Ben Larrison – Additional Pylons

Code tour of BotMovement.cc

```
std::cout << mapinfo.width;
for (int i = 0; i < 144; ++i) {
    for (int j = 0; j < 160; ++j) {

        std::cout << general_IM[i][j];
    }
    std::cout << std::endl;
}
```

Path function:

First we check to see if the path entered is negative. If it is, it is labled do_not_enter. If the point entered is negative, we adjust the path and call the Path function again recursively with the updated point.

```
//If the point is do_not_enter.
if(general_IM[rand.x][rand.y] == do_not_enter){
    end_pos.x = end_pos.x - 1; //scan left until we find something that doesn't have do_not_enter
    std::cout << "Avoided Negative influence" << "\n\n\n\n\n\n" << std::endl;
    Path(start_pos, end_pos);
    }
```

Next we scan around the area to see if there are low, medium, or high areas of influence. We define a variable check_box, which is the radius around the path that we will check. We check farther out for high influence than medium and low influence. Here is an example of the loop for medium influence:

```
//Scan area around path -- scans a 3x3 square for medium medium
check_box = 3;
for(int i = 0; i < check_box; ++i){
    for(int j = 0; j< check_box;++j){
        if(general_IM[rand.x + i][rand.y + j] == medium_influence && general_IM[rand.x][rand.y] != medium_influence){
            std::cout << "Adjusted path to medium influence\n\n\n\n\n\n" << std::endl;
            end_pos_2.x = rand.x + i; //location of point
            end_pos_2.y = rand.y + j; //location of point
            //Path(start_pos, end_pos); //new end point

        }

    }

}
```

Finally, at the end of Path, we call MoveTo to move to the new adjusted points.

```
MoveTo(start_pos, end_pos_2); //Go to the new point
```

MoveTo

In this function, we call the S2API's unit movement based on a given point. Debug fns are commented out because one day I would like to implement drawing on the game to show it working.

```
virtual void MoveTo(sc2::Point3D start_pos, sc2::Point3D end_pos) final {
    sc2::Units units = Observation()->GetUnits(sc2::Unit::Alliance::Self);
    for (auto& it_unit : units) {

        //sc2::Point2D target = sc2::FindRandomLocation(Observation()->GetGameInfo()); //get random point



        sc2::GameInfo mapinfo = Observation()->GetGameInfo();
        //height: 160
        //width: 144
        sc2::Rect2D rect_test;
        rect_test.from = start_pos;
        rect_test.to = start_pos;

        Actions()->UnitCommand(it_unit, sc2::ABILITY_ID::SMART, end_pos);




        /*DEBUG FNS FOR DRAWING LINE ON MAP
         //const sc2::Unit **unit_pos_x = *it_unit;
         //std::cout << unit_pos_x << std::endl;
         //Debug()->sc2::DebugInterface::DebugLineOut(start_pos, end_pos);
         //const sc2::ObservationInterface* obs = FooBot->Observation();

         //This should be it but...
         //sc2::DebugInterface* debug;  //works
         //debug->DebugIgnoreMineral(); //this automatically closes the game right when it opens...
         //debug->DebugLineOut(start_pos, end_pos); //this is what we want...
         //debug->SendDebug();

         */
    }
};
```

Generate Example Influence map:

Here I generate an example influence map with all the available variables.

```
for (int i = 40; i < 70; ++i) {
    for (int j = 45; j < 75; ++j) {
        general_IM[i][j] = do_not_enter;
    }

}

for (int i = 70; i < 80; ++i) {
    for (int j = 45; j < 65; ++j) {
        general_IM[i][j] = low_influence;
    }

}

for (int i = 80; i < 90; ++i) {
    for (int j = 45; j < 65; ++j) {
        general_IM[i][j] = medium_influence;
    }

}

for (int i = 30; i < 40; ++i) {
    for (int j = 40; j < 50; ++j) {
        general_IM[i][j] = high_influence;
    }

}
```

Here is where my path planning function is actually called. Other bots can use our code in this manner to enhance their path planning. For now, we send it to random locations to exhibit its functionality.

```
sc2::Point2D rand = sc2::FindRandomLocation(Observation()->GetGameInfo());
sc2::Point3D target;
target.x = rand.x;
target.y = rand.y;

sc2::Point3D start;
target.x = 100;
target.y = 100;



Path(start, target);
std::cout << "loop"<<std::endl;
arrived = 1;
```