Exercise – Interpolation

The exercises for this session make use of the AIE Bootstrap codebase.

1. Create Linear Interpolation methods for your Vector classes.

Note: Interpolation works on many different types of variables, not just Points and Vectors. Consider making a template Linear Interpolation function.

1. Test out your Interpolation methods graphically using Bootstrap.

For example, you could draw two sprites at different locations on the screen and then draw a third sprite moving between the two points using Linear Interpolation.

1. Using the below Bootstrap code as a base, implement various Easing functions to change the way that the sprite scales.

The code is used within a project using the Renderer2D class to draw 2D shapes. It is recommended that you implement the easing functions within a containing class or namespace, and you may wish to implement them as template methods so that they can be used to interpolate various data types that allow interpolation.

The example sprite scales up from 0% to 100% over 1 second, then stays at a scale of 100% for 1 second, then scales down to 0% over 1 second before repeating the process. You should aim to effect the rate of the ramp up and down by using easing methods.

// within Application::draw()...

// create a basic scale that ramps from 0 to 1

// then stays at 1 for a short period of time

// then ramps down to 0 and repeats

float scale = fmod(getTime(), 3.0f);

if (scale > 1 &&

scale < 2)

scale = 1;

if (scale > 2)

scale = 1 - (scale - 2);

**// do something with scale here using easing!**

**// ...**

// render a box that scales

m\_2dRenderer->setRenderColour(1, 0, 1);

m\_2dRenderer->drawBox(getWindowWidth() \* 0.5f,

getWindowHeight() \* 0.5f,

200 \* scale,

200 \* scale);