```
/*
 * File:   tracker.cpp
 * Author: Mike
 *
 * Created on June 22, 2014, 11:08 AM
 */

#include "tracker.h"

#define sqr(a)  ((a) * (a))

int tracker::randomInt(int min, int max) {
    return (min + (rand() % (int) (max - min + 1)));
}

tracker::tracker() {
    nTracks = 0;
}

tracker::tracker(Mat *imgOut) {
    nTracks = 0;
    img = imgOut;
}

tracker::tracker(const tracker& orig) {
}

tracker::~tracker() {
}

list<PotentialBug>::iterator tracker::findNearestNeighbor(PotentialBug& pb, double *MinDis2, double
current, bool newPoint) {
    double MinDis = 100000000, dis;
    Point2f old;
    //I think there is a problem here.  The method can run without setting it.  Perhaps start by setting it to
potentialBugs.end()?)
    list<PotentialBug>::iterator it;
    it = potentialBugs.end();
    current = sqr(current);
    for (list<PotentialBug>::iterator list_iter = potentialBugs.begin();
            list_iter != potentialBugs.end(); list_iter++) {
        old = list_iter->GetLocation();
        dis = (sqr(old.x - pb.GetLocation().x) + sqr(old.y - pb.GetLocation().y));
        if (dis < MinDis && ((dis > (current + 0.00001)) || newPoint)) {
            //if (dis < MinDis) {
            MinDis = dis;
            it = list_iter;
        }
    }
    *MinDis2 = sqrt(MinDis);
    return it;
}

void tracker::evaluatePoint(PotentialBug *pb, unsigned int frameCount) {
    double NNDistance;
    list<PotentialBug>::iterator it, it2;
    PotentialBug replacedBug;
    vector<PotentialBug> OneBug;
    double dist2, dist1, dist3;
    int idx, tmpdex;

    it = findNearestNeighbor(*pb, &NNDistance, 0, true);

    if (NNDistance > TrackerMinDistance ||(it == potentialBugs.end())) { //This seems to be a new bug
        pb->SetLineColor(Scalar(randomInt(0, 255), randomInt(0, 255), randomInt(0, 255)));
        pb->SetIndex(nTracks++);
        vector<PotentialBug> newvec;
        newvec.push_back(*pb);
        tracks.push_back(newvec);
        potentialBugs.push_front(*pb);
        //OneBug = tracks[index];
    } else {
        OneBug = tracks[it->GetIndex()];
        //tmpdex = it->GetIndex();

        while (OneBug.size() > 1 && (pb->GetLastFrame() == OneBug[OneBug.size() - 1].GetLastFrame())) {
           // int tmpidx = pb->GetLastFrame();
           // int tmpidx2 = OneBug[OneBug.size() - 1].GetLastFrame();
            //int tmp = it->GetIndex();
            //int tmp2 = tracks.size();
            //if (OneBug.size() > 1 && (pb->GetLastFrame() == OneBug[OneBug.size() - 1].GetLastFrame())) {
//We have a conflict
            idx = OneBug.size() - 1;
            dist2 = sqrt(sqr(OneBug[idx].GetLocation().x - OneBug[idx - 1].GetLocation().x) +
sqr(OneBug[idx].GetLocation().y - OneBug[idx - 1].GetLocation().y));
            if (NNDistance < dist2) { //we need to swap in pb and continue search for old value
                replacedBug = OneBug[idx];
                unsigned tmp3 = pb->GetIndex();
                OneBug[idx] = *pb;
                *pb = replacedBug;
                it = findNearestNeighbor(*pb, &NNDistance, dist2, false);
            } else //(dist2 < NNDistance))
                it = findNearestNeighbor(*pb, &NNDistance, NNDistance, false);
            //if (it == 0){
            //    NNDistance = 10000;
            //}
            if (NNDistance > TrackerMinDistance)
