Subject: Re: New parallelization pattern with CoWork
Posted by Oblivion on Tue, 26 Dec 2017 23:36:36 GMT
View Forum Message <> Reply to Message

Hello Mirek,

This is good news. Thanks for your efforts!
U++ MT is really improved over time.

My initial impression: It looks and works good.

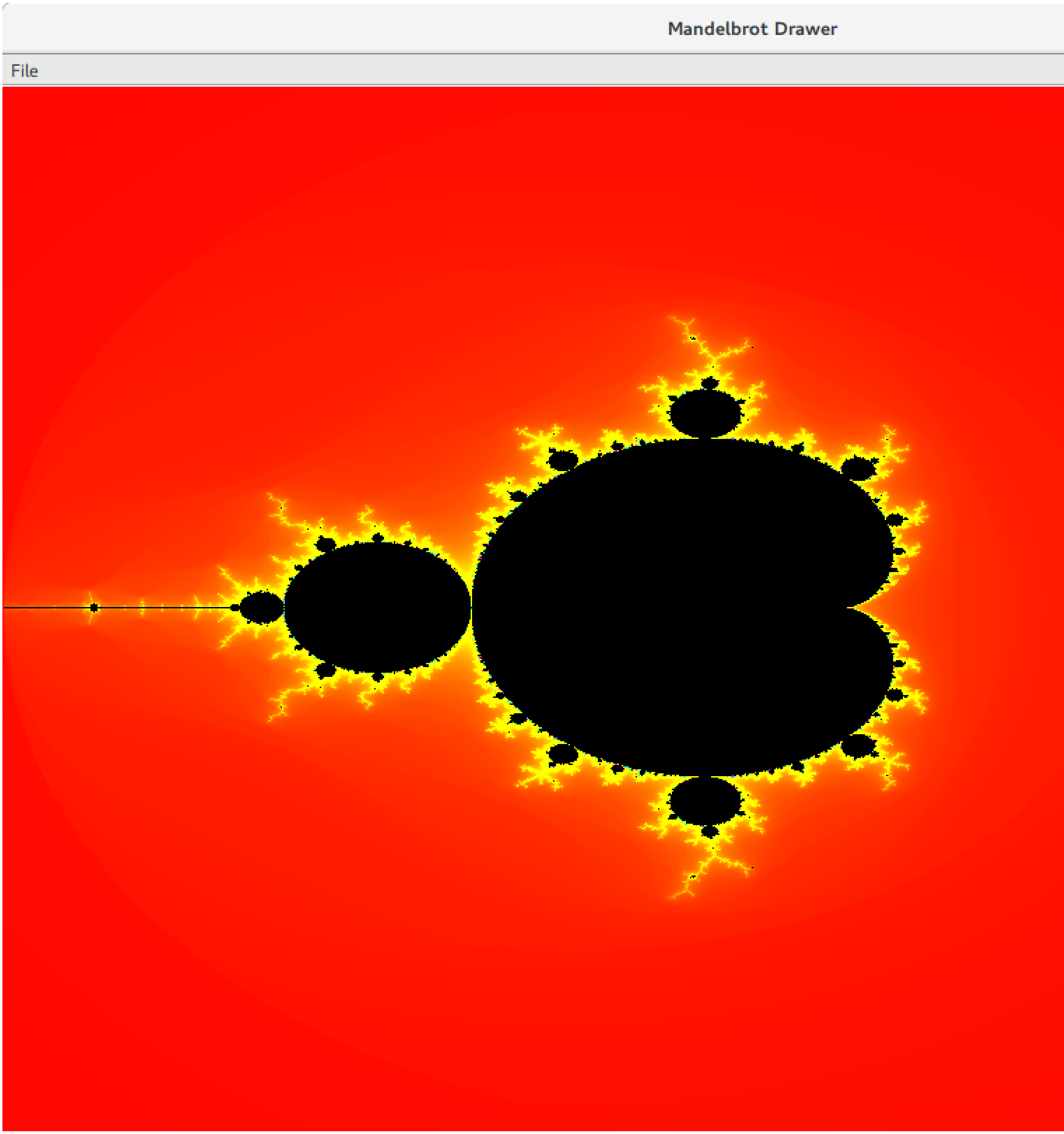
Here's another example (a visually more pleasing, and more concrete one I hope)
This is to give some basic idea about it's possible usage to our fellow U++ users and newcomers.

```
void Mandelbrot::DrawFractal(const Rectf& r, int ix, int wcount)
{
 // This method draws the famous Mandelbrot fractal, using the escape time algorithm (slow).
 // Important note: Imagebuffer is accessed without using any locking mechansim.
 // While it is OK here, do not do this in real life unless you know what you're doing!

 auto sz = GetSize();
 Vector<Rect> regions;
 Sizef scale = iscale(sz, Sizef(1.0, 1.0), r.GetSize());

 for(int i = 0, cx = sz.cx / wcount, mod = sz.cx % wcount; i < wcount; i++) {
  auto x = i * cx;
  regions.Add(Rect(x, 0, x + cx + (i == wcount - 1 ? mod : 0), sz.cy));
 }
 ImageBuffer canvas(sz);

 if(CoWork::GetPoolSize() < wcount)
  CoWork::SetPoolSize(wcount);
 CoWork co;
 {
  RTIMING("Mandelbrot calculation with CoWork");
  co * [&] {
   int j = 0;
   while((j = co.Next()) < regions.GetCount()) {
    const auto& rr = regions[j];
    for(auto y = rr.top; y < rr.bottom; y++) {
     RGBA *pixel = canvas[y];
     for(auto x = rr.left; x < rr.right; x++) {
      Complex c(r.left + x / scale.cx, r.top + y / scale.cy);
      auto z = c;
      auto i = 0;
      while(abs(z) < 2 && i < ix) {
       z = z * z + c;
       i++;
      }
      if(i < ix) {
        double jj = i + 1 - log(log2(abs(z)));
        *(pixel + x) = HsvColorf(jj / double(ix),  1.0, (double(ix) / 256.0) * 1.888);
      }
      else *(pixel + x) = Black();
     }
    }
   }
  };
 }
 img = canvas;
 Refresh();
}
```

Best regards,
Oblivion

File Attachments

1) Mandelbrot.png, downloaded 749 times


Mandelbrot Drawer
File