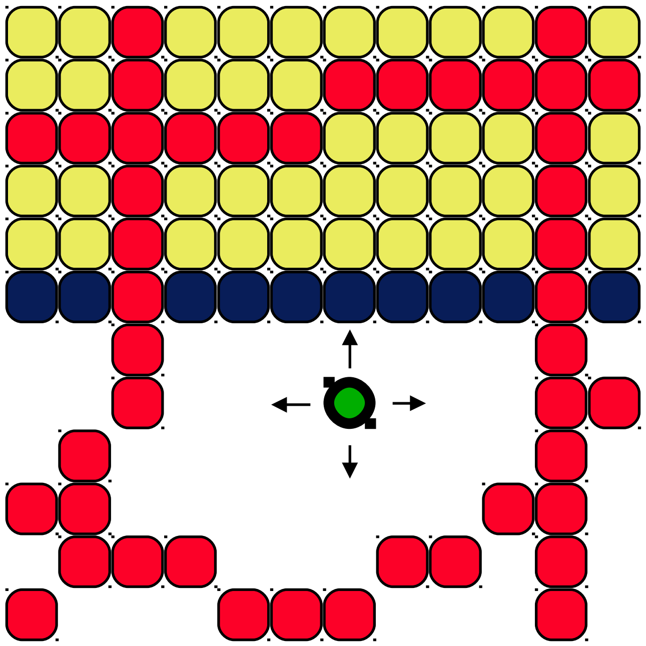
Computer
Graphics
Charles
University

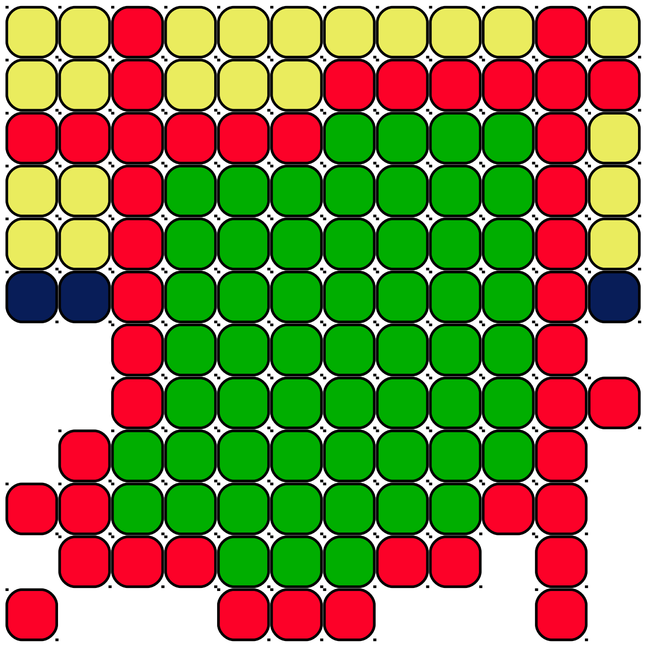
Filling Continuous Areas

© 1995-2015 Josef Pelikán & Alexander Wilkie
CGG MFF UK Praha

pepca@cgg.mff.cuni.cz
http://cgg.mff.cuni.cz/~pepca/

Border Filling

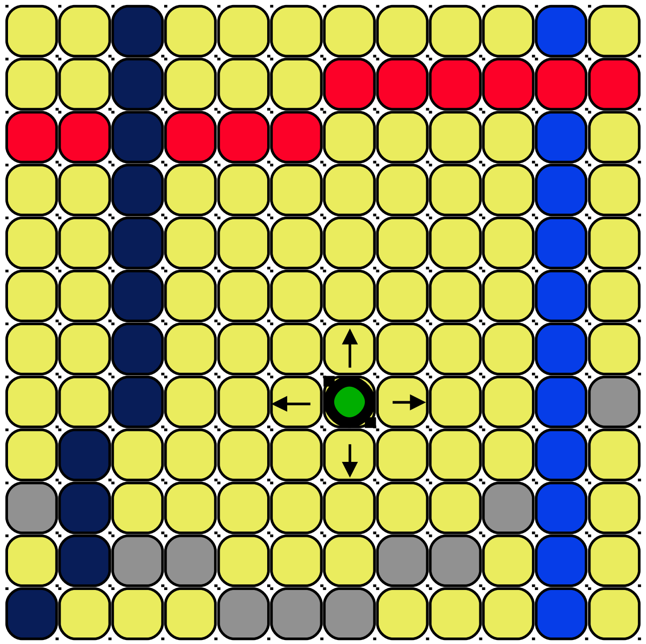


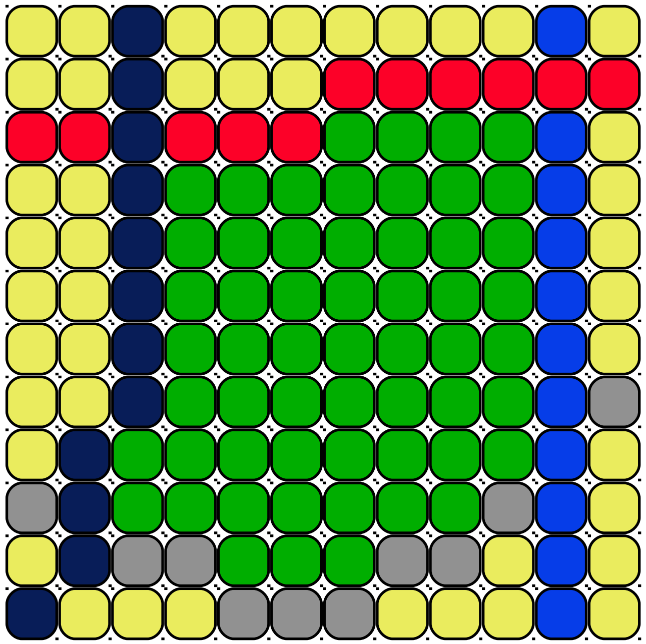


