```cpp
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <cmath>
#include <fstream>
#include <vector>
#include <algorithm>
#include <cstring>

#include "Angles.hpp"
#include "GPData.hpp"

using namespace std;

//-----------------------------------------------------------------------
//  Purpose: Class Angles Implementation Files
//
//     [Angles.1.b1.plot_angle_tree.cpp]
//
//      Angles::plot_angle_tree()
//
//      - to plot a binary tree angles
//
//  Discussion:
//
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    2013.07.27
//
//  Author:
//
//    Young Won Lim
//
//  Parameters:
//      m : m-th level
//      n : n-th node in the m-th level
//
//  Outputs:
//      egb1.All_11.ang_tree1.n4095.eps
//      egb1.All_11.ang_tree2.n4095.eps
//      egb1.All_11.ang_tree3.n4095.eps
//      egb1.Leaf_11.sub_tree1.n2048.eps
//      egb1.Leaf_11.sub_tree2.n2048.eps
//
//-----------------------------------------------------------------------
void B1_plot_subtree(int m, int mode, char * fname, Angles *Ang);
void B1_plot_ancestor(int m, int mode, char * fname, Angles *Ang);
void B1_run_gnuplot(Angles *Ang, GPData *G, int mode, int m);

//-----------------------------------------------------------------------
//  Plot a binary angle tree
//-----------------------------------------------------------------------
//  the [n]-th node in the [m]-th level
//-----------------------------------------------------------------------
void Angles::plot_angle_tree (int m, int n)
{

  // int level, leaves;
  int mode;


  if (checkNIters("plot_angle_tree")) return;
```

```
  // cout << "nIters  = " << nIters << endl;
  // cout << "nAngles = " << nAngles << endl;


  ofstream myout;
  char fname[256];

  //.....................................................................
  // plot subtree  : mode=0: (Leaf) : block index view
  //                 mode=0: (All)  : block index view
  //                 mode=1: (Leaf) : offset index view
  //                 mode=1: (All)  : offset index view
  //.....................................................................
  // plot ancestor : mode=2: (Leaf) : common ancestor view
  //                 mode=3: (All)  : different ancestor view
  //.....................................................................
  // the [n]-th node in the [m]-th level
  // in the [m]-th level, there are 2^m nodes, so 2^m subtrees(subblocks)
  // plot leaf arrows for each of 2^m subtrees(subblocks)
  // (2^nIters) / (2^m) leaves belong to each subtree
  // to see if overlapped angle ranges between subtrees
  //.....................................................................

  for (mode=0; mode<=2; ++mode) {

    // "angle0" (mode=0: block view)
    // "angle1" (mode=1: offset view)
    // "angle2" (mode=2: common ancestor view)
    // "angle3" (mode=3: different ancstor view)

    sprintf(fname, "angle%d", mode);

    GPData G(GnuTerm, nAngles);

    switch (mode) {
      case 0:
      case 1: B1_plot_subtree(m, mode, fname, this);
              break;
      case 2:
      case 3: B1_plot_ancestor(m, mode, fname, this);
              break;
      default: break;
    }

    //--------------------------------------
    B1_run_gnuplot(this, &G, mode, m);  // mode=0,1,2,3
    //--------------------------------------

  }

  return;

}


//-----------------------------------------------------------------------------
// B1_plot_subtree
//-----------------------------------------------------------------------------
//  the n-th node in the [m]-th level
//  mode=0: height --> i: block index   : "angle0"
//  mode=1: height --> j: offset index  : "angle1"
//-----------------------------------------------------------------------------
void B1_plot_subtree(int m, int mode, char * fname, Angles *Ang)
{
    int nIters = Ang->getnIters();
    int i, j, k, leaves, gsize;
```

```
int l, x, y;
char fname1[256], fname2[256];

if (mode >= 2) {
  printf("* plot_subtree : mode=0, 1\n");
  return;
}

ofstream myout;

// mode=0: fname1="angle0.dat", fname2="angle0a.dat"
// mode=1: fname1="angle1.dat", fname2="angle1a.dat"
strcpy(fname1, fname);
strcat(fname1, ".dat");
strcpy(fname2, fname);
