**Assignment #3 Report**

CPSC 481 - Artificial Intelligence

**A) Team Members: OwlHackers**

|  |  |  |  |
| --- | --- | --- | --- |
| **Names** | **Email** | **Percentage** | |
| Rocio Salguero | [salgueroroci@csu.fullerton.edu](mailto:salguero.roci@csu.fullerton.edu) | 23% | |
| Thao Nguyen | [tnguyen29@csu.fullerton.edu](mailto:tnguyen29@csu.fullerton.edu) | 21% | |
| Sharayu Shetty | [sharayu333@csu.fullerton.edu](mailto:sharayu333@csu.fullerton.edu) | [14%](mailto:sharayu333@csu.fullerton.edu) | |
| Long Nguyen | [long261vn@csu.fullerton.edu](mailto:long261vn@csu.fullerton.edu) | 21% | |
| Sanika Deshpande | sanikadesh198@csu.fullerton.edu | 21% | |

**Project Information:**

**Project Name:** turtlecatch

**File Name:** hw3.cpp, CMakeLists.txt, package.xml

**Launch File:** hw3.launch

**Other File:** readme.txt

**B) Strategy (or heuristic function):**

**/\*------Movement Functions-------\*/**

A function what will keep moving the turtle till it catches all target turtles and uses the heuristic functions

Checks what target turtle (TX) is closest to mainturtle.

f(x) to find the heuristic value of next move - use f\_func(x,y)

must test if failed - use check\_fail(x,y) and system("rosservice call /kill mainTurtle")

If captured a target turtle - use check\_target() and killtargetturtle(turtle index)

To move to location - use move(speed, distance) and relative\_rotate(x,y)

**/\*-----Heuristic Functions------\*/**

double f\_func()

return g\_func() + h\_func(location); // f(n) = g(n) + h(n)

double g\_func()

return distance traveled;

double h\_func(location)

if( out of bounds or near villain turtle ){

return MAX\_VAL;

Else

return distance between main turtle and closest target turtle.

**/\*-----Evaluate function------\*/**

Test all location nodes near using f\_func() then return the location with lowest heuristic value

Using a **Steepest-ascent/descent Hill-climbing** to evaluate all nodes close to mainturtle.

Issues: **Local Maxima/Min**

**C) Pseudocode:**

**// Calculates the heuristic value of the next location**

**// Gets the location and the target turtle to go to**

double f\_func(x, y , turtletarget) {

return g\_func() + h\_func(x, y, turtletarget);

}

**// Returns the distance traveled**

double g\_func() {

return turtleDist;

}

**// Returns MAX\_VAL if near villain or out of bounds, else returns the distance to   
// nearest target turtle**

double h\_func(x, y, turtletarget){

if (out\_of\_bounds() or near\_villain() )

return MAX\_VAL;

else

return distFunc(nextX, nextY, targets[turtletarget].x, targets[turtletarget].y);

}

**// Main function**

main() {

// Spawn target turtles

// Spawn villain turtles

// Calls moveAI() function to start moving the mainturtle

}

**// Calculates the closest target turtle**

int closest\_target() {

closest = -1;

double dist = 999, double current;

for i = 0 to target\_turtle\_number

current = distance (mainturtle.x, mainturtle.y , turtle[i].x , turtle[i].y)

if( current < dist)

dist = current;

closest = turtle[i];

return closest;

}

**// Moves mainturtle until all 3 target turtles are captured or mainturtle is   
// captured by Villain**

void moveAI() {

nextMove;

turtle = closest\_target();

while(true) {

turtle = closest\_target();

//evaluate min of near by

nextMove = evaluate(turtle);

//Move to that location

relative\_rotate(nextMove.nextx, nextMove.nexty);

move(0.5,moveDist);

// Update distance traveled

distanceTraveled += 0.5 \* (moveDist/0.5); // distance += speed \* time

//check if target captured

if(all targets captured) break;

//check if failed

else if( check\_fail() ) {

ROS\_INFO("Main turtle killed. Failed!");

system("rosservice call /kill mainTurtle");

return;