Filling until a border of given colour is reached

GetPixel(x,y) <> border_colour

Flood Filling

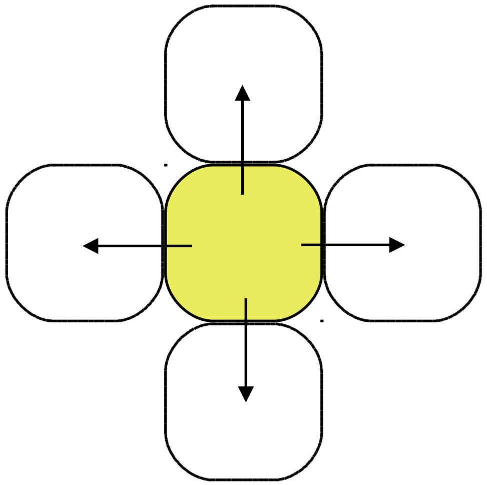


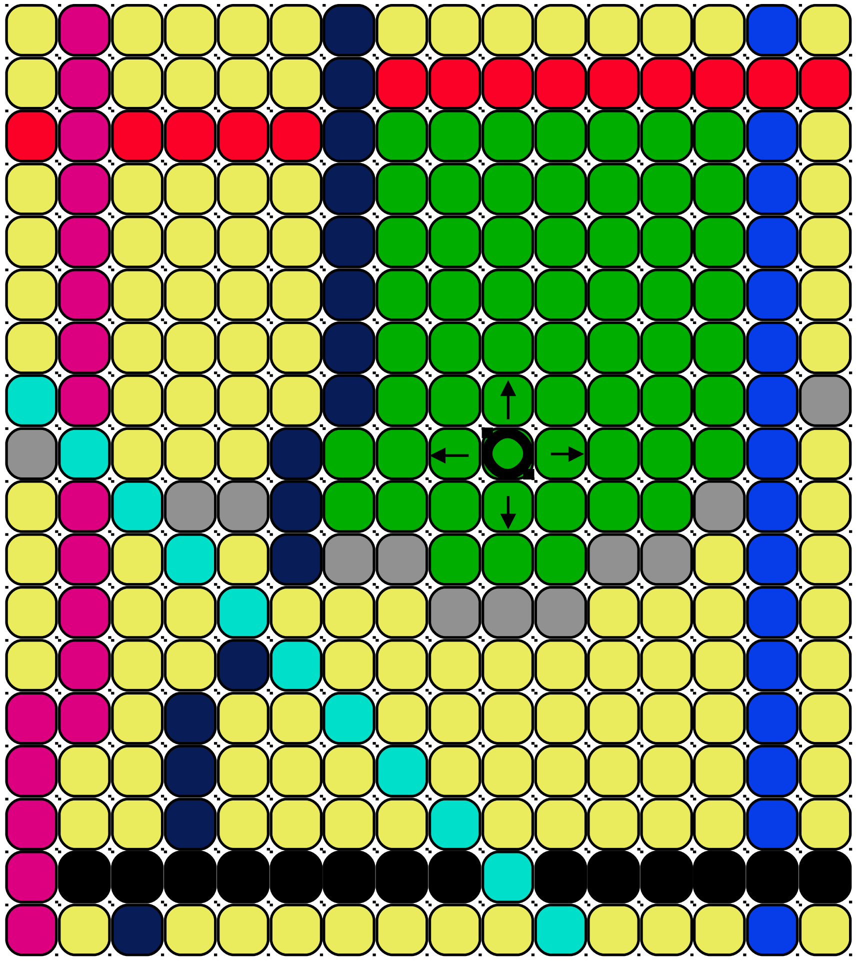


Re-colouring pixels of a given colour

GetPixel(x,y) = given_colour

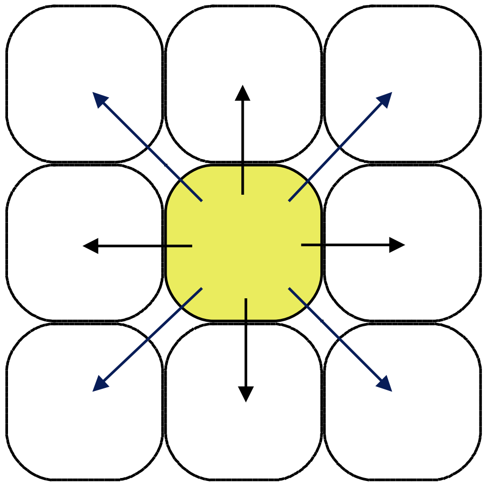
4-Neighbourhood



