Region detection in FlowSimulation project

Dávid Éles

February 3, 2015

1 Problem description

Given a 2 dimensional plane which contains regions. The goal to identify these regions. The regions are important to identify with a given discretisation. Actually our goal to determine which primitive cell belongs to which region.

2 Algorithm

We know that we have a given discretisation, lets denote these with h_1, h_2. Further more we know the range of our plane:

$$w, h$$

We resolve our plane to primitive cells and we say from a point $x \in \mathbb{R}^2$ that it is belongs to a cell, if

$$\exists k, l \in \mathbb{Z} : kh_1 \leq x_1 < (k+1)h_1$$

$$lh_2 \leq x_2 < (l+1)h_2$$

We assume that the border of a region is a closed continuous curve. If a point of a curve is inside a cell we know that there is a point in the cell's boundary. In this way there is an error and we could miss one cell at the boundary of the plane but we accept this.

Due to the above mentioned statements the first part of the algorithm goes through all of the cells boundary and checks whether there is any point of a curve or not. If there is we store that the given cell is a boundary-cell.

The second part actually the feeding of the regions.

```
Result: Identified all region
while x = FindFreeCell() do
    InitDeQueue(queue);
    regionId = nextRegionId();
    PushBack(queue, x);
    while not IsEmpty(queue) do
        x = popFront(queue);
        RegisterrCellToRegion(x, regionId);
        forall the p in NeighBorns(x) do
            if IsFreeCell(p) then
                PushBack(queue, p)
            end
        end
    end
end
```

Algorithm 1: Basic algorithm

In the algorithm we call a cell to free, if it is not a boundary point and not registered yet to any region.

3 Summarization

This second algorithm actually a breath first graph visitation, but it is implemented with a dequeue. The reason is simple, probably we will have lots of cells and the regions could be huge, therefore a recursive algorithm not so efficient.