Табличный синус (плавающие числа)

Идея: поскольку | sin'x | < 1, можно считать его значения по таблице с равномерным шагом, равным требуемой точности результата.

Реализация:

```
diff -ruN src-org/sunset.cpp src-more/sunset.cpp
--- src-org/sunset.cpp  2007-09-16 12:04:44.000000000 +0400
+++ src-more/sunset.cpp 2007-11-07 09:55:27.000000000 +0300
@@ -45,6 +45,8 @@
 #include <omp.h>
 #endif
 #include "sunset.h"
+#include <ippvm.h>
+#include <ippm.h>

 #define MIN(x,y)    (((x) < (y)) ? (x) : (y))
 #define MAX(x,y)    (((x) < (y)) ? (y) : (x))
@@ -115,6 +117,33 @@
     }
 }

+#define SIN_TAB_SZ 16384
+#define PI2 (2*3.141592653589793f)
+float g_sinTab[2*SIN_TAB_SZ+1];
+
+void fillSinTab()
+{
+  for(int i=0;i<2*SIN_TAB_SZ+1;++i)
+  {
+    g_sinTab[i]=sinf((i-SIN_TAB_SZ)*PI2/SIN_TAB_SZ);
+  }
+}
+
+inline float tab_sinf(float v)
+{
+  int idx=(((int)(v*(SIN_TAB_SZ/PI2))) % SIN_TAB_SZ)+SIN_TAB_SZ;
+  return g_sinTab[idx];
+}
+
+inline float lin_sinf(float v)
+{
+  float i=fmodf(v,PI2)*SIN_TAB_SZ/PI2+SIN_TAB_SZ;
+  size_t idx=i;
+  float d=i-idx;
+
+  return g_sinTab[idx]*(1-d)+g_sinTab[idx+1]*d;
+}
```

```
+
 /*-------------------------------------------------------------*/

 #define DECLARE_ALIGNED_PTR(type, pName) \
@@ -320,6 +349,7 @@
 /*-------------------------------------------------------------------------
--*/
     if(iCurFrame == 1)
     {
+        fillSinTab();
         iWaveMeshSize = iWaveHarmNum * iAngleHarmNum;

         if(iAllocated == 1)
@@ -730,30 +760,34 @@
 !!!!!!!!!!!!!!!! Water surface modelling !!!!!!!!!!!!!!!!!!!!!!!!
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 */
+                pFlTmp = flArgSin[currentthread].aptr;
+                flDerivX = 0.;
+                flDerivY = 0.;
+#if 1
                 for(t = 0; t < NKMAX; t++)
                 {
-                    OT  = flOmega[t] * flTime;
+                    OT  = flOmegaTime[t];
                     KX1 = flK[t] * flDecartX[i][j];
                     KY1 = flK[t] * flDecartY[i][j];

                     for(l = 0; l < iAngleHarmNum; l++)
                     {
                         iSinIndex1 = t * iAngleHarmNum + l;
-                        flArgSin[currentthread].aptr[iSinIndex1] = OT -
+                        pFlTmp[iSinIndex1] = OT -
                             KX1 * flAzimuthCosFi[l] - KY1 *
flAzimuthSinFi[l] +
-                            flRandomPhase[t*iAngleHarmNum + l];
+                            flRandomPhase[iSinIndex1];
                     } /* end for l */
                 } /* end for t */

-                pFlTmp = flArgSin[currentthread].aptr;
-
+//                ippsSin_32f_A11(pFlTmp,pFlTmp,iWaveMeshSize);
                 #pragma ivdep
+                #pragma vector
                 for(t=0; t<iWaveMeshSize; t++)
-                    pFlTmp[t] = (float)sinf(pFlTmp[t]);
+                {
+                  pFlTmp[t] = (float)tab_sinf(pFlTmp[t]);
```

```
+                }

                 /* initialize the values of derivation */
-                flDerivX = 0.0f;
-                flDerivY = 0.0f;

                 /* dot product to compute derivation */
                 for(t = 0; t < iWaveMeshSize; t++)
@@ -761,6 +795,7 @@
                     flDerivX += pFlTmp[t] * flAmplitudeX[t];
                     flDerivY += pFlTmp[t] * flAmplitudeY[t];
                 }
+#endif

                 /* Near horizont area correction */
                 flDerivX *= P2;
```

Результат: gcc – медленно, icc – догоняем ippsSinf. Основной тормоз – смешанная арифметика, перевод плавающего числа в целое.