Subject: Funny way how NOT to speedup sorting of small arrays
Posted by mirek on Sun, 01 Feb 2009 10:30:52 GMT
View Forum Message <> Reply to Message

Well, I got a nice idea that does not work, but I want to share anyway.

One of critical issues in polygon rasterizer (which I could not resist to work on in the end) is the scanline x positions sorting.

Usually the number of elements to sort is quite low. So I was thinking about optimizing selection sort:

```
template <class I, class Less>
inline I SelectMin2(I ptr, const Less& less)
{
 return less(ptr[0], ptr[1]) ? ptr : ptr + 1;
}

template <class I, class Less>
inline I SelectMin3(I ptr, const Less& less)
{
 I l = SelectMin2(ptr, less);
 I h = ptr + 2;
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin4(I ptr, const Less& less)
{
 I l = SelectMin2(ptr, less);
 I h = SelectMin2(ptr + 2, less);
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin5(I ptr, const Less& less)
{
 I l = SelectMin4(ptr, less);
 I h = ptr + 4;
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin6(I ptr, const Less& less)
{
 I l = SelectMin4(ptr, less);
 I h = SelectMin2(ptr + 4, less);
```

```
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin7(I ptr, const Less& less)
{
 I l = SelectMin4(ptr, less);
 I h = SelectMin3(ptr + 4, less);
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin8(I ptr, const Less& less)
{
 I l = SelectMin4(ptr, less);
 I h = SelectMin4(ptr + 4, less);
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin9(I ptr, const Less& less)
{
 I l = SelectMin8(ptr, less);
 I h = ptr + 8;
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin10(I ptr, const Less& less)
{
 I l = SelectMin8(ptr, less);
 I h = SelectMin2(ptr + 8, less);
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin11(I ptr, const Less& less)
{
 I l = SelectMin8(ptr, less);
 I h = SelectMin3(ptr + 8, less);
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin12(I ptr, const Less& less)
{
 I l = SelectMin8(ptr, less);
 I h = SelectMin4(ptr + 8, less);
```

```
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin13(I ptr, const Less& less)
{
 I l = SelectMin8(ptr, less);
 I h = SelectMin5(ptr + 8, less);
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin14(I ptr, const Less& less)
{
 I l = SelectMin8(ptr, less);
 I h = SelectMin6(ptr + 8, less);
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin15(I ptr, const Less& less)
{
 I l = SelectMin8(ptr, less);
 I h = SelectMin7(ptr + 8, less);
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
inline I SelectMin16(I ptr, const Less& less)
{
 I l = SelectMin8(ptr, less);
 I h = SelectMin8(ptr + 8, less);
 return less(*l, *h) ? l : h;
}

template <class I, class Less>
void FwSort(I begin, int len, const Less& less)
{
 switch(len) {
 case 16: IterSwap(begin, SelectMin16(begin, less)); begin++;
 case 15: IterSwap(begin, SelectMin15(begin, less)); begin++;
 case 14: IterSwap(begin, SelectMin14(begin, less)); begin++;
 case 13: IterSwap(begin, SelectMin13(begin, less)); begin++;
 case 12: IterSwap(begin, SelectMin12(begin, less)); begin++;
 case 11: IterSwap(begin, SelectMin11(begin, less)); begin++;
 case 10: IterSwap(begin, SelectMin10(begin, less)); begin++;
 case  9: IterSwap(begin, SelectMin9(begin, less)); begin++;
 case  8: IterSwap(begin, SelectMin8(begin, less)); begin++;
```

```
 case  7: IterSwap(begin, SelectMin7(begin, less)); begin++;
 case  6: IterSwap(begin, SelectMin6(begin, less)); begin++;
 case  5: IterSwap(begin, SelectMin5(begin, less)); begin++;
 case  4: IterSwap(begin, SelectMin4(begin, less)); begin++;
 case  3: IterSwap(begin, SelectMin3(begin, less)); begin++;
 case  2: IterSwap(begin, SelectMin2(begin, less));
 }
}
```

The final result: it works, but it is not faster than normal loop based sort:) Maybe if I could make C++ to emit CMOV, it would be better...

Mirek
