Subject: Re: TRIANGULATE. Finding contiguous cells efficiently?
Posted by Wox on Sat, 24 Feb 2007 13:34:31 GMT
View Forum Message <> Reply to Message

The first part is with loop, the second is vectorized. Maybe the three rebin/reform can be done in one step? Anyway, what do you think?

```
pro test
x=[1.5,0.,0.5,1,1]
y=[1.,0,1,0,2]
TRIANGULATE, X, Y, triangles
s=size(triangles,/dimensions)
ctriangles=fltarr(s[0],s[1],3)

; 1. With loop
for i=0,s[1]-1 do begin
 t=total((triangles eq triangles[0,i])+$
 (triangles eq triangles[1,i])+$
 (triangles eq triangles[2,i]),1)

 ind=where(t eq 2,ct)
 if ct ne 0 then ctriangles[*,i,0:ct-1]=triangles[*,ind]
endfor
; Third dimension gives the three contiguous neighbours
print, ctriangles

; 2. Vectorized
tr1=rebin(triangles,s[0],s[1],s[1])
block=tr1 eq
 rebin(reform(triangles[0,*],1,1,s[1]),s[0],s[1],s[1],/sample )
block+=tr1 eq
 rebin(reform(triangles[1,*],1,1,s[1]),s[0],s[1],s[1],/sample )
block+=tr1 eq
 rebin(reform(triangles[2,*],1,1,s[1]),s[0],s[1],s[1],/sample )
block=total(block,1)
ind=where(block eq 2,ct)
x = ind mod s[1]
y = ind/s[1]
block=[transpose(y),triangles[*,x]]
; First column: triangle index
; Other three columns: vertices for a contiguous neighbour
print,block
end
```

On 23 Feb 2007 18:39:39 -0800, "Libertan" <tbethell@umich.edu> wrote:

> My question: For a given row in TR, let us say row n, how best to find

> all the other rows in TR which share two elements with row n? (The
> algorithm must of course reject row n in its resulting list). I have
> a rather inefficient solution, by far the slowest part of my code, and
> I'm desperate to increase its speed (hopefully ten fold). My routine
> uses one FOR loop (over cells n) within which are many WHERE commands
> (which search for neighbour candidates).
