Algorithm for Samsung's 'shock wave' animation mini-project:

- Two height maps are used to store the current and previous states of the battlefield. Each map should be the same size as the image. In my implementation I've used a single array of shorts that is large enough to cater for both states, plus, I've added two rows per state. This was so that the ripples could reach the top and bottom without crossing over into the opposite state map.
- Each frame we toggle between state maps. I'm using a simple offset to swap between starting locations within the array.
- For each array element in the current state array:-
    - Look at the neighbouring pixels from the previous state, i.e. above, below, left and right. Take the sum and divide by 2.
    - Now subtract the value in the current state map.
    - If we left it like that the ripples would never subside so we need to diminish the strength of the ripple every pass. The most realistic way of doing this is to reduce the resulting height by a fraction (e.g. 1/32nd) of itself.
    - We now need to distort the background image based on the height of the ripple in the current location. We do this by calculating an offset. We calculate an X/Y offset based on the current distance from the centre of the ripplemap and the magnitude of the ripple at this point.
    - Perform a bounds check on the offset, i.e. check that the offset coordinates are not negative or larger than the size of the texture image.
    - Plot the pixel at current ripplemap x/y location using the texel at the calculated offset.

Here is a website describing the physics and Math behind this exact same algorithm:
http://www.cppblog.com/kesalin/archive/2010/09/01/android_ripple.html
(it even has an OpenGL-based Android implementation)