}

}

// Output that the target turtles are captured

// Output the total distance traveled

}

**// Evaluates the nearby nodes and returns the location with the least f\_func()   
// (or heuristic) value.**

coord evaluate(int currtarget) {

//get the minimum f\_func() from the point near mainturtle

for location = 0 to 2\*PI

current = f\_func(location.x, location.y) ;

if(current < minVal) {

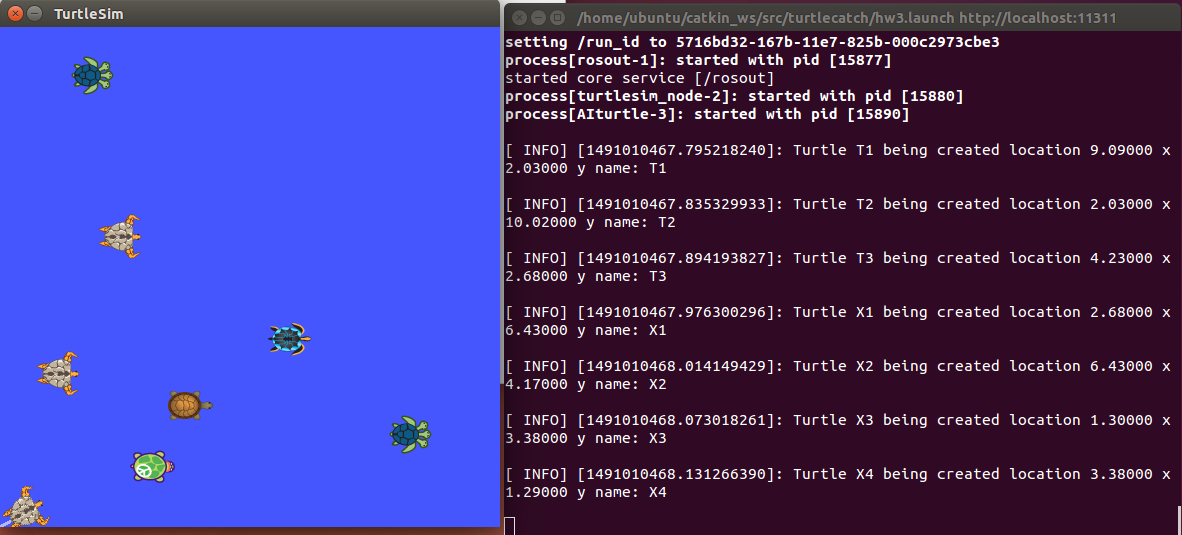
minVal = current; minLocation = location;

return minLocation;