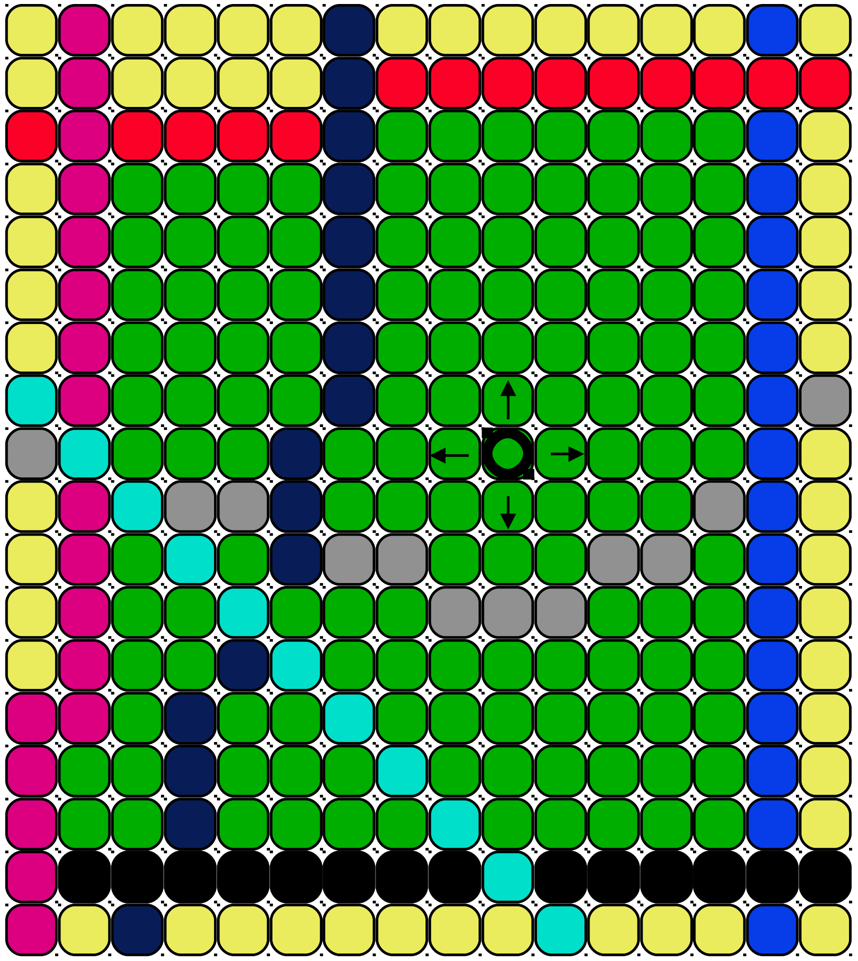


Flood Fill variant

8-Neighbourhood



Flood Fill variant



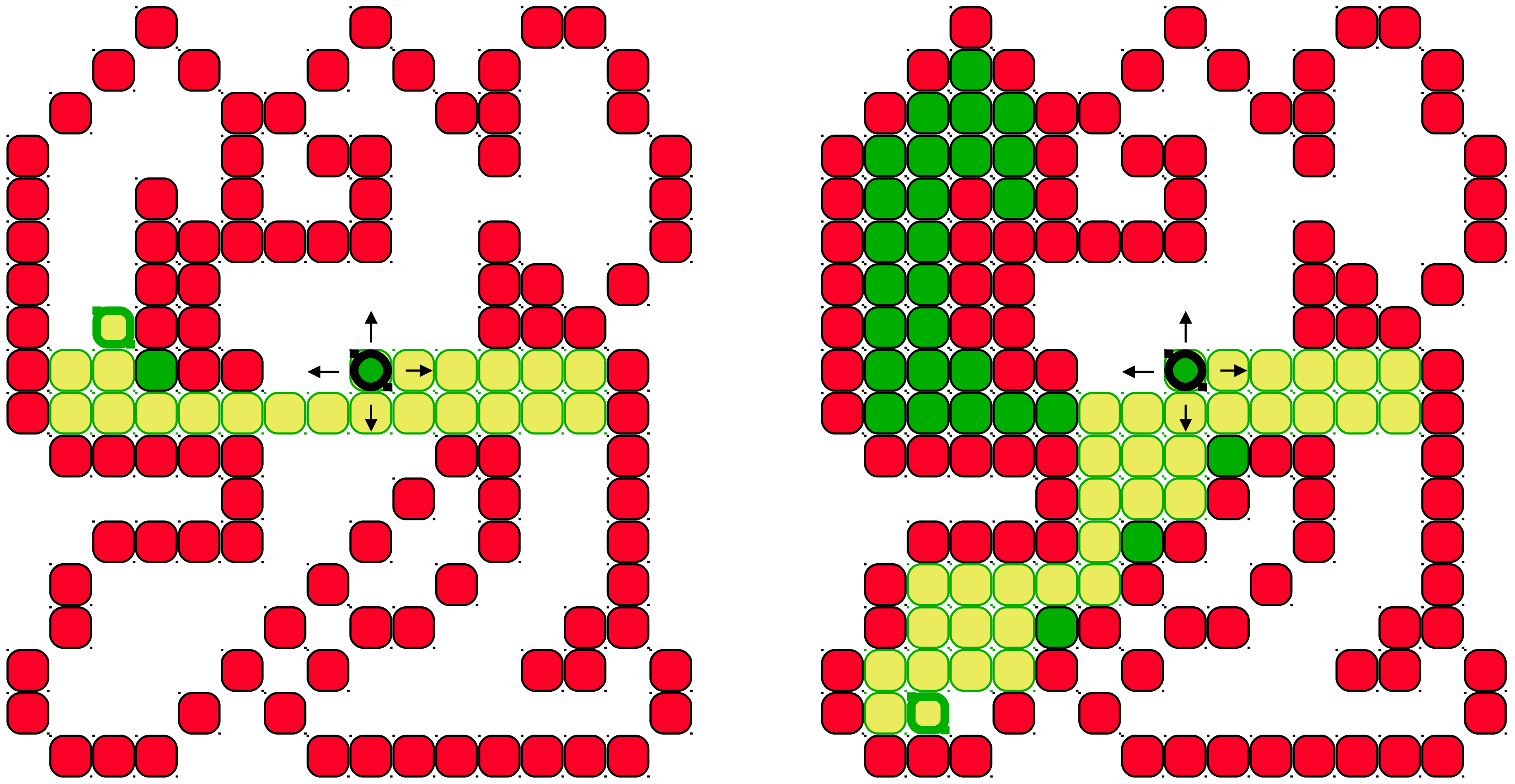
Naive Recursive Algorithm

```
procedure FloodFill4 ( x, y, oldc, newc : integer );
      { continuous 4 neighbour flood fill, oldc <> newc }
begin
  if GetPixel(x,y) = oldc then
    begin         { pixel [x,y] belongs to fill area }
      PutPixel(x,y,newc);
      FloodFill4(x+1,y,oldc,newc);  { four neighbours: }
      FloodFill4(x-1,y,oldc,newc);
      FloodFill4(x,y+1,oldc,newc);
      FloodFill4(x,y-1,oldc,newc);
    end;
end;
```

Border fill version:

```
(GetPixel(x,y) <> boundc) and
(GetPixel(x,y) <> newc)
```

Progression of the Fill Operation:



🟥 **border** 🟩 **filled** 🟨 **stack**

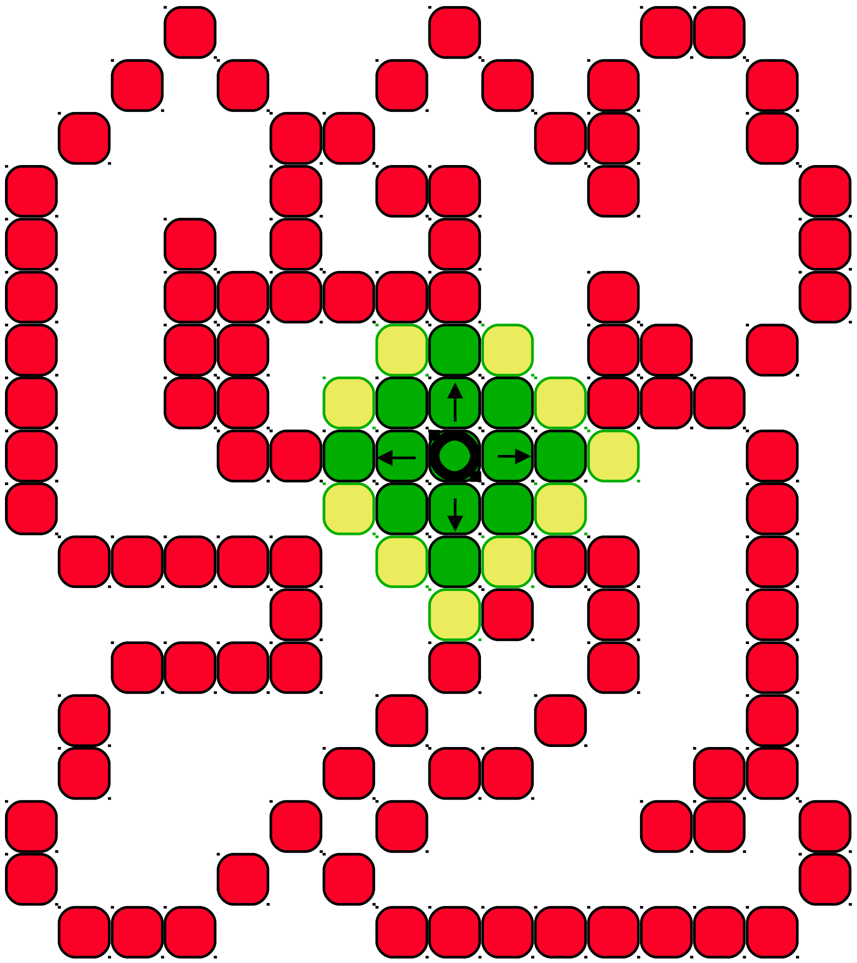
Queue Instead of Stack

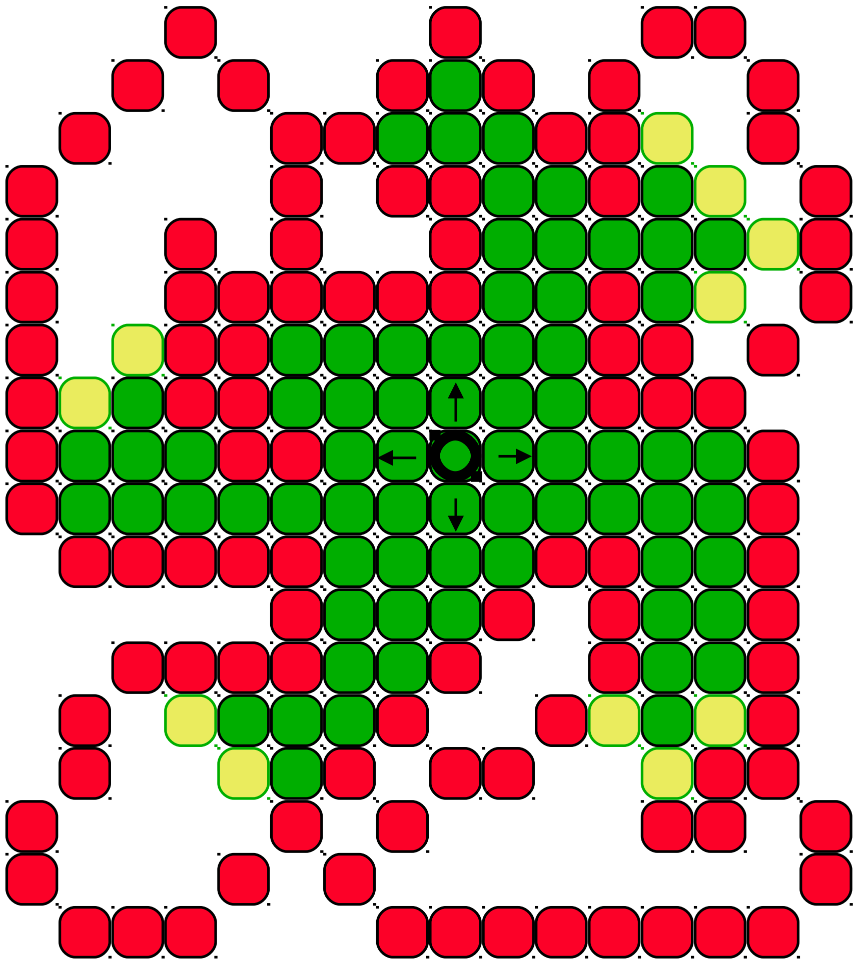
```
procedure FloodFill4 ( x, y, oldc, newc : integer );
      { continuous 4 neighbour flood fill, oldc <> newc }
var Q : Queue;
begin
  Q.Init; Q.Put(x,y);
  repeat
    Q.Get(x,y);
    if GetPixel(x,y) = oldc then
      begin      { pixel [x,y] belongs to fill area }
        PutPixel(x,y,newc);
        Q.Put(x+1,y); Q.Put(x-1,y);
        Q.Put(x,y+1); Q.Put(x,y-1);
      end;
  until Q.Empty;
end;
```

More Efficient Version

```
procedure FloodFill4 ( x, y, oldc, newc : integer );
      { continuous 4 neighbour flood fill, oldc <> newc }
var Q : Queue;
  procedure NextPixel ( x, y : integer );
  begin     { if pixel is in the area, put in queue }
    if GetPixel(x,y) = oldc then
      begin
        PutPixel(x,y,newc); Q.Put(x,y);
      end;
  end;
begin
  Q.Init; NextPixel(x,y);                { start pixel }
  repeat
    Q.Get(x,y);
    NextPixel(x+1,y); NextPixel(x-1,y);    { 4 neighbours: }
    NextPixel(x,y+1); NextPixel(x,y-1);
  until Q.Empty;
end;
```

Progression of the Fill Operation:



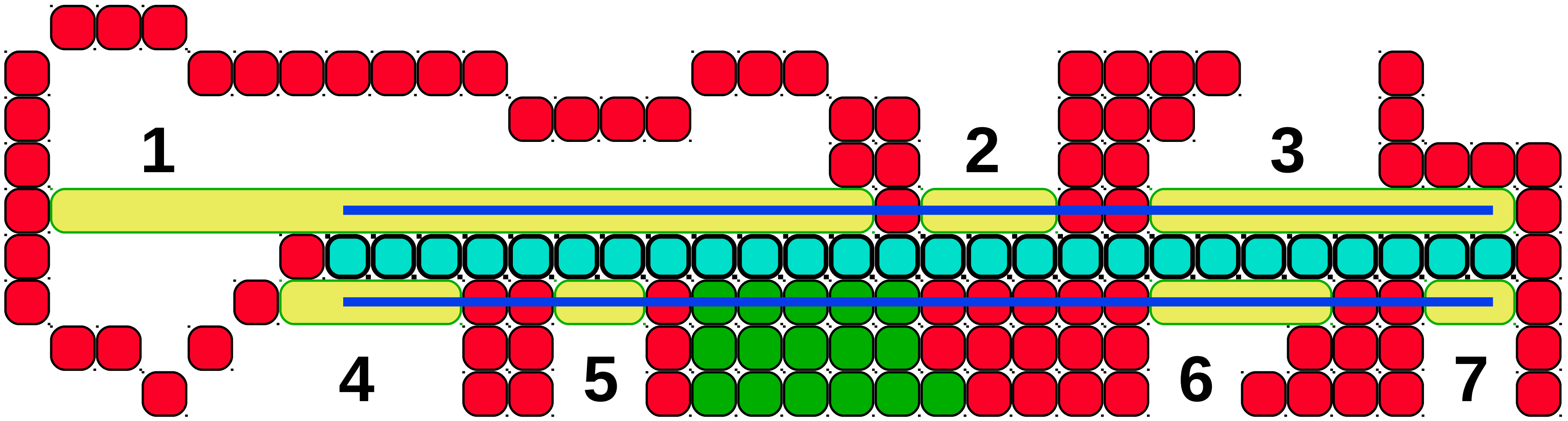


Line Fill

```
procedure LineFloodFill4 ( x, y, oldc, newc : integer );
      { continuous 4 neighbour flood fill, oldc <> newc }
var S : Stack;                { entries: [Xmin,Xmax,y] }
    Xmin, Xmax : integer;           { for the current scaline }

  procedure Search ( Xmin, Xmax, y : integer );
  var Xm : integer;
  begin     { goes over all line segments }
    while GetPixel(Xmin-1,y) = oldc do Dec(Xmin);
    repeat          { testing [Xmin,y] }
      Xm := Xmin;         { searching for the segment end: }
      while GetPixel(Xm+1,y) = oldc do Inc(Xm);
      S.Push(Xmin,Xm,y);
      Xmin := Xm+2;       { searching for the next segment: }
      while (Xmin <= Xmax) and (GetPixel(Xmin,y) <> oldc) do
        Inc(Xmin);
    until Xmin > Xmax;
  end;
...
```

Search for New Segments



🟥 **border** 🟩 **pixels filled earlier**

🟦 **recently filled pixels**

― **search lines**

1-7 ▭ **new segments on the stack**

Line Fill

```
...
begin
  S.Init; Search(x,x,y);       { first point (seed) }
  repeat
    S.Pop(Xmin,Xmax,y);
    if GetPixel(Xmin,y) = oldc then
      begin                    { segment not filled yet }
        Line(Xmin,y,Xmax,y,newc);
        Search(Xmin,Xmax,y-1);
        Search(Xmin,Xmax,y+1);
      end;
  until S.Empty;
end;
```

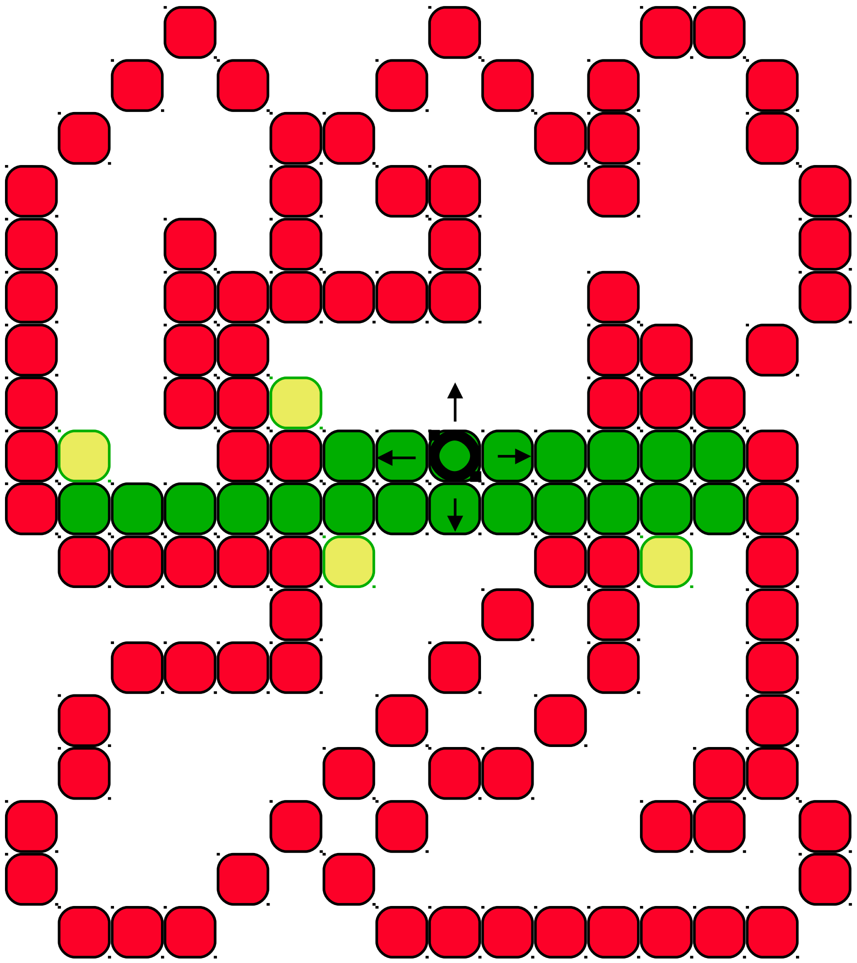
Border version:

