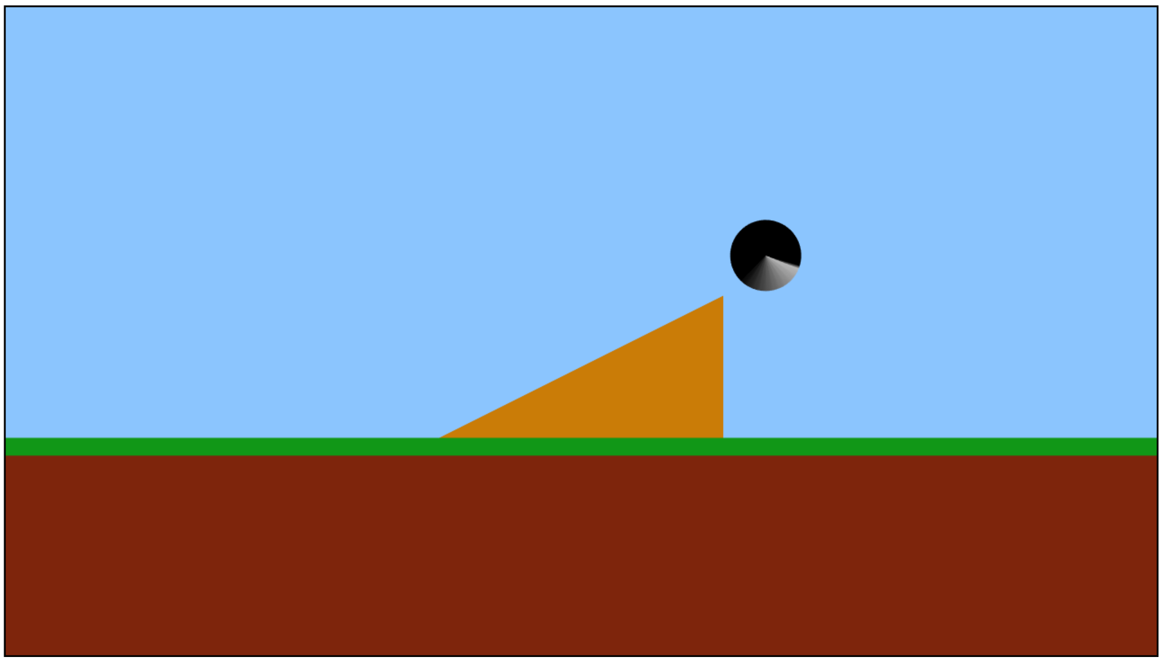
ETH Zürich
Computer Science Department

Visual Computing

Exercise 10- Animation



This exercise includes 5 different files:

- *index.html*: Used to start the scripts. Open this file in your browser.
- *webgl.css*: Style sheet for index.html.
- *gl-matrix.js*: Contains auxiliary functions.
- *mainScript.js*: Script in charge of running and rendering.
- *Exercise.js*: Script containing the information about the animation.

You can have a peek at all the files (and change them if you want), but you can fully solve the exercise just by editing *Exercise.js*.

10.1 Nearest Neighbor

Have a look at the function *nearestNeighbor*. Before applying the actual nearest neighbor scheme, the methods *getPrevFrameIdx*, *getNextFrameIdx*, and *getNormalizedTimeOffset* are called. Make sure you understand their implementation and what they compute, since you will need to use them in the next tasks. After, implement the nearest neighbor interpolation function and uncomment the following code to observe the sphere snapping to the nearest keyframe.

```

// Task 1
sphere_x = nearestNeighbor(keyframes_x);
sphere_y = nearestNeighbor(keyframes_y);
sphere_alpha = nearestNeighbor(keyframes_alpha);
```

10.2 Linear Interpolation

As you can see, the nearest neighbor interpolation scheme is not very convincing for animations, since it is not a continuous function. Implement the linear interpolation scheme, uncomment the following code, and observe how things improve.

```

// Task 2
sphere_x = linearInterpolation(keyframes_x);
sphere_y = linearInterpolation(keyframes_y);
sphere_alpha = linearInterpolation(keyframes_alpha);
```

10.3 Cubic Spline Interpolation

Although continuous, simple linear interpolation is not powerful enough to generate a physically plausible animation for our sphere.

a) Implement a the cubic Hermite spline interpolation scheme and, as usual, uncomment the following code to see your animation.

```

// Task 3
sphere_x = cubicSplineInterpolation(keyframes_x);
sphere_y = cubicSplineInterpolation(keyframes_y);
sphere_alpha = cubicSplineInterpolation(keyframes_alpha);
```

b) For your convenience, the keyframes' times and values are already set. However, only the tangents for the parameter *sphere_x* have meaningful values. Change the tangents values for *sphere_y* and *sphere_alpha* to create a better looking animation.