Salamander

[CBPrice 19-11-20]

There are two modes of locomotion WALK and SWIM. The code template provided manages the transition between these two modes.

You will code the salamander procedurally, based on equations from the literature

Swim locomotion mode

Here a sine wave travels down the salamander's body. When this interacts with the water, the water produces a forward thrust on the salamander. Here's the maths required.

A simple sine wave along the body is given by the expression

$$y(x) = \sin\left(\frac{2\pi}{L}x\right)$$

Where *x* is the distance along the salamander's body, starting from the head, and *L* is its length. So at the tail we have *x = L* so we have one complete sine curve.

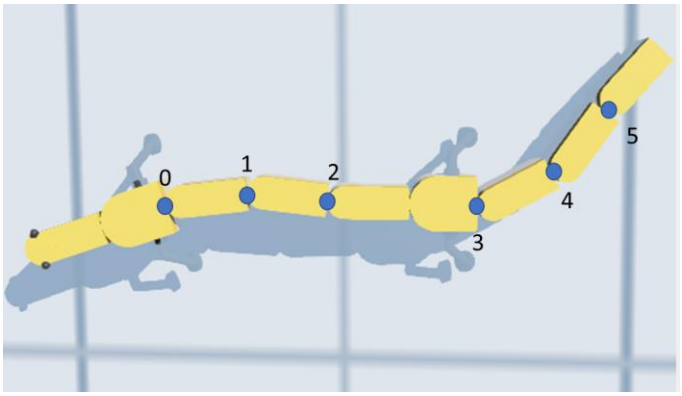
Now we make the sine wave travelling. We do this as follows

$$y(x,t) = A\sin\left(\frac{2\pi}{L}x - 2\pi ft\right)$$

Where *t* is time and *f* is the frequency of the salamander's beating, and *A* is the amplitude of the beating. Finally we arrange the amplitude to change along the length of the body, so we add another factor

$$y(x,t) = A\left(\frac{x/L + b}{1 + b}\right)\sin\left(\frac{2\pi}{L}x - 2\pi ft\right)$$

So, how to code this? Let's have a look at the salamander structure. As shown below there are 6 rotational motors which rotate the segments. The rotation angles are loaded into an array **target_angle[i]** which is used to render the salamander.



So let's see how to code the various bits of the last maths expression above. Here we are interested in the motors 0 to 5 which correspond to the labelled segments in the diagram above.

1) First $-2\pi ft$. This corresponds to the variable **phase** and we code this as

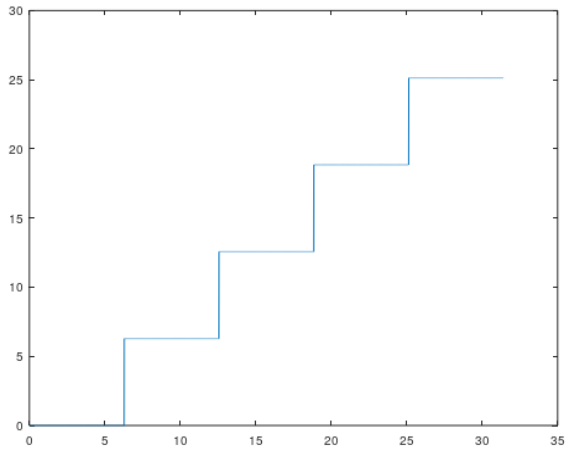
phase -= FREQUENCY * 2 * M_PI * (double)CONTROL_STEP/1000.0; The last term is the time interval between each pass of the main while(…) loop.

2) Now $\frac{2\pi}{L}x$ This is simply **(2 * M_PI / L) * i;** where **i** is the index into the target_angle[…] array and plays the role of the maths symbol *x.*

3) Finally the term $A\left(\frac{x/L+b}{1+b}\right)$. This is straightforward. Remember to use **i** in place of *x* and *A* is coded as the variable **SWIM_AMPL**.

4) Finally, finally, we need to get the salamander to avoid obstacles. This means bending its body, in other words adding a little offset to the target_angle[…] array. So add the variable **spine_offset** (which is already declared) to your expression for target_array[i]. We'll come back to this later.

Now we have to deal with the legs during the swimming mode. You can just see them pointing backwards in the above figure. We would like them to rotate quickly then point backwards on each cycle of motion. So the leg angles should advance with time something like this

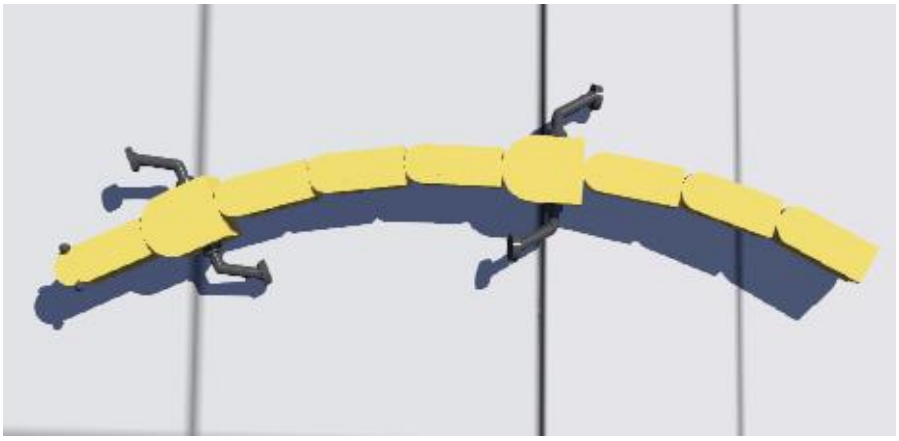


You see how they stay the same, then suddenly increase. The function which produces this behaviour is simply

phase – fmod(phase,2.0*M_PI) so this should be assigned to target_angle[i] for the legs (I = 6,7,8,9).

Walk locomotion mode

This is simpler than swim. The body just wiggles as shown below. But look at how the legs are positioned.



So for the segments we will have something like this, which is just a bend, not a travelling wave since there is no time (**phase**) in the expression.

target_angle[i] = WALK_AMPL * sin(phase) + spine_offset;

Then you will have to work out how to get the legs to walk. They will all rotate, but there will be some phase difference between them, as you can see in the figure above.. So you'll need to use **phase** and code something like

//rotate legs

target_angle[6] =

target_angle[7] =

target_angle[8] =

target_angle[9] =

Obstacle Avoidance

Here we need to compute **spine_offset**. The salamander has two eyes and these give you a signal **eyeL_val** and **eyeR_val**. Work out how to use these to get the salamander to bend in the correct way to avoid the walls. HINT: Do not use if(…)'s. It's really quite simple.