               break;
            //tmpdex = it->GetIndex();
            OneBug = tracks[it->GetIndex()];
         }

         if (NNDistance > TrackerMinDistance) { //This seems to be a new bug
            pb->SetLineColor(Scalar(randomInt(0, 255), randomInt(0, 255), randomInt(0, 255)));
            pb->SetIndex(nTracks++);
            vector<PotentialBug> newvec;
            newvec.push_back(*pb);
            tracks.push_back(newvec);
            potentialBugs.push_front(*pb);
            //OneBug = tracks[index];
         } else {
            line(*img, pb->GetLocation(), it->GetLocation(), it->GetLineColor(), 2, 8);
            //tracks[pb->GetIndex()].push_back(*pb);
            //int dex = it->GetIndex();
            it->SetLocation(pb->GetLocation());
            it->SetLastFrame(frameCount);
            //if (NNDistance != 0) {
            tracks[it->GetIndex()].push_back(*pb);
            //}
         }
      }
   }

void tracker::writeATrack(int index, int mAvr, double fps, char * baseFileName, char *trackname) {
   vector<vector<PotentialBug> >::iterator tracked;
   vector<PotentialBug> OneBug;
   char trackFileName[250];
   FILE *trackerFile;
   trackerFile = fopen(baseFileName, "w");
   printf("Writing track %u\n", index);
   //fprintf(trackerFile, "Mean distance %7.4f, mean degrees %7.4f, fps: %7.3f\n",
getTrackAverageDistance(index), getTrackAverageDegreesTurned(index, mAvr), fps);
   //fprintf(trackerFile, "Mean distance %7.4f, mean weighted degrees %7.4f, unweighted degrees %7.4f,
Total distance %7.4f, AverageOver %d, fps: %7.3f\n", getTrackAverageDistance(index, mAvr),
getTrackAverageDegreesTurned(index, mAvr), getTrackAverageUnweighterDegreesTurned(index,
mAvr), getTrackDistance(index, mAvr), mAvr, fps);
   fprintf(trackerFile, "X, Y, Frame \n");

   OneBug = tracks[index];
   for (int j1 = 0; j1 < OneBug.size(); j1++) {
      //fprintf(trackerFile, "%5.2f, %5.2f, %u, %6.3f, %6.3f\n", OneBug[j1].GetLocation().x,
OneBug[j1].GetLocation().y, OneBug[j1].GetLastFrame(), getPointDegreesTurned(index, j1, mAvr),
getPointSignedDegreesTurned(index, j1, mAvr));
```

```
        fprintf(trackerFile, "%5.2f, %5.2f, %u\n", OneBug[j1].GetLocation().x, OneBug[j1].GetLocation().y,
OneBug[j1].GetLastFrame());
    }
    fclose(trackerFile);
}

void tracker::cleanupPotentialBugs(unsigned int currentFrame) {
    for (list<PotentialBug>::iterator list_iter = potentialBugs.begin();
            list_iter != potentialBugs.end(); list_iter++) {
        if ((currentFrame - list_iter->GetLastFrame()) > BUGCLEANOUT) {
            list_iter = potentialBugs.erase(list_iter);
        }
    }
}

unsigned int tracker::getNumTracks() {
    return tracks.size();
}

unsigned int tracker::getTrackSize(unsigned int j) {
    return tracks[j].size();
}

unsigned int tracker::getInitialTrackFrame(unsigned int j) {
    vector<PotentialBug> OneBug;
    OneBug = tracks[j];
    return OneBug[0].GetLastFrame(); //This is a problem, we don't necessarily store the initial frame
}

unsigned int tracker::getFinalTrackFrame(unsigned int j) {
    vector<PotentialBug> OneBug;
    OneBug = tracks[j];
    return OneBug[OneBug.size() - 1].GetLastFrame();
}

double tracker::getTrackAverageDistance(unsigned int j, int secs) {
    double sumDistance = 0.0;