strcat(fname2, "a.dat");

// group size: 2^m nodes (subtrees) in the [m]-th level
gsize  = 1 << m;
leaves = 1 << nIters; // no of leaves

int index;   // leaf node index of A
int offset;  // index to where the leaf node begins in A
double h;    // arrow height 1.0 for Leaf, 0.5 for All nodes

if (Ang->getLeaf()) {
  h = 1.0;  offset = 0;
}
else {
  h = 0.5;  offset = leaves -1;
}

// "angle0.dat" : block index view
// "angle1.dat" : offset index view
myout.open(fname1);

for (i=0; i<gsize; ++i) {
  for (j=0; j<leaves/gsize; ++j) {

    // mode=0: height --> i: block index (inter group index)
    // mode=1: height --> j: offset index (intra group index)
    k = ((mode==0) ? i : j);

    index = offset + i*(leaves/gsize)+j;
    myout << Ang->A[index]*180/pi << " ";
    myout << k << " 0.0 " << h << endl;
  }
}

myout.close();


if (Ang->getLeaf()) return;


// "angle0a.dat" : block index view (All nodes)
// "angle1a.dat" : offset index view (All nodes)
myout.open(fname2);

// gsize = 2^m : no of nodes at the level m
// these nodes start at gsize - 1 in A
// i : index enumerating nodes at the level m
// for each node, consider its subtree
// l : index of the first node as the level y increases
// x : index enumerating nodes at the given level y
// y : increasing values of the levels from m to nIters
for (i=0; i<gsize; ++i) {
```

```
      l = gsize - 1 + i;
      j=0;
      for (y=m; y<nIters; ++y) {
        for (x=0; x<(1<<(y-m)); ++x) {
          k = ((mode==0) ? i : j);
          myout << Ang->A[l+x]*180/pi << " ";
          myout << k+0.5 << " 0.0 0.5" << endl;
          j++;
        }
        l = l*2 +1;
      }
    }

    myout.close();

}


//-----------------------------------------------------------------------------
// B1_plot_ancestor
//-----------------------------------------------------------------------------
//  the n-th node in the [m]-th level
//  mode=2: common ancestor view     : "angle2"
//  mode=3: different ancestor view  : "angle3"
//-----------------------------------------------------------------------------
void B1_plot_ancestor(int m, int mode, char * fname, Angles *Ang)
{
    int nIters = Ang->getnIters();
    int i, j, k, leaves, gsize;
    int l, x, y;
    char fname1[256], fname2[256];

    if (mode <= 1) {
      printf("* plot_ancestor : mode=2, 3\n");
      return;
    }

    ofstream myout;


    // "angle2.dat" (mode=2: common ancestor view)
    // "angle3.dat" (mode=3: different ancstor view)
    strcpy(fname1, fname);
    strcat(fname1, ".dat");


    // group size: 2^m nodes (subtrees) in the [m]-th level
    gsize  = 1 << m;
    leaves = 1 << nIters; // no of leaves

    //-------------------------------------------------
    if (Ang->getLeaf() && mode==2) {
    //-------------------------------------------------

    myout.open(fname1);  // "angle2.dat"

    for (i=1; i<nIters; ++i) {
      for (j=0, k=0; j<leaves; j++) {
        if (j%(1<<i) == 0) k = (k+1)%2;
         myout << Ang->A[j]*180/pi << " ";
         myout << i+k*0.4 << " 0.0 0.4 " ;
         myout << (j/(1<<i)+2+3)%8+1 << endl;  // 5th field for line types
      }
    }

    myout.close();
```

```
// system("ls -l angle2.dat");
char cmd[256];
char prn[256] = "{print $1, $2, $3, $4}";


// fname2="angle2.1.dat" ... "angle2.8.dat"
for (int t=1; t<9; ++t) { // line type index
  sprintf(fname2, "%s.%d.dat", fname, t);
  sprintf(cmd, "awk '$5==%d %s' < %s > %s", t, prn, fname1, fname2);
  system(cmd);
}

/*
system("awk '$5==1 {print $1, $2, $3, $4}' < angle3.dat > angle3.1.dat");
system("awk '$5==2 {print $1, $2, $3, $4}' < angle3.dat > angle3.2.dat");
system("awk '$5==3 {print $1, $2, $3, $4}' < angle3.dat > angle3.3.dat");
system("awk '$5==4 {print $1, $2, $3, $4}' < angle3.dat > angle3.4.dat");
system("awk '$5==5 {print $1, $2, $3, $4}' < angle3.dat > angle3.5.dat");
system("awk '$5==6 {print $1, $2, $3, $4}' < angle3.dat > angle3.6.dat");
system("awk '$5==7 {print $1, $2, $3, $4}' < angle3.dat > angle3.7.dat");
system("awk '$5==8 {print $1, $2, $3, $4}' < angle3.dat > angle3.8.dat");
*/

//-------------------------------------------------
} else if (~(Ang->getLeaf()) && mode==3) {
