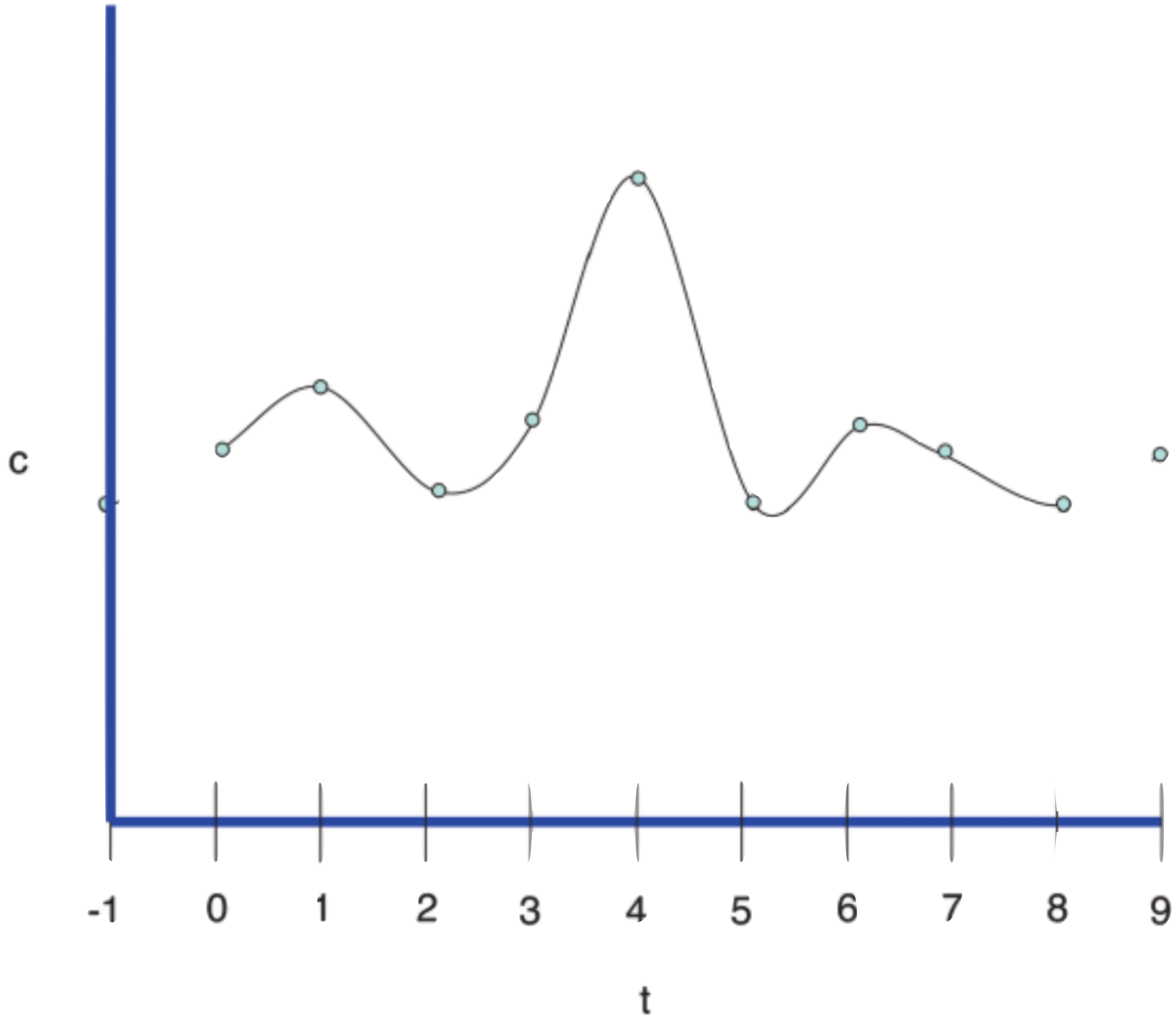
Interpolation

Linear interpolation (LERP)

