

Non Deterministic Paper Tear Algorithm

By William W Johnson, 25th May 2009

One of the biggest areas today in computational science is the modelling of nature. There are times where by observing nature, the events that unfold each day could be considered random. A great deal of effort and research is undertaken in all of today's modern sciences to discover the unknown, what, how and why are common questions which engage scientists daily.

It is our understanding that nature has n laws and nothing in nature is exempt from these laws. The initial problem is merely identifying the laws and thereby accepting them, our progression in the world of physics illustrates this time and time again. In the world of computing, nature is one aspect that may need to be modelled (simulations, mathematical modelling, games, etc), applying the current known laws of nature is not something that need be difficult which this algorithm will emphasize.

Take for instance the concept of a person tearing an extract from a newspaper. Whilst the event is triggered by a human, nature is fundamentally the element that tears the paper in relation to the angle and force applied by the human. We can logically conclude that from a physics perspective, the resultant tear of the paper is due to the angle, applied force and the microscopic densities that exist in the paper. These varying densities in the fabric cannot be observed with the naked eye but they exist and contribute to the resultant tear none the less.

In order for a piece of software to simulate this tear, there needs to be a mechanism that can produce viable results since both the human and nature are absent at least "practically". In computer science, this is where algorithms can be applied to fulfil the role of nature and the human. An algorithm is a sequence of statements that the computer computes to complete a task, other terminology would state an algorithm is a finite procedure of statements.

The first problem that needs to be solved is identifying where the tear would originate from. If the person were to tear a newspaper print in half, the start point can be found by **halving** the given fabrics **height** boundary and then picking a **random point** say 5% left **or** right of the halved height boundary.

After the first point (origin) is chosen, an **algorithm** can be applied which traverses up to the fabrics **width** boundary, As this **traversal** proceeds, the **left or right adjacent point** can be chosen to represent the varying degrees of **density** found in a fabric.

Essentially the algorithm randomly chooses between the left or right adjacent point as it progresses left to right of the width bound (linear north walk), this can be envisioned by having the choice of two routes where one route is blockaded, the route which is free of obstacle offers less resistance thus is the one who should be chosen, this principle is applied to choose between two points in the fabric, one of which is more dense than the other, the same reason why a paper tear leaves a jagged path.

This algorithm when applied and given the same input will much likely return a different output, the probability of this algorithm yielding the same output is improbable, it is therefore denoted a "non deterministic algorithm" as the output is not predictable. This correlates to the improbability that a person tearing a newspaper print would have two prints identically torn.

The following is a Matlab implementation of this algorithm which tears a bottom segment from a given photograph.

Input



The following is an output after the image is passed to the algorithm discussed in this article, the Matlab code is below. The only difference is the initial point chosen which is relative to the bottom of the fabric as opposed to somewhere in the middle.

Output



Code

```
function tImage = tornImage(mImage)

% obtain size attributes
mUnits = size(mImage);
iWidth = mUnits(1, 2);
iHeight = uint16(mUnits(1, 1));
```

% Tear @ Bottom Algorithm:-

*% Pick a random point within a small percentage of height bound then
% continuously walk up to width bound whilst hopping onto either the
% left or right adjacent point to simulate a realistic paper tear.*

```
iTearPoint = 0.01 * iHeight;           % start point

for i = 1 : iWidth                     % width bound

    if iTearPoint ~= iHeight           % test for upper boundary

        if ( rand * 2 ) > 1            % tear to the right
            iTearPoint = iTearPoint + 1;
        else                            % tear to the left

            if ( iTearPoint ~= 1 )     % improbable altho not impossible
                iTearPoint = iTearPoint - 1;
            end
        end
    else                                % move away from upper bound, go left
        iTearPoint = iTearPoint - 1;
    end

    % populate torn region
    mImage(iHeight-iTearPoint:iHeight, i, 1:3) = 255;
end

tImage = mImage;                       % return torn image
```