//-------------------------------------------------

// mode=3: fname1="angle3.dat", fname2="angle3a.dat"
strcpy(fname1, fname);
strcat(fname1, ".dat");
strcpy(fname2, fname);
strcat(fname2, "a.dat");


myout.open(fname1); // "angle3.dat"

k=0;

// i: the tree level index
for (i=0; i<=nIters; ++i) {
  level = i;
  leaves = 1 << level;
  for (j=0; j<leaves; ++j) {
    // ancestor condition
    cond1 = (i <= m) && (j == n / (1 << (m-i))) ;

    // descendant condition
    minj = n * (1 << (i-m));
    maxj = (n+1) * (1 << (i-m));
    cond2 = (i >  m) && (minj <= j) && (j < maxj);

    if (cond1 || cond2 ) {
    // printf("[i=%d j=%d] \n", i, j);

    myout << A[k+j]*180/pi << " " <<  i << " 0.0 1.0" << endl;
    }
  }
  k += leaves;
}

myout.close(fname1); // "angle3.dat"


// using final minj & maxj
// mark all arrows in the range of [minj, maxj]

myout.open(fname2); // "angle3a.dat"
```

```
    k=0;

    // i: the tree level index
    for (i=0; i<=nIters; ++i) {
      level = i;
      leaves = 1 << level;
      for (j=0; j<leaves; ++j) {
        // ancestor condition
        cond2 = (minj <= j) && (j < maxj);

        if (cond2) {
        // printf("[i=%d j=%d] \n", i, j);

        myout << A[k+j]*180/pi << " " <<  i << " 0.0 1.0" << endl;
        }
      }
      k += leaves;
    }

    myout.close(fname2);  // "angle3a.dat"

    //-------------------------------------------------
    }
    //-------------------------------------------------

    return;

 }




//-----------------------------------------------------------------------------
// run_gnuplot
//-----------------------------------------------------------------------------
// plot_subtree : mode=0: block index view
//              : mode=1: offset index view
//-----------------------------------------------------------------------------
// All:  mode=1 : (m, n-1) descendants
// All:  mode=2 : (m, n-1) & (m, n) descendants
// All:  mode=3 : (m, n-1) & (m, n) & (m, n+1) descendants
//-----------------------------------------------------------------------------
void B1_run_gnuplot(Angles *Ang, GPData *G, int mode, int m)
{
  ofstream myout;

  // writing gnuplot commands
  myout.open("command.gp");

  G->set_prefix(Ang);
  G->set_suffix(Ang);

  myout << "set terminal " << GnuTerm << endl;

  char fstr[256];

  if (strcmp(GnuTerm.c_str(), "wxt") != 0) {
    sprintf(fstr, "sub_tree%d", mode);
    G->set_fname(Ang, "egb1", fstr);
    Ang->epsList.push_back(G->fname);
    cout << "set output '" << G->fname << "'" << endl;
    // cout  << "pause" << endl;
    myout << "set output '" << G->fname << "'" << endl;
  }
```

```
//-------------------------------------------------------------
// Leaf: mode=0 : block index view
// Leaf: mode=1 : offset index view
//-------------------------------------------------------------
char tstr[256];

//-----------------------------------------------------
if (mode == 0) {
//-----------------------------------------------------

  sprintf(tstr, "Subtree plot of a level %d nodes (block index view)", m);
  G->set_title(Ang, tstr);

  myout << "set title '" << G->title << "' " << endl;
  myout << "set xlabel \"Angles in degree\" " << endl;
  myout << "set ylabel \"block index \" " << endl;
  myout << "set format x \"%.0f\" " << endl;
  myout << "set format y \"%.0f\" " << endl;

  myout << "set xrange [-100:100]" << endl;

  myout << "plot 'angle0.dat' using 1:2:3:4  notitle  ";
  myout << "with vectors head filled lt 3  " ;

  if (!(Ang->getLeaf())) {
    myout << ", 'angle0a.dat' using 1:2:3:4  notitle  ";
    myout << "with vectors head filled lt 4  " ;
  }

  myout << endl;


  if (strcmp(GnuTerm.c_str(), "wxt") == 0)
    myout << "pause mouse keypress" << endl;

//-----------------------------------------------------
} else if (mode == 1) {
//-----------------------------------------------------

  sprintf(tstr, "Subtree plot of a level %d nodes (offset index view)", m);
  G->set_title(Ang, tstr);

  myout << "set title '" << G->title << "' " << endl;
  myout << "set xlabel \"Angles in degree\" " << endl;
  myout << "set ylabel \"offset index \" " << endl;
  myout << "set format x \"%.0f\" " << endl;