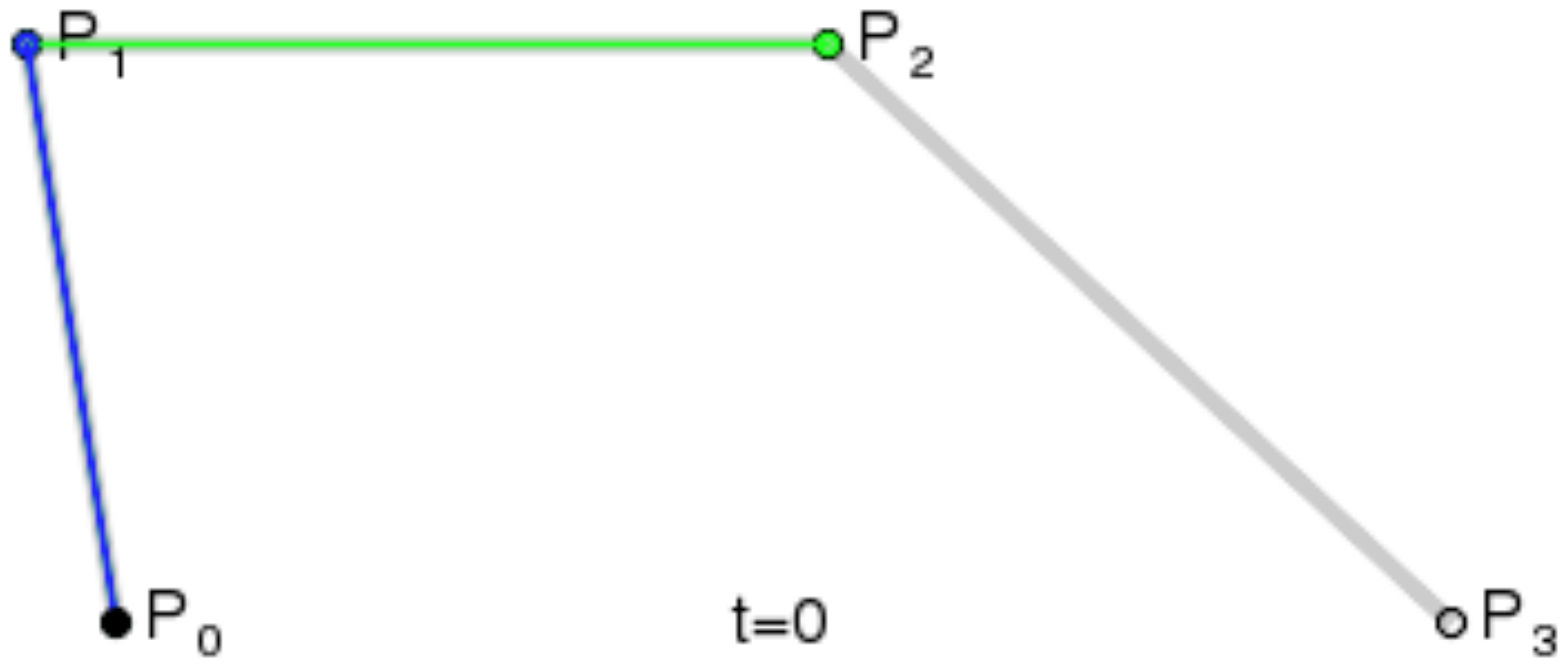
```
float lerp(float v0, float v1, float t) {
    return (1.0-t)*v0 + t*v1;
}
```