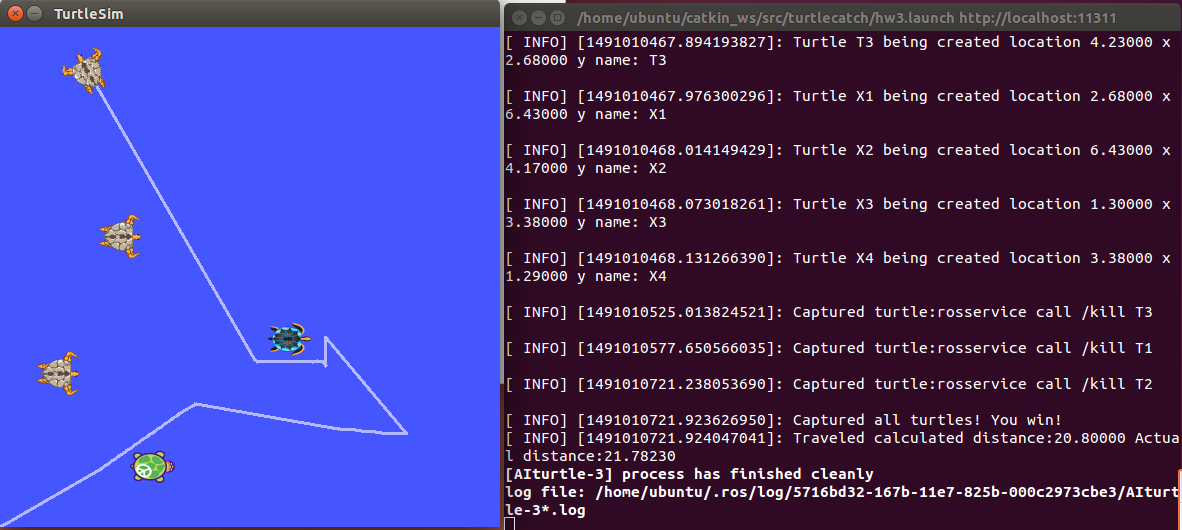
}

**D) Screenshot of A Successful Run:**

* Start:

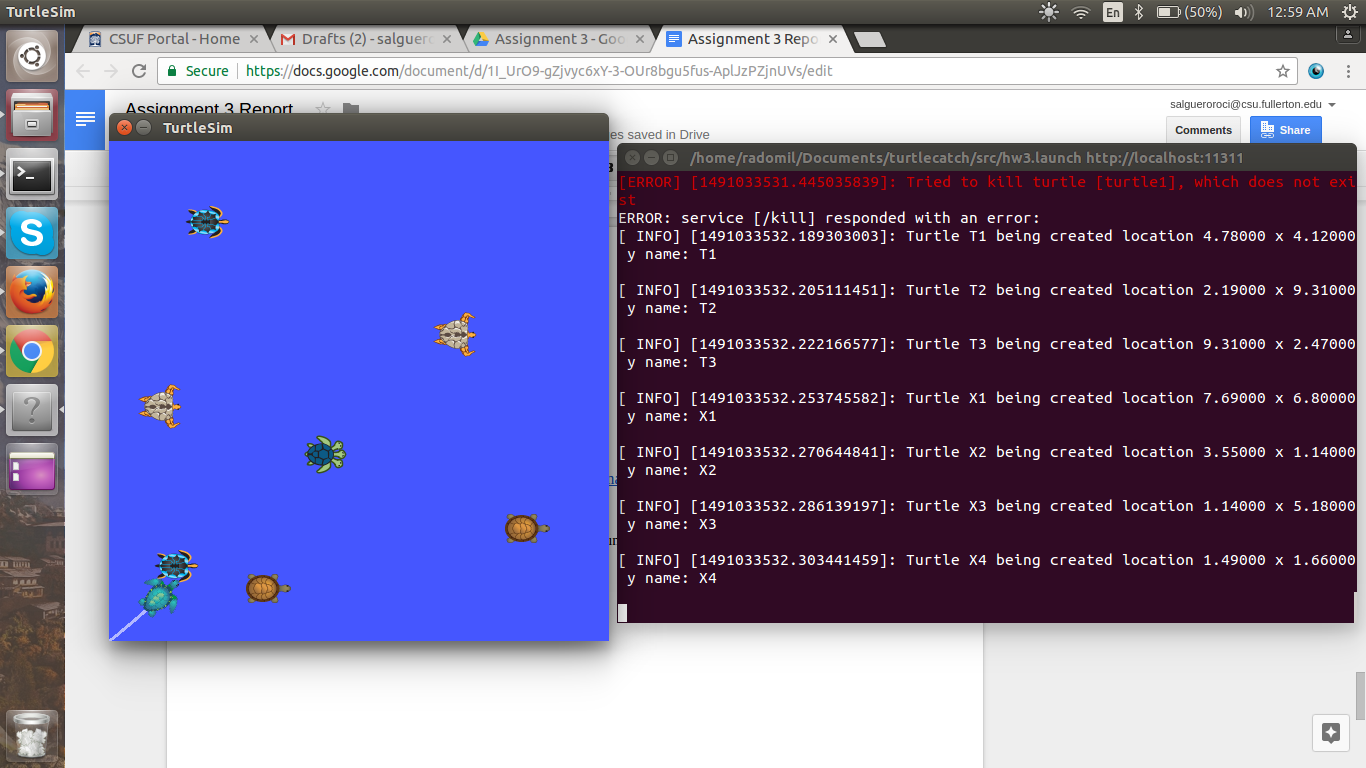


* End:

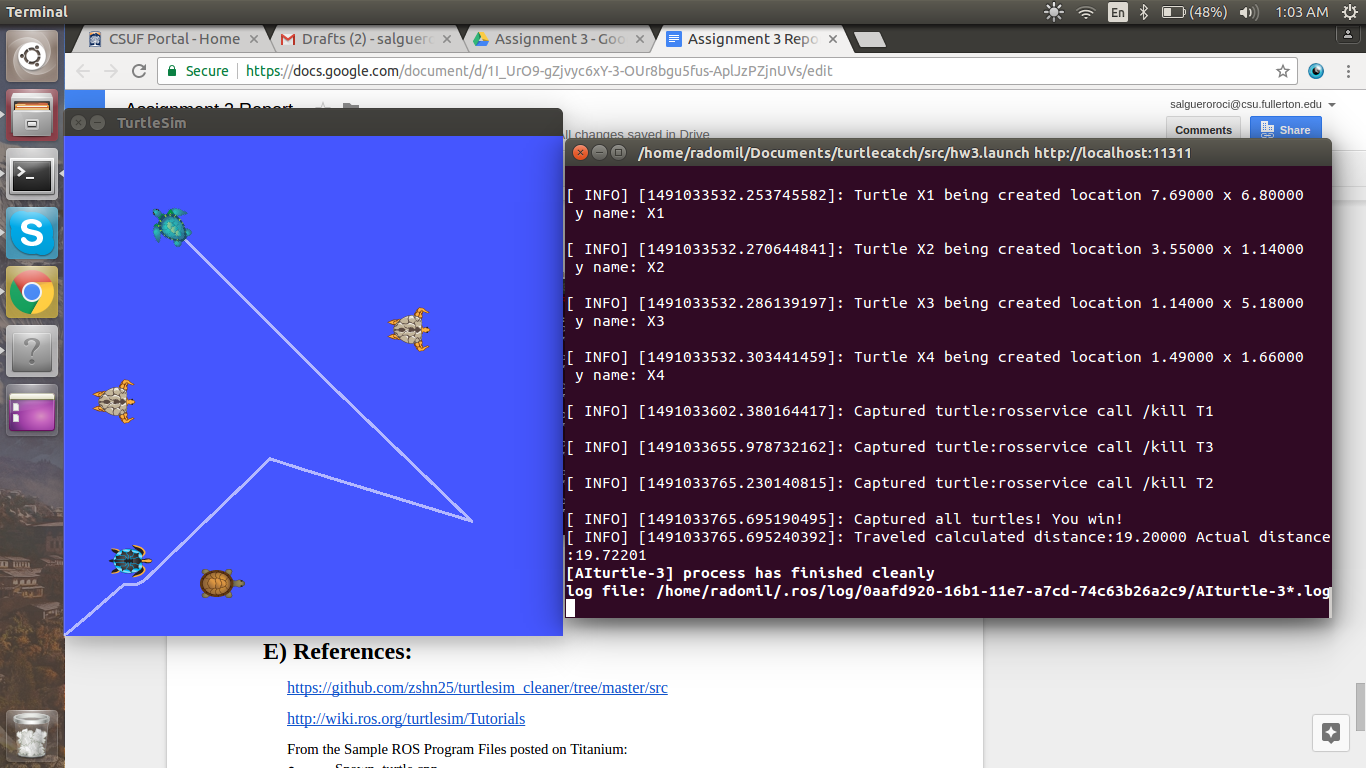


**Another run:**

**Start:**

****

**Finished:**

****

**E) References:**

<https://github.com/zshn25/turtlesim_cleaner/tree/master/src>

<http://wiki.ros.org/turtlesim/Tutorials>

From the Sample ROS Program Files posted on Titanium:

* Spawn\_turtle.cpp
* Robot\_drive\_square.cpp
* Robot\_cleaner.cpp