  myout << "set format y \"%.0f\" " << endl;

  myout << "set xrange [-100:100]" << endl;

  myout << "plot 'angle1.dat' using 1:2:3:4  notitle ";
  myout << "with vectors head filled lt 3  " ;

  if (!(Ang->getLeaf())) {
    myout << ", 'angle1a.dat' using 1:2:3:4  notitle  ";
    myout << "with vectors head filled lt 4  ";
  }

  myout << endl;

  if (strcmp(GnuTerm.c_str(), "wxt") == 0)
    myout << "pause mouse keypress" << endl;

//-----------------------------------------------------
} else if (Ang->getLeaf() && mode==2) {
//-----------------------------------------------------
```

```
    sprintf(tstr, "Ancestor plot of a level %d nodes ", m);
    G->set_title(Ang, tstr);

    myout << "set title '" << G->title << "' " << endl;
    myout << "set xlabel \"Angles in degree\" " << endl;
    myout << "set ylabel \"offset index \" " << endl;
    myout << "set format x \"%.0f\" " << endl;
    myout << "set format y \"%.0f\" " << endl;

    myout << "set xrange [-100:100]" << endl;

    myout << "plot 'angle2.1.dat' using 1:2:3:4  notitle ";
    myout << "with vectors head filled lt 1,  " ;
    myout << "     'angle2.2.dat' using 1:2:3:4  notitle ";
    myout << "with vectors head filled lt 2,  " ;
    myout << "     'angle2.3.dat' using 1:2:3:4  notitle ";
    myout << "with vectors head filled lt 3,  " ;
    myout << "     'angle2.4.dat' using 1:2:3:4  notitle ";
    myout << "with vectors head filled lt 4,  " ;
    myout << "     'angle2.5.dat' using 1:2:3:4  notitle ";
    myout << "with vectors head filled lt 5,  " ;
    myout << "     'angle2.6.dat' using 1:2:3:4  notitle ";
    myout << "with vectors head filled lt 6,  " ;
    myout << "     'angle2.7.dat' using 1:2:3:4  notitle ";
    myout << "with vectors head filled lt 7,  " ;
    myout << "     'angle2.8.dat' using 1:2:3:4  notitle ";
    myout << "with vectors head filled lt 8  " ;


    myout << endl;

    if (strcmp(GnuTerm.c_str(), "wxt") == 0)
      myout << "pause mouse keypress" << endl;

//-----------------------------------------------------
} else if (~Ang->getLeaf() && mode==3) {
//-----------------------------------------------------



//-----------------------------------------------------
}
//-----------------------------------------------------

//-------------------------------------------------------------
// All: mode=1 : (m, n-1) descendants
// All: mode=2 : (m, n-1) & (m, n) descendants
// All: mode=3 : (m, n-1) & (m, n) & (m, n+1) descendants
//-------------------------------------------------------------
/* } else {
  if (mode == 1) {
    G->set_title(Ang, "Binary Angle Tree - consecutive subtrees 1");

    myout << "plot 'angle1.dat' using 1:2:3:4  notitle ";
    myout << "with vectors head filled lt 3  " << endl;

    if (strcmp(GnuTerm.c_str(), "wxt") == 0)
      myout << "pause mouse keypress" << endl;
  } else if (mode == 2) {
    G->set_title(Ang, "Binary Angle Tree - consecutive subtrees 1,2");

    myout << "plot 'angle1.dat' using 1:2:3:4  notitle  ";
    myout << "with vectors head filled lt 3 ,  " ;
    myout << "     'angle2.dat' using 1:2:3:4  notitle ";
    myout << "with vectors head filled lt 4  " << endl;

    if (strcmp(GnuTerm.c_str(), "wxt") == 0)
```

```
        myout << "pause mouse keypress" << endl;
    } else if (mode == 3) {
      G->set_title(Ang, "Binary Angle Tree - consecutive subtrees 1,2,3");

      myout << "plot 'angle1.dat' using 1:2:3:4  notitle ";
      myout << "with vectors head filled lt 3 ,  " ;
      myout << "     'angle2.dat' using 1:2:3:4  notitle ";
      myout << "with vectors head filled lt 4  , " ;
      myout << "     'angle3.dat' using 1:2:3:4  notitle ";
      myout << "with vectors head filled lt 5" << endl;

      if (strcmp(GnuTerm.c_str(), "wxt") == 0)
        myout << "pause mouse keypress" << endl;
    } */
  //-------------------------------------------------------------


  myout.close();

  cout << "............................................................." << endl;
  cout << G->title << endl;
  cout << "............................................................." << endl;

  system("gnuplot command.gp");

}
```