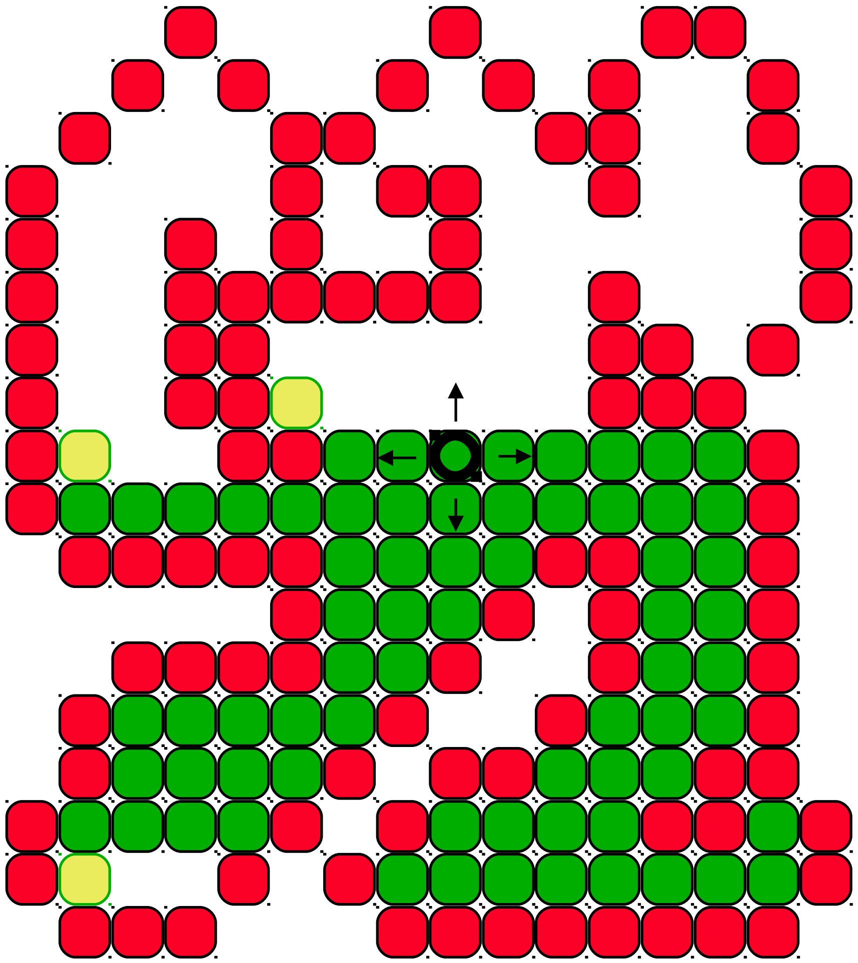
```
(GetPixel(Xmin,y) <> boundc)
      and (GetPixel(Xmin,y) <> newc)
```

8-neighbour:

