Turing Trains

Rule-based management of a dynamic display of images

James Harland

Basic Idea

Imagine a computer screen divided into rectangular tiles, each of which displays a different image. A "train" works its way across the screen tile-by-tile, according to some internal instructions of its own, changing some images and leaving others alone, and changing direction as it sees fit. The aim of this project is to explore the development of movies made in this manner, including experimenting with tiles consisting of colours, photographs, or other visual images of interest (perhaps a live webcam?). The internal logic of the "train" will be based on the well-known computing concept of a Turing machine. At any particular point in time, the machine will be examining a single tile (which may be thought of as the position of the train's engine). The machine will examine the tile, and according to a set of rules internal to the machine, either leave it as it is, or change it for a different image. It will then move in one of three directions (right, left or straight ahead) to another tile. The previous 10 or so tiles that have been visited will be highlighted, so that the movement of the train and its "carriages" will be simulated by the highlighting of recently visited tiles. Some potentially interesting trains may arise from busy beaver Turing machines, which result in some surprisingly complex behaviour.

Scope

This is intended as a broad environment in which several different ideas can co-exist and bounce off each other. Hence whilst here are plenty of different ways in which something like this could work, it is intended that there not be a "received" or "superior" way of doing things. Hence anyone who is interested is more than welcome to join in and try out ideas.

Getting Started

An example of the a transition between two tile configurations is given below. The first one shows the path of a Turing train which is replacing blank squares (here a pale yellow) with images of Gandalf and Aragorn alternately. As the "memory" or "carriage length" is 5, there are only the 5 most recent images displayed. Hence in the next configuration, the oldest image, the one of Gandalf in the top left hand corner, is no longer highlighted.

In order to work out what can and should be done here, the first tasks is to develop a very basic tool which can be used to experiment with tile patterns and the like. This tool will allow the user to specify a collection of images to be used on the tiles, including a blank image. In the example below, there are three images (which we can label as 0, 1 and 2), with 0 being the blank image (in this case the pale yellow tile), 1 being Gandalf (as it is the first non-blank image used) and 2 being Aragorn. As experimenting with different images will be critical, there should be a side bar or something similar displaying the current images in use (and which numbers they are, as changing the order in which they are used may be significant). Once this tool has a way of displaying updated tiles, and a means for the user to manually change tiles (such as clicking on them to change them), then some rudimentary experiments can begin. Please note that this tool is very likely to follow a common pattern of research software, which that it is used to work out what the detailed specification should be, and hence at some point gets thrown away in favour of a better designed and fully functional version.

The train changes the blank square under the "engine" to Aragorn's image and moves to the tile on the right. The updated position is shown in the second configuration.

Somewhere there should be a sidebar or similar with the images below. The idea is that the user can change the images below and hence change what is displayed on each tile.

0 1 2

Using this mapping, the tile manager can view the above sequence of changes as follows, where the *Visited* value specifies how many time steps have elapsed since the tile was visited.

First configuration:

Image 1 Visited: 5	Image 2 Visited: 2	Image 1 Visited: 1	Image 0 Visited: ??
Image 2 Visited: 4	Image 1 Visited: 3	Image 0 Visited: 0	Image 0 Visited: ??

Second configuration:

Image 1 Visited: 6	Image 2 Visited: 3	Image 1 Visited: 2	Image 0 Visited: ??
Image 2 Visited: 5	Image 1 Visited: 4	Image 2 Visited: 1	Image 0 Visited:0

This will also allow for machine control of the tiles, by having the Turing machine (or any other way of determining a sequence of updates to tiles) merely specifying the number of the new image to be displayed, together with which tile is to be replaced. This makes it simple to interpret a Turing machine transition such as "If the input is 1, replace it with 2 and move right" as an instruction to change Gandalf's image into Aragorn's.

Size controls

It is anything but clear what the best tile size is going to be, or what the best number of tiles per movie should be. Hence it would be useful to be able to control both of these aspects with a slider or some similar mechanism, as well as the overall size of the screen.

Capturing human moves

In order to experiment with sequences of image changes, a user will most likely find it helpful to click around on a series of tiles and then have the system play them back. Hence a record facility would be useful, in which the use would click a record button, indicate some sequence of image updates by clicking on tiles, and then have the system play them back, at some default rate of say one change per second. As with most things, it would be useful to have this as a slider of some kind, so that experiments can be done with rates of playback and the like. A single-step mode would be useful as well.

Automatically generated sequences

Once the user can generate their own sequences in this way, the next step is to provide a mechanism to allow these sequences to be generated by a program. There are two main aspects to this: one is providing an API or similar interface so that a some other application can generated sequences. The other is to provide a mechanism for specifying a given Turing machine, and seeing what affect it would have. Hence a simple Turing machine emulator will need to be written here, and interfaced with the tile manager, so that a user can specify a Turing machine (for simplicity, in terms of the standard transition quintuple of State, Input, Output, Direction and NewState). The main novelties here from a Turing machine perspective are that there are now three possible directions to move (left, right, ahead), and that moving off the edge of the screen will result in wrapping (so that moving right in any tile of the rightmost column will move the "engine" to the leftmost column of the same row, and similarly for the other boundaries). If a combination of state and input is encountered that is not covered by the transitions given, the machine terminates.

At a later point, it would be useful to develop an interface which allowed a less mathematical way of specifying transitions. It would also be interesting to develop less strict versions, such as a machine with some random elements. However, these are a much lower priority to begin with.

Just in case it is relevant, it would seem that eventually a client-server architecture would be useful here, with the server being the tile manager, which in principle could be serving many clients (ie Turing trains) at once.

More Adventurous Ideas

Other directions that could be pursued include

- **multiple Turing trains** Having more than one train at a time allows for some interesting effects in its own right, as well as some potentially more complex ones, such as what happens when trains intersect. One possibility is to make colliding trains swap some of their internal

logic, so that neither is exactly the same again after the collision.

- **randomized choices** Turing machines in the busy beaver world are deterministic, ie they behave the same way every time for the same input. There is a well-known variety of Turing machine in which a choice needs to be made between transitions that may occur.. In this application, it may be interesting to allow a random choice between the possibilities, or to allow a random choice of replacement images or directions (or both) after a fixed number of moves (say 25).
- **dynamic tiles** In the standard Turing machine environment, nothing changes unless the Turing machine changes it. Here one could imagin the tiles change the images displayed autonomously (or, if you like, according to some process of their own, including random selection). Even a simplistic repetitive display of images may provide a sufficient level of nondeterminism, in that the time takes for a given Turing train to arrive at a given trile is not always obvious in advance.
- **changing multiple tiles** Turing machines come bundled with the philosophy that only one cell is changed at a time. Whilst this gives an obvious focus for changes, it may be interesting to explore other means of changing tiles, including, for example, changing an entire row or column of tiles at once. This will take some more intricate specification of the change to be made, it may be a way to introduce some more spectacular visual effects
- **greater context for changes** Conway's Game of Life is based on the use of the properties of neighbours to determine the next tile to be displayed. Introducing similar behaviour here may be interesting, ie taking into account neighbouring tiles or other contextual properties (in a manner similar to cellular automata). Wolfram's lengthy book *A New Kind of Science* has a multitude of variations on this theme.
- **manual changes** A simple way to introduce some different behaviour is to allow the user to change tiles manually while the train/s are running.