    nPointsInTrack = 0;
    vector<PotentialBug> OneBug;
    OneBug = tracks[j];
    if (OneBug.size() == 1) return 0.0;
    // for (int j1 = 1; j1 < OneBug.size() - 1; j1++) {
    //     sumDistance += sqrt(sqr(OneBug[j1].GetLocation().x - OneBug[j1 - 1].GetLocation().x) +
sqr(OneBug[j1].GetLocation().y - OneBug[j1 - 1].GetLocation().y));
    // }
    // return sumDistance;

    int initialIndex, hingeIndex;
    double sumx1, sumy1, sumx2, sumy2, meanPos1, meanPos2;
    int counter1, counter2;
    for (int j1 = 1; j1 < OneBug.size() - 1; j1++) {
        if (OneBug[j1].GetLastFrame() - OneBug[0].GetLastFrame() < secs)
            continue;
        initialIndex = hingeIndex = j1;
        while (((OneBug[j1].GetLastFrame() - OneBug[initialIndex].GetLastFrame()) < secs) && (initialIndex >
0)) {
            initialIndex--;
        }
        while (((OneBug[j1].GetLastFrame() - OneBug[hingeIndex].GetLastFrame()) < (secs / 2)) &&
(hingeIndex > (initialIndex + 1)))
            hingeIndex--;

        sumx1 = sumy1 = counter1 = 0;
        for (int j2 = initialIndex; j2 < hingeIndex; j2++) {
            sumx1 += OneBug[j2].GetLocation().x;
            sumy1 += OneBug[j2].GetLocation().y;
            counter1++;
        }
        if (counter1 > 0) {
            sumx1 = sumx1 / counter1;
            sumy1 = sumy1 / counter1;
        }

        sumx2 = sumy2 = counter2 = 0;
        for (int j2 = hingeIndex; j2 <= j1; j2++) {
            sumx2 += OneBug[j2].GetLocation().x;
            sumy2 += OneBug[j2].GetLocation().y;
            counter2++;
        }
        if (counter2 > 0) {
            sumx2 /= counter2;
            sumy2 /= counter2;
        }

        if (counter1 > 0 && counter2 > 0) {
            sumDistance += sqrt(sqr(sumx1 - sumx2) + sqr(sumy1 - sumy2));
            nPointsInTrack++;
        }

    }
    return (sumDistance / nPointsInTrack); //The factor of 2 puts it into per second
}

double tracker::getTrackDistance(unsigned int j, int mAvr) {
    vector<PotentialBug> OneBug;
    OneBug = tracks[j];
    if (tracks[j].size() > 1)
        //return getTrackDistance(j) / (OneBug[OneBug.size() - 1].GetLastFrame() -
OneBug[0].GetLastFrame());
        return getTrackAverageDistance(j, mAvr) * nPointsInTrack / mAvr;
    else
        return 0;
}

double tracker::getTrackUnweightedDegreesTurned(unsigned int j, int mAvr) {
    int firstPoint, hingePoint, movingAverage = mAvr;
    nPointsInTrack = 0;
    firstPoint = movingAverage - 1;
    hingePoint = movingAverage / 2;
    double degreesTurned = 0.0;
    double x, y, z, tmp;
    vector<PotentialBug> OneBug;
    OneBug = tracks[j];
    if (OneBug.size() <= firstPoint) return 0.0;
    for (int j1 = firstPoint; j1 < OneBug.size() - 1; j1++) {
        x = sqrt(sqr(OneBug[j1 - hingePoint].GetLocation().x - OneBug[j1 - firstPoint].GetLocation().x) +
sqr(OneBug[j1 - hingePoint].GetLocation().y - OneBug[j1 - firstPoint].GetLocation().y));
        y = sqrt(sqr(OneBug[j1 - hingePoint].GetLocation().x - OneBug[j1].GetLocation().x) + sqr(OneBug[j1 -
hingePoint].GetLocation().y - OneBug[j1].GetLocation().y));
        z = sqrt(sqr(OneBug[j1].GetLocation().x - OneBug[j1 - firstPoint].GetLocation().x) +
sqr(OneBug[j1].GetLocation().y - OneBug[j1 - firstPoint].GetLocation().y));
        //acos domain is -1 to 1
