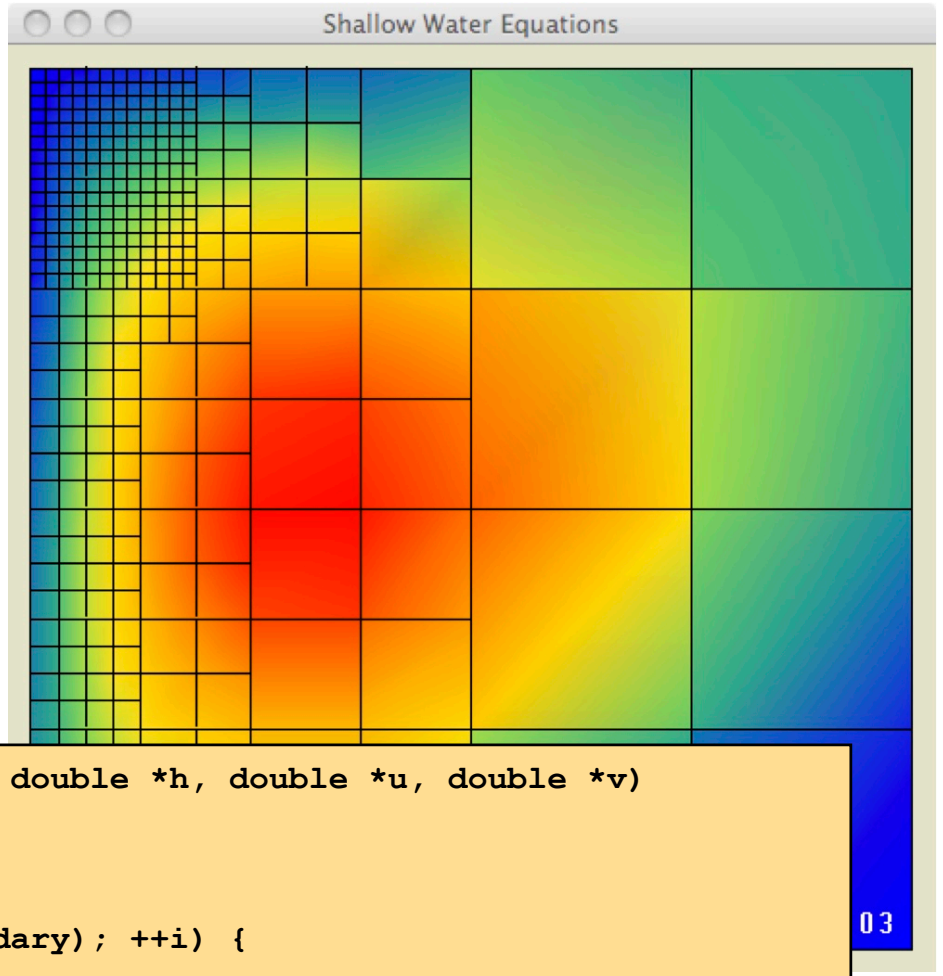
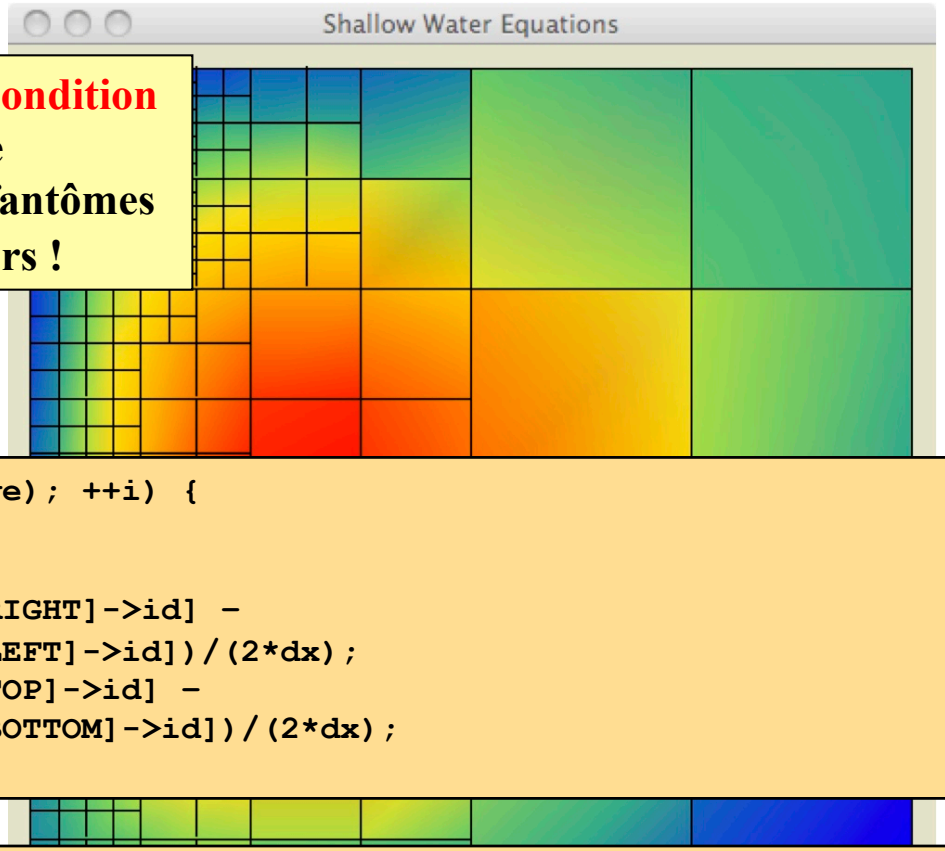
A few words about boundary conditions




```
void sweBoundaryConditions(Swe *swe, double *h, double *u, double *v)
{
  int i;
  Grid *g = swe->grid;
  for (i = 0; i < vectorSize(g->boundary); ++i) {
    Boundary *b = &g->boundary[i];
    double *un = (b->side == LEFT || b->side == RIGHT) ? u : v;
    double *ut = (b->side == LEFT || b->side == RIGHT) ? v : u;
    h[b->cell[1]->id] = h[b->cell[0]->id];
    un[b->cell[1]->id] = -un[b->cell[0]->id];
    ut[b->cell[1]->id] = ut[b->cell[0]->id];
  }
}
```

Afin de n'imposer qu'une unique condition au frontière, nous n'utilisons pas le gradient prescris dans les cellules fantômes pour calculer les gradients intérieurs !




```
for (i = 0; i < vectorSize(grid->active); ++i) {
  Cell *c = grid->active[i];
  double dx = gridDx(grid, c);
  gradx[c->id] = (field[c->neighbour[RIGHT]->id] -
                  field[c->neighbour[LEFT]->id])/(2*dx);
  grady[c->id] = (field[c->neighbour[TOP]->id] -
                  field[c->neighbour[BOTTOM]->id])/(2*dx);
}
```

```
for (i = 0; i < vectorSize(grid->boundary); ++i) {
  Cell *c = grid->boundary[i].cell[0];
  double dx = gridDx(grid, c);
  switch(grid->boundary[i].side) {
    case LEFT :
      gradx[c->id] = 2*(field[c->neighbour[RIGHT]->id] - field[c->id]) / dx
                        - gradx[c->neighbour[RIGHT]->id];
      break;
  }
}
```