1
0
-1
1
2
3
4
5
6

Smooth interpolation


c
-1
0
1
2
3
4
5
6
7
8
9
t

Cubic bezier functions


P1
P2
P0
t=0
P3

$$f = (1-t)c_0 + td_0$$

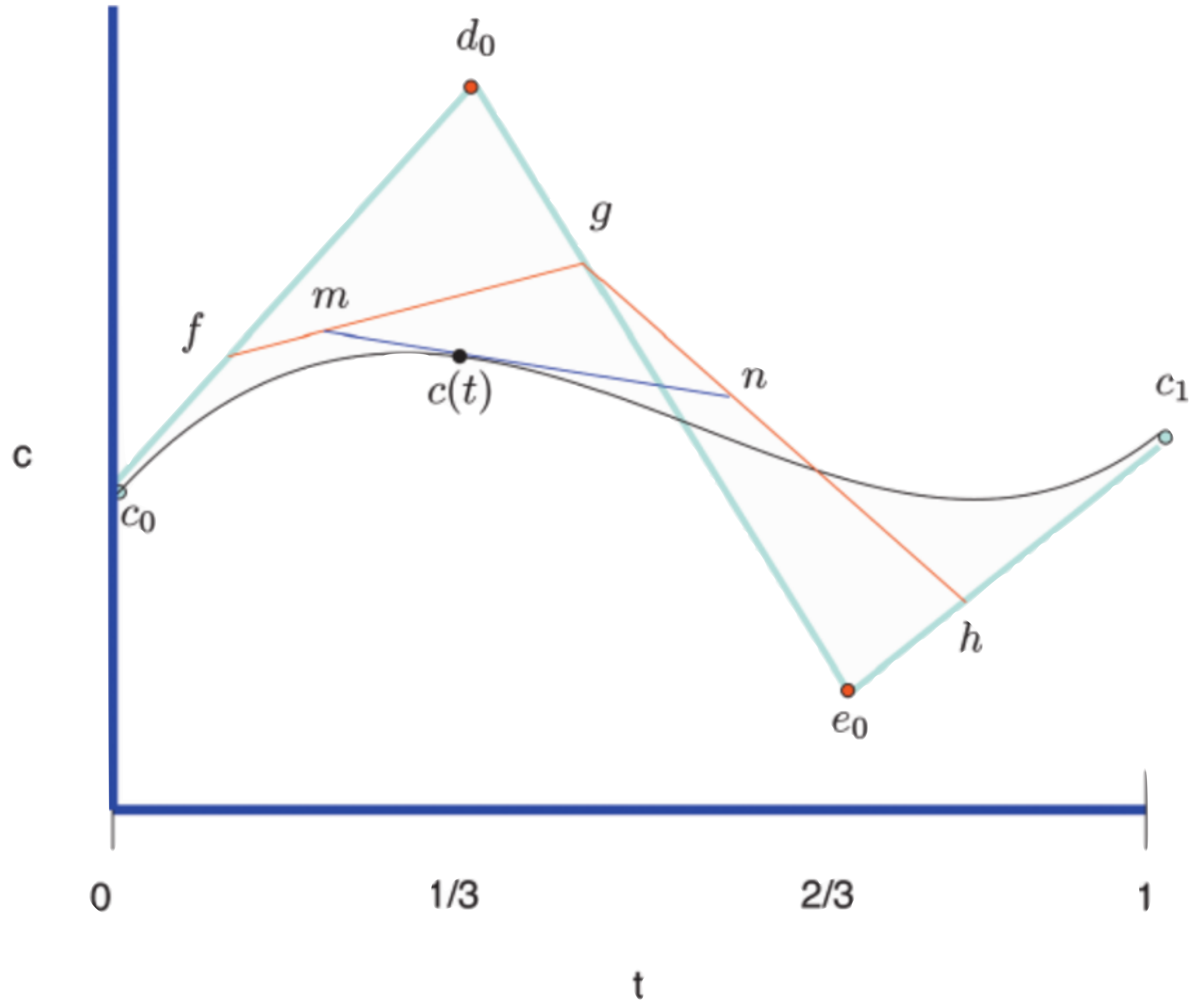
$$g = (1-t)d_0 + te_0$$

$$h = (1-t)e_0 + tc_1$$

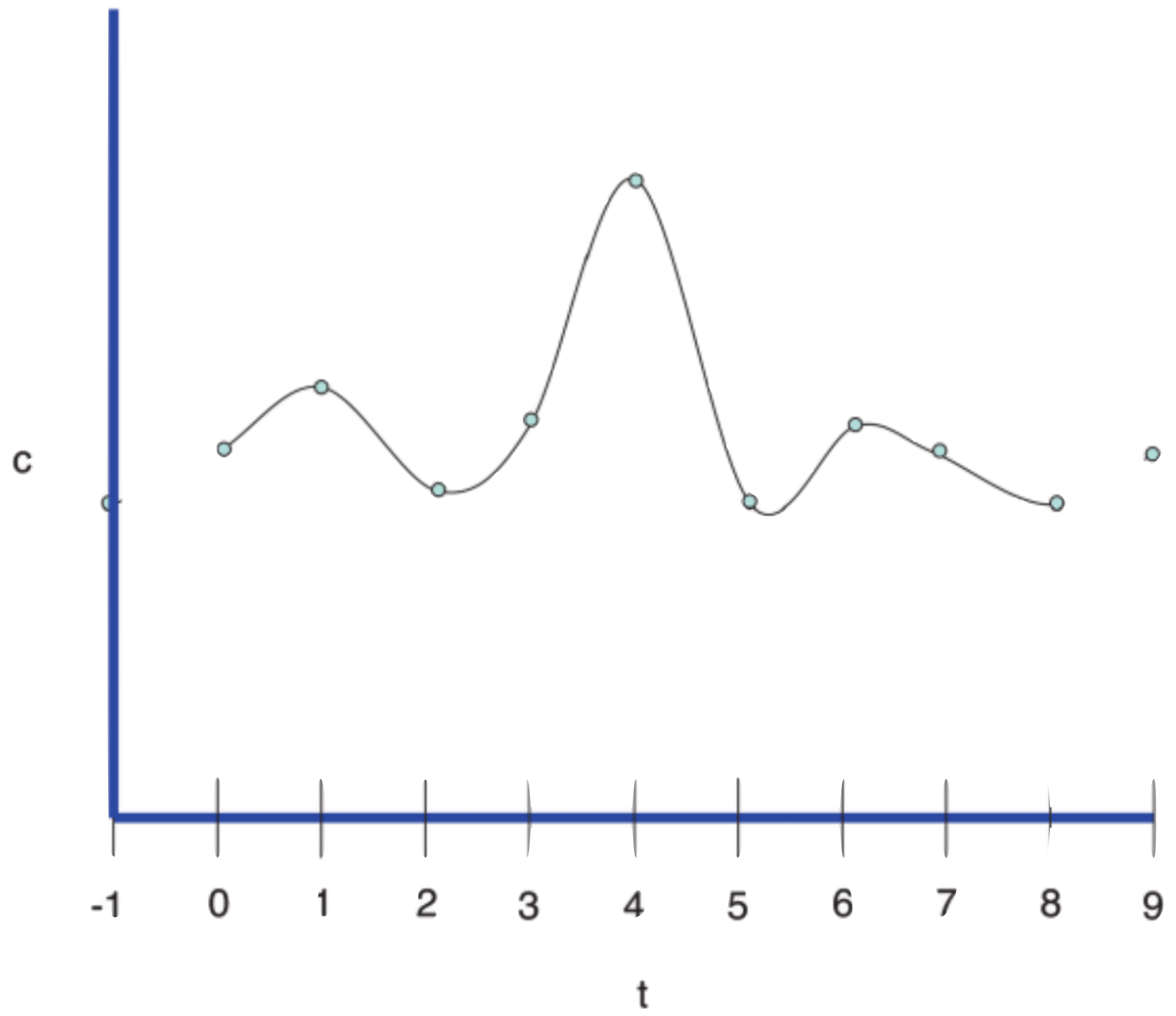
$$m = (1-t)f + tg$$

$$n = (1-t)g + th$$

$$c(t) = (1-t)m + tn.$$



