

Исследование вопроса

пока я выложу свои изыскания. если появятся альтернативы - сделаем разные странички, или секции - как будет удобнее.

План

1. Найти hotspots
2. Сравнить их относительный вклад во время работы
3. Наметить пути оптимизации

Hotspots

Профилировал gcc/gcov. Для этого патчил Makefile (cd src ; patch -p1 Makefile-profiling.patch):

```
diff -ruN src-org/Makefile src1/Makefile
--- src-org/Makefile 2007-09-17 17:43:08.000000000 +0400
+++ src1/Makefile 2007-10-31 22:58:53.000000000 +0300
@@ -59,14 +59,14 @@
 CINC      = -I$(SRC_DIR)
 CDEFS     =
 COBJ      = -c -o$(OBJ_DIR)/$@
-CDEFOPT   = -O2
+CDEFOPT   = -g -pg -fprofile-arcs -ftest-coverage
 COPT      =
-CFLAGS    =
+CFLAGS    = -O3
 CFLAGS_ALL = $(CFLAGS) $(CINC) $(CDEFS) $(CDEFOPT) $(CPROC) $(CPLAT)

 LD        = g++
 LDPLAT   =
-LDFLAGS   =
+LDFLAGS   = -g -pg -ax -fprofile-arcs -ftest-coverage
 LDOUTOPT = -o "$(OUT_DIR)/$(BENCHMARK)"
 LIBS     = -lm -lc
 LIBS_ALL = $(LIBS)
```

Результаты (./sunset -cfg ../input/Sample01.cfg) - основное:

```
  - :
729:!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
  - : 730:!!!!!!!!!!!!!!!!!!!! Water surface modelling
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
  - :
731:!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
  - : 732:*/
```

```
247782400: 733:         for(t = 0; t < NKMAX; t++)
-: 734:         {
240217600: 735:             OT = flOmega[t] * flTime;
240217600: 736:             KX1 = flK[t] * flDecartX[i][j];
240217600: 737:             KY1 = flK[t] * flDecartY[i][j];
-: 738:
7927180800: 739:             for(l = 0; l < iAngleHarmNum; l++)
-: 740:             {
7686963200: 741:                 iSinIndex1 = t * iAngleHarmNum + l;
-: 742:
flArgSin[currentthread].aptr[iSinIndex1] = OT -
-: 743:                 KX1 * flAzimuthCosFi[l] - KY1 *
flAzimuthSinFi[l] +
7686963200: 744:                 flRandomPhase[t*iAngleHarmNum +
l];
-: 745:                 } /* end for l */
-: 746:             } /* end for t */
-: 747:
7564800: 748:             pFlTmp = flArgSin[currentthread].aptr;
-: 749:
-: 750:             #pragma ivdep
7753920000: 751:             for(t=0; t<iWaveMeshSize; t++)
7746355200: 752:                 pFlTmp[t] = (float)sinf(pFlTmp[t]);
-: 753:
-: 754:             /* initialize the values of derivation */
7564800: 755:             flDerivX = 0.0f;
7564800: 756:             flDerivY = 0.0f;
-: 757:
-: 758:             /* dot product to compute derivation */
7753920000: 759:             for(t = 0; t < iWaveMeshSize; t++)
-: 760:             {
7746355200: 761:                 flDerivX += pFlTmp[t] *
flAmplitudeX[t];
7746355200: 762:                 flDerivY += pFlTmp[t] *
flAmplitudeY[t];
-: 763:             }
-: 764:
```

Интерпретация: алгоритм проходит по всем точкам изображения (7564800 действий). Для прообраза каждой точки изображения, находящегося на поверхности воды рассчитывается $iWaveHarmNum * iAngleHarmNum$ дополнительных значений (во всех примерах - $32*32 == 1024$). Это - аргументы синусов, сами синусы и скалярные произведения амплитуд на эти синусы (7746355200 действий).

Начальное время на моей машине: 45.457/кадр

Вклад во время

закомментировал блок целиком:

```
diff -ruN src-org/sunset.cpp src1/sunset.cpp
--- src-org/sunset.cpp 2007-09-16 12:04:44.000000000 +0400
+++ src1/sunset.cpp 2007-10-31 23:34:04.000000000 +0300
@@ -730,6 +730,11 @@
!!!!!!!!!!!!!!!!!!!!!! Water surface modelling !!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!
*/
+
+      /* initialize the values of derivation */
+      flDerivX = 0.0f;
+      flDerivY = 0.0f;
+
+#if 0
+
+      for(t = 0; t < NKMAX; t++)
+      {
+          OT = flOmega[t] * flTime;
@@ -751,17 +756,13 @@
+      for(t=0; t<iWaveMeshSize; t++)
+          pFlTmp[t] = (float)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++)
-      {
-          flDerivX += pFlTmp[t] * flAmplitudeX[t];
-          flDerivY += pFlTmp[t] * flAmplitudeY[t];
-      }
-
+#endif
+
+      /* Near horizont area correction */
+      flDerivX *= P2;
+      flDerivY *= P2;
```

Время: 1.225/кадр.

1. Оставил только расчет аргументов. Время: 4.771/кадр.
2. Оставил только расчет скалярного произведения. Время: 2.436/кадр.

Результат: основное время уходит в тригонометрию, затем – в генерацию аргументов, затем – в скалярное произведение.

Пути оптимизации

From:
<http://wiki.osll.ru/> - **Open Source & Linux Lab**

Permanent link:
http://wiki.osll.ru/doku.php/etc:common_activities:intel_students_cup:tour2?rev=1193860087

Last update: **2008/01/03 02:32**