        nPointsInTrack++;
        if (z == (x + y)) //There was no turn
            continue;
        if (z == abs(x - y)) {//180 degree turn
            degreesTurned += 180;
            continue;
        }
        tmp = (sqr(x) + sqr(y) - sqr(z)) / (2 * x * y);
        if (tmp <= 1.0 && tmp >= -1.0)
            degreesTurned += (180 - (acos(tmp) * 180 / 3.14159));
    }
    return degreesTurned / nPointsInTrack;
}

double tracker::getTrackAverageUnweighterDegreesTurned(unsigned int j, int mAvr) {
    vector<PotentialBug> OneBug;
    OneBug = tracks[j];
    if (tracks[j].size() > 1)
        //return getTrackDegreesTurned(j, mAvr) / (OneBug[OneBug.size() - 1].GetLastFrame() -
OneBug[0].GetLastFrame());
        return getTrackUnweightedDegreesTurned(j, mAvr);
    else
```

```
        return 0;
}

double tracker::getTrackDegreesTurned(unsigned int j, int mAvr) {
    int firstPoint, hingePoint, movingAverage = mAvr;
    nPointsInTrack = 0;
    firstPoint = movingAverage - 1;
    hingePoint = movingAverage / 2;
    double degreesTurned = 0.0;
    double x, y, z, tmp;
    vector<PotentialBug> OneBug;
    OneBug = tracks[j];
    if (OneBug.size() <= firstPoint) return 0.0;
    for (int j1 = firstPoint; j1 < OneBug.size() - 1; j1++) {
        x = sqrt(sqr(OneBug[j1 - hingePoint].GetLocation().x - OneBug[j1 - firstPoint].GetLocation().x) +
sqr(OneBug[j1 - hingePoint].GetLocation().y - OneBug[j1 - firstPoint].GetLocation().y));
        y = sqrt(sqr(OneBug[j1 - hingePoint].GetLocation().x - OneBug[j1].GetLocation().x) + sqr(OneBug[j1 -
hingePoint].GetLocation().y - OneBug[j1].GetLocation().y));
        z = sqrt(sqr(OneBug[j1].GetLocation().x - OneBug[j1 - firstPoint].GetLocation().x) +
sqr(OneBug[j1].GetLocation().y - OneBug[j1 - firstPoint].GetLocation().y));
        //acos domain is -1 to 1
        nPointsInTrack++;
        if (z == (x + y)) //There was no turn
            continue;
        if (z == abs(x - y)) {//180 degree turn
            degreesTurned += 180 * (x / ((OneBug[j1 - hingePoint].GetLastFrame() - OneBug[j1 -
firstPoint].GetLastFrame()) +
                    y / (OneBug[j1 - hingePoint].GetLastFrame() - OneBug[j1 - firstPoint].GetLastFrame())));
            continue;
        }
        tmp = (sqr(x) + sqr(y) - sqr(z)) / (2 * x * y);
        if (tmp <= 1.0 && tmp >= -1.0)
            degreesTurned += (180 - (acos(tmp) * 180 / 3.14159)) * (x / ((OneBug[j1 -
hingePoint].GetLastFrame() - OneBug[j1 - firstPoint].GetLastFrame()) +
                y / (OneBug[j1 - hingePoint].GetLastFrame() - OneBug[j1 - firstPoint].GetLastFrame())));
    }
    return degreesTurned / nPointsInTrack;
}

double tracker::getTrackAverageDegreesTurned(unsigned int j, int mAvr) {
    vector<PotentialBug> OneBug;
    OneBug = tracks[j];
    if (tracks[j].size() > 1)
        //return getTrackDegreesTurned(j, mAvr) / (OneBug[OneBug.size() - 1].GetLastFrame() -
OneBug[0].GetLastFrame());
        return getTrackDegreesTurned(j, mAvr);
    else
        return 0;
```

```
}

unsigned int tracker::findNearestTrackToPoint(Point2f p) {
   vector<PotentialBug> OneBug;
   double MinDis = 10000000;
   double distance;
   unsigned int trackIndex;
   for (int j = 0; j < tracks.size(); j++) {
      OneBug = tracks[j];
      for (int k = 0; k < OneBug.size() - 1; k++) {