$$c(t) = c_0(1-t)^3 + 3d_0t(1-t)^2 + 3e_0t^2(1-t) + c_1t^3.$$

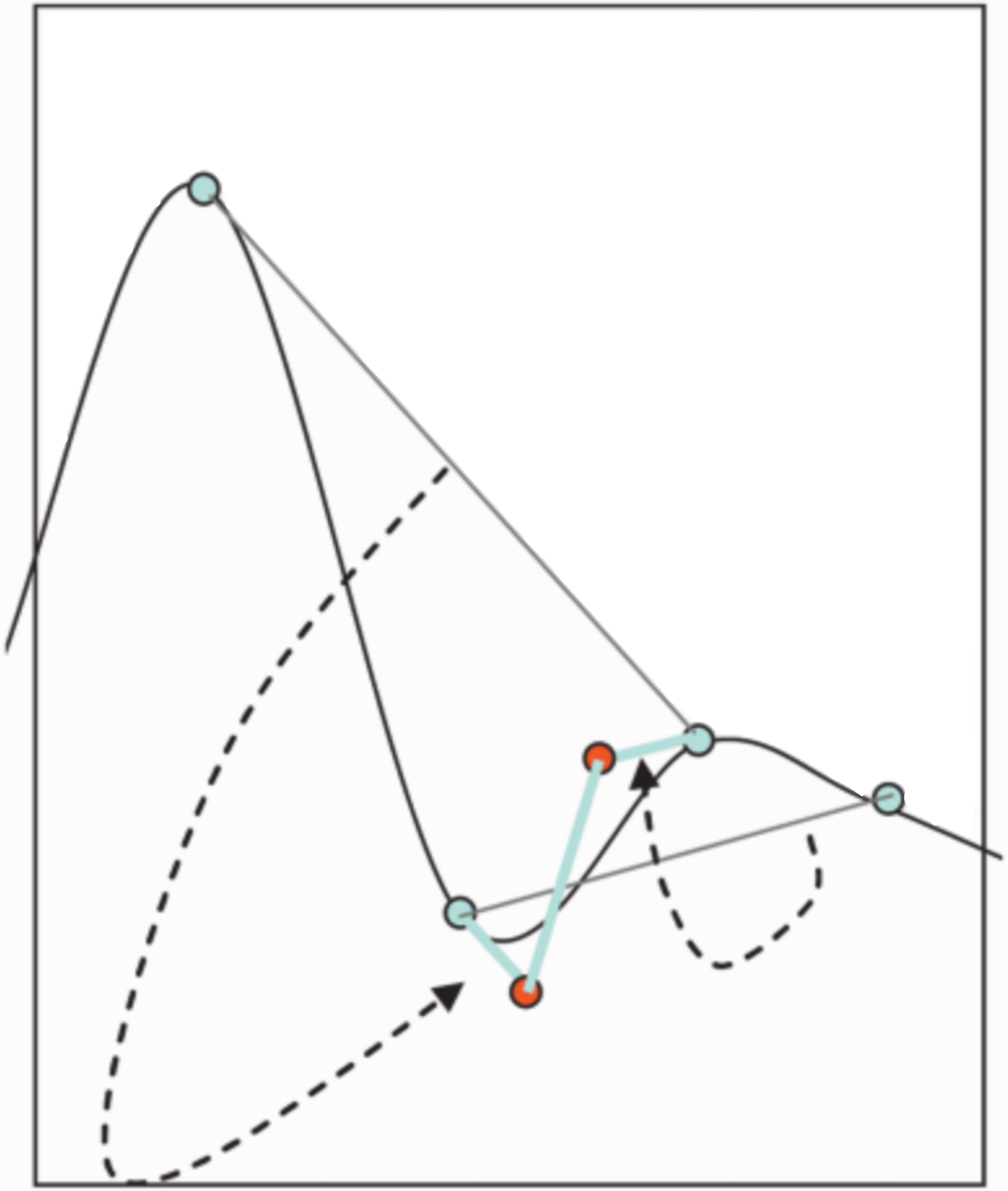

c
-1
0
1
2
3
4
5
6
7
8
9
t

$$
\begin{aligned}
f &= (1-t+i)c_i + (t-i)d_i \\
g &= (1-t+i)d_i + (t-i)e_i \\
h &= (1-t+i)e_i + (t-i)c_{i+1} \\
m &= (1-t+i)f + (t-i)g \\
n &= (1-t+i)g + (t-i)h \\
c(t) &= (1-t+i)m + (t-i)n.
\end{aligned}
$$

Catmull-Rom Splines

$$d_i = \frac{1}{6}(c_{i+1} - c_{i-1}) + c_i$$

$$e_i = \frac{-1}{6}(c_{i+2} - c_i) + c_{i+1}.$$



Interpolating rotations

Slerp (Spherical LERP)

$$r = (1-t)p + tq$$

$$r = \texttt{slerp(p,q,t)},$$

$$d_i = \left[(c_{i+1} c_{i-1}^{-1})^{\frac{1}{6}} \right] c_i$$

$$e_i = \left[(c_{i+2} c_i^{-1})^{\frac{-1}{6}} \right] c_{i+1}.$$