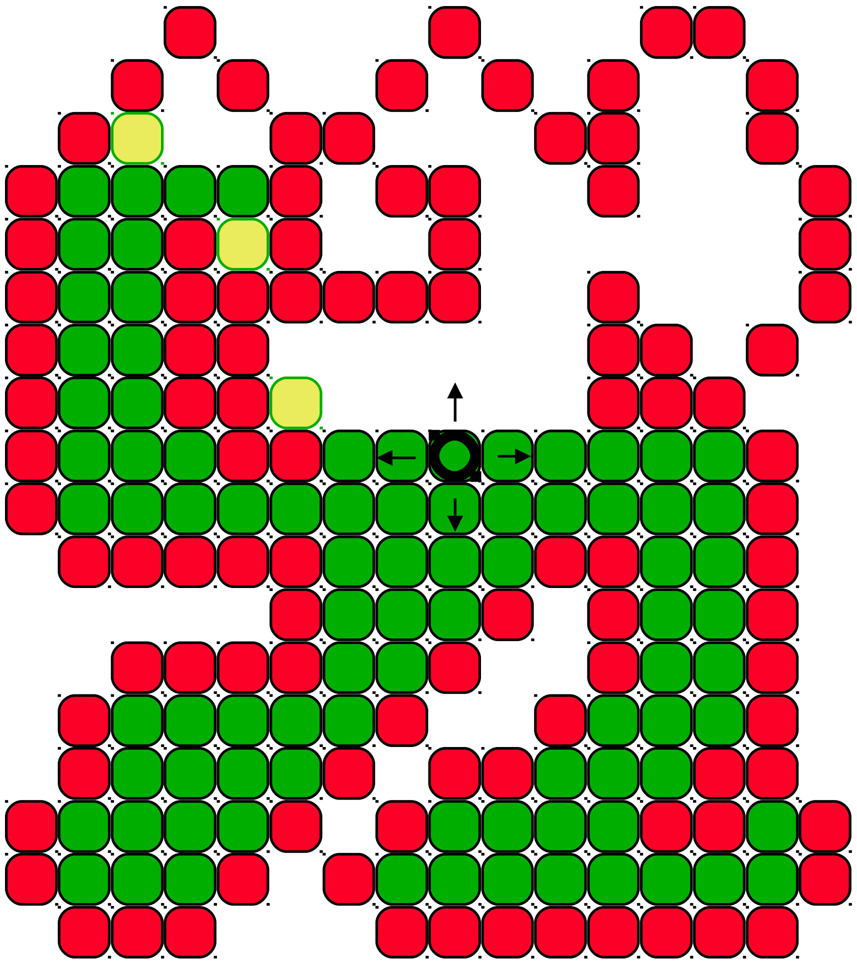
```
Search(Xmin-1,Xmax+1,*)
```

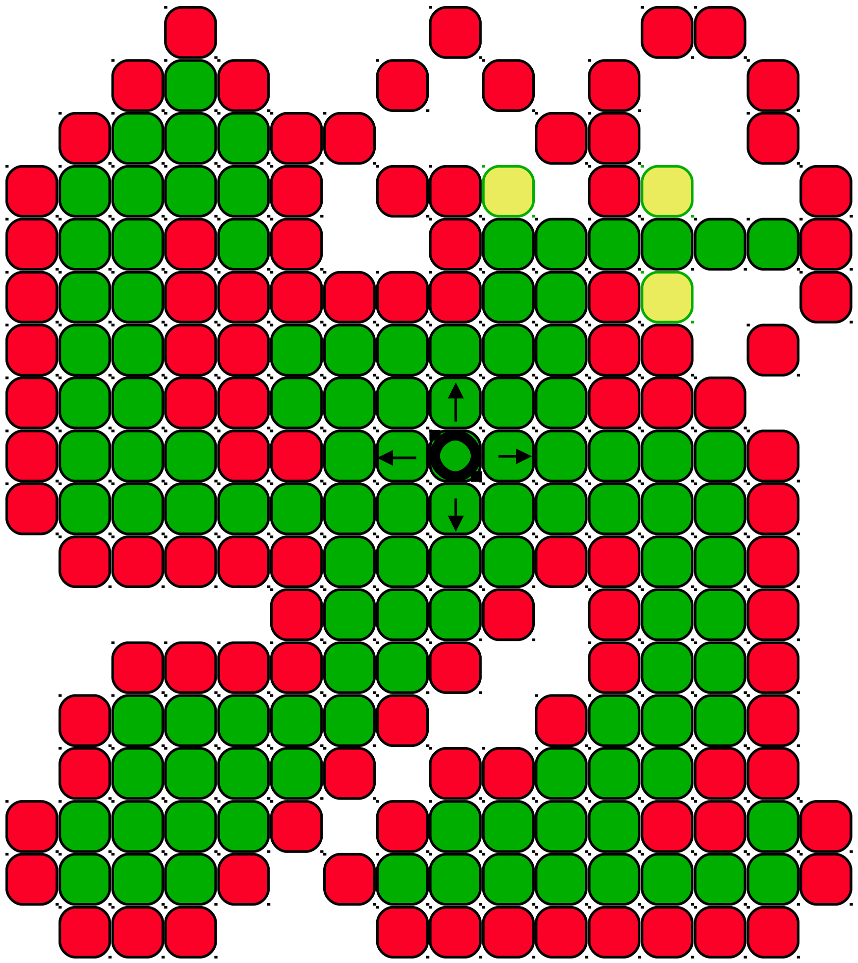
Progression of the Fill Operation:





Progression of the Fill Operation:





Advantages of the Line Algorithm

- ✚ **Less memory consumption**
  - – Usually, the stack only grows slowly
- ✚ **Higher speed**
  - – Better memory access for entire scanlines
- ◆ **Stack** versus **queue**:
  - – More local memory access for stacks
  - – More efficient for paging of video RAM

End

Further Information

- **J. Foley, A. van Dam, S. Feiner, J. Hughes**: ***Computer Graphics, Principles and Practice***, 979-982
- **Jiří Žára a kol.**: ***Počítačová grafika***, principy a algoritmy, 142-147