These methods will fill the entire space with a specified value or object. The difference between the ‘fill’ and ‘fastFill’ methods is that the former use putObject and putValue internally, in other words, they are sensitive to any subclassing and alteration of these methods, whereas the latter use low level access routines
directly and are therefore much faster (and remain unaffected by subclassing).

- copyDiscrete2d: (Discrete2d *)a toDiscrete2d: (Discrete2d *)b;
  This method copies the data in one Discrete2d object to another Discrete2d object. It assumes that both objects already exist.

- (int)setDiscrete2d: (Discrete2d *)a toFile: (const char *)filename;
  This method reads a PGM formatted file and pipes the data into a Discrete2d object.

Accessing elements

The Discrete2d is essentially a 2d array of ids. However, we also allow users to access the array as if it were an array of integers: the integers are cast into ids for storage. The Object methods retrieve values as objects, the Value methods cast things to integers before returning. We believe this is safe on all architectures.

- getObjectAtX: (int)x Y: (int)y
  Return the pointer stored at (x,y).

- getValueAtX: (int)x Y: (int)y
  Return the integer stored at (x,y).

- putObject: anObject atX: (int)x Y: (int)y
  Put the given pointer to (x,y) overwriting whatever was there.

- putValue: (int)v atX: (int)x Y: (int)y
  Put the given integer to (x,y) overwriting whatever was there.

Low level access

For speed, sometimes you want to go below the method call layer and access cells quickly. To allow this Discrete2d defines some macros that give you internal access. Use these at your own peril! In particular, subclasses have no way to redefine the macros.

- (id *)getLattice
  Returns the lattice pointer - use this for fast access.

- discrete2dSiteAt(lattice, offsets, x, y)
  Macro to return a pointer to the value at site x, y. lattice is the return value from getLattice, offsets is the precomputed offsets (stored in self->offsets).

Miscellaneous

These methods are used in - createEnd to actually allocate the array. Subclasses might need to use these.

New Methods

- `updateLattice`
  Copy newLattice to lattice, in effect updating the lattice.
- `(id *)getNewLattice`
  Like `- (id *)getLattice`: return a pointer to the newLattice buffer.

Overridden Methods

- `createEnd`
  Rewrites the method from Discrete2d. Allocate two lattices, makes the offsets.
- `putObject: anObject atX: (int)x Y: (int)y` and `- putValue: (int)value atX: (int)x Y: (int)y` and
  Overridden so writes happen to newLattice.

Ca2d

Inherits from DblBuffer2d, defines abstract protocol for cellular automata.

New Methods

- `setNumStates: (int)d`
  Record the number of states the CA understands.
- `initializeLattice`
  Use this to set up your CA to a default initial state. Unimplemented in Ca2d.
- `stepRule`
  One iteration of the CA rule. Unimplemented in Ca2d.

Overridden Methods

- `createEnd`
  Check that numStates has been set.

ConwayLife2d

Classic 2d Conway's Life CA.

Creation Phase

- setDisplayWidget: (Raster *)r
  Set the display widget to use for drawing.
- setDiscrete2dToDisplay: (Discrete2d *)c
  Set the 2d array to draw
- setDisplayMessage: (SEL)s
  Set the message to be sent to each object in the grid to make it draw itself.
- setObjectCollection: objects
  (optional) set a collection of objects to be displayed. If this is not given, then Object2dDisplay loops through the 2d grid sending draw messages to all objects it finds there. Giving an explicit collection of objects to draw is more efficient if your grid is sparsely populated.

Use

- display
  Draw all objects in the array (or optionally, the collection) on the raster widget. All that happens here is the display message is sent to each object - it is the object's responsibility to render itself.
- makeProbeAtX: (int)x Y: (int)y
  Find an object at the given (x,y) coordinate and build a probe display for it. This is a good method to make a button client for a raster widget like

```
[aRaster setButtonClient: aObject2dDisplay Message: M(makeProbeAtX:Y:)]
```

Value2dDisplay

Value2dDisplay helps display 2d arrays of values. Value2dDisplay goes through a given Discrete2d array, turn states into colours, and draws them into a Raster widget.

Creation Phase

- setDisplayWidget: (Raster *)r Colormap: (XColormap *)c
  Set the display widget and the colourmap to use to draw the value array.
- setDiscrete2dToDisplay: (Discrete2d *)c
  Set which array to draw.
- setDisplayMappingM: (int)m C: (int)c
  Linear transform of states to colours for drawing.

This message is optional. It is used when the target Discrete2d contains objects. If a particular location in the space has no resident object, the argument of this message is the value which gets writtent to the file. The default background value is 0.

- fileTo: (const char *)fileName
  When the Int2dFiler receives this message, it opens a file called *fileName*, stores the state of a pre-specified space into it, and then closes the file.

Marcus G. Daniels <mgd@santafe.edu>
(mailto:mgd@santafe.edu)

Last modified: 1997-12-17