         distance = sqr(OneBug[k].GetLocation().x - p.x) + sqr(OneBug[k].GetLocation().y - p.y);
         if (distance < MinDis) {
            trackIndex = j;
            MinDis = distance;
         }
      }
   }
   return trackIndex;
}

double tracker::getPointDegreesTurned(unsigned int j, unsigned int j1, int mAvr) {
   int firstPoint, hingePoint, movingAverage = mAvr;
   firstPoint = movingAverage - 1;
   hingePoint = movingAverage / 2;
   double degreesTurned;
   double x, y, z, tmp;
   vector<PotentialBug> OneBug;
   OneBug = tracks[j];
   if (j1 < firstPoint) return 0.0;
   if (OneBug.size() <= firstPoint) return 0.0;
   x = sqrt(sqr(OneBug[j1 - hingePoint].GetLocation().x - OneBug[j1 - firstPoint].GetLocation().x) +
sqr(OneBug[j1 - hingePoint].GetLocation().y - OneBug[j1 - firstPoint].GetLocation().y));
   y = sqrt(sqr(OneBug[j1 - hingePoint].GetLocation().x - OneBug[j1].GetLocation().x) + sqr(OneBug[j1 -
hingePoint].GetLocation().y - OneBug[j1].GetLocation().y));
   z = sqrt(sqr(OneBug[j1].GetLocation().x - OneBug[j1 - firstPoint].GetLocation().x) +
sqr(OneBug[j1].GetLocation().y - OneBug[j1 - firstPoint].GetLocation().y));
   //acos domain is -1 to 1
   if (abs(z - (x + y)) < 0.0001) //There was no turn
      return 0.0;
   ;
   if (z == abs(x - y)) //180 degree turn
      return 180 * (x / (OneBug[j1 - hingePoint].GetLastFrame() - OneBug[j1 - firstPoint].GetLastFrame())
+
         y / (OneBug[j1 - hingePoint].GetLastFrame() - OneBug[j1 - firstPoint].GetLastFrame()));
   tmp = (sqr(x) + sqr(y) - sqr(z)) / (2 * x * y);
   if (tmp <= 1.0 && tmp >= -1.0)
      degreesTurned = 180 - (acos(tmp) * 180 / 3.14159);
   return degreesTurned * (x / (OneBug[j1 - hingePoint].GetLastFrame() - OneBug[j1 -
firstPoint].GetLastFrame()) +
           y / (OneBug[j1 - hingePoint].GetLastFrame() - OneBug[j1 - firstPoint].GetLastFrame()));
}

double tracker::getPointSignedDegreesTurned(unsigned int j, unsigned int j1, int mAvr) {
   int firstPoint, hingePoint, movingAverage = mAvr;
   double angle1, angle2, angle;
   firstPoint = movingAverage - 1;
   hingePoint = movingAverage / 2;
   double degreesTurned;
   double x, y, z, tmp;
   vector<PotentialBug> OneBug;
   OneBug = tracks[j];
   //*distance = sqrt((sum1.x - sum0.x) * (sum1.x - sum0.x) + (sum1.y - sum0.y) * (sum1.y - sum0.y));
   if (j1 < firstPoint) return 0.0;
   if (OneBug.size() <= firstPoint) return 0.0;
   angle1 = atan2((OneBug[j1 - hingePoint].GetLocation().y - OneBug[j1 - firstPoint].GetLocation().y),
(OneBug[j1 - hingePoint].GetLocation().x - OneBug[j1 - firstPoint].GetLocation().x)) *180 / 3.14159;
   angle2 = atan2((OneBug[j1].GetLocation().y - OneBug[j1 - hingePoint].GetLocation().y),
(OneBug[j1].GetLocation().x - OneBug[j1 - hingePoint].GetLocation().x)) *180 / 3.14159;
   x = sqrt(sqr(OneBug[j1 - hingePoint].GetLocation().x - OneBug[j1 - firstPoint].GetLocation().x) +
sqr(OneBug[j1 - hingePoint].GetLocation().y - OneBug[j1 - firstPoint].GetLocation().y));
   y = sqrt(sqr(OneBug[j1 - hingePoint].GetLocation().x - OneBug[j1].GetLocation().x) + sqr(OneBug[j1 -
hingePoint].GetLocation().y - OneBug[j1].GetLocation().y));
   angle = angle2 - angle1;
   if (abs(angle) < 0.000001)
      return 0;
   while (angle < -180)
      angle += 360;
   while (angle > 180)
      angle -= 360;
   return angle;
}
```