CS 123
Computational Lab III
Spring 2008
Department of Computer Science
Drexel University

Copyright by the authors. Permission is given for students and staff of CS 123 Spring 2008 to reproduce these notes for their own use.
<table>
<thead>
<tr>
<th>Activity</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab 1</td>
<td>Week 2</td>
</tr>
<tr>
<td>Quiz 1</td>
<td>Week 3</td>
</tr>
<tr>
<td>Lab 2</td>
<td>Week 4</td>
</tr>
<tr>
<td>Quiz 2</td>
<td>Week 5</td>
</tr>
<tr>
<td>Lab 3</td>
<td>Week 6</td>
</tr>
<tr>
<td>Quiz 3</td>
<td>Week 7</td>
</tr>
<tr>
<td>Lab 4</td>
<td>Week 8</td>
</tr>
<tr>
<td>Quiz 4</td>
<td>Week 9</td>
</tr>
<tr>
<td>Lab exam (in class)</td>
<td>Week 10</td>
</tr>
</tbody>
</table>
Proficiency Exam ON WEEK 10

The Exam policy is exactly like the previous term:

- We will have about 20 questions of different labs and different difficulty levels and about 90 minutes

- We will have how did I do button

- The exam will be during the last week of classes, week 10, IN UC 145, YOUR NORMAL CLASS AND YOU SHOULD ATTEND YOUR OWN SECTION

- In case of MAKE UP you should contact your instructor before the exams, only in certain cases we will accept make ups
Important Note about Today’s Lab

1) You are required to work in a group
2) You don't get credit for the lab until you show a TA or instructor your results from the end of the lab. You can do this at the end of the lab period, or by dropping into the Cyber Learning Center next week. If you do not get "checked in" by a staff member, then you will receive no credit for this lab even if you attended a Lab session.
So, What is today’s Lab about?

- Learn what an Application Programming Interface (API) is.
- Use an API to control movements in a simulator for autonomous driving.
- Practice Maple programming involving if-then-elses, for loops, and while loops to handle autonomous driving situations.
- Load CarSimulator package explicitly
The CarSimulator API

- CarSimulator package
- At the beginning of each session, or after a restart
  read CarSimulator.....m;
  with(plottools): with(plots):
  with(CarSimulator):
  libname := “.”, libname:

  Package contains a number of procedures and variables to control a car.
- This “boilerplate” is presented at the beginning of the worksheet. It is always the same.
A simulator for autonomous driving

After you load the simulator package, here’s how you use it:

1. initialize();
   do setBackGround steps to establish walls, targets
   set initial position of car
   The commands are usually presented as “hit enter once to do it all”.

2. programName := proc ..... end;  #enter the control program for the car
   This is the part you have to supply.

3. run(programName);  # You do this after you define the procedure.

4. carMovie(stateTable);  # You do this after you run the procedure
Protections built into the simulator

- There is a variable maxMoves initially set to 40. The simulator will automatically stop if you try to run a simulation beyond 40 moves.
- You can get complaints about exceeding 40 moves if you forget to initialize(); again before you re-run a program.
- You can assign maxMoves to a larger value if you need more than 40 moves to get the work done.
Some key operations in the CarSimulator package

- move(1); #moves the car forward one step
- turn(angle); turns the car (in place, without moving forward) from present angle
  - turn(Pi/2); #turn left 90 degrees
  - turn(Pi); #turn around
  - turn(-Pi/2) #turn right 90 degrees
stateTable

- stateTable – after you run a simulation, this table contains a list of all the moves of the car. stateTable[0] is a list describing the first position of the car:
  - [x coord, y coord, angle, state]

  [0, 0, \(\pi/2\), CARNORMAL] means car is at (0,0) facing northwards in a normal state

- [5, 4, 0, CARBUMPED] means car is at (5,4) facing eastwards, and has just bumped into a wall

- [10,2,\(\pi\), CARFINAL] means the car is at (10,2) facing westwards, and has reached a TARGET square (colored blue).

- stateTable[1] shows the state of the car after one move, stateTable[2] after two moves, etc.
carMovie(stateTable);

- The simulation sets up stateTable for you – you never have to do any programming to change it yourself.
- carMovie(stateTable) creates an animation from the stateTable.
Let’s look at one example

- we want you to write a program that will cause the car to move forward 3 steps [Problem 1]
initialize();
setBackground([[0,0],[1,0],[2,0],[3,0]], MARKING): #more initialization -- mark the path to be travelled so that the
  #student can see that they've covered the expected squares.
establishBoundaries(20,20): #draw a 20 x 20 sandbox for the car to roam around in. Coordinates
  #range from [0,0] to [20,20].
drawBackground(): #Create a plot of the wall and the target for later use, but don't display it yet
carMovie(stateTable);  #Show initial configuration

initialize()
carMovie(stateTable)

We are now going to play a pre-recorded movie showing you what's supposed to happen -- the car moves forward three squares.
read "runSimpleMoveMovie.m";
runSimpleMoveMovie; #Click on the movie and play it.
So how can we code the Procedure to make that animation?

runSimpleMove := proc()
    #Put your programming here.
    move(1);
    move(1);
    move(1);
    move(1);
end;

→ All we need to do is to move 3 times ,…Mmmm, how about calling move(1)? [check the help references]
Now let’s TURN!

run2a := proc();
    # move east 3 times
    move(1);
    move(1);
    move(1);
    turn(Pi/2); # turn north
    move(1); # move north four times
    move(1);
    move(1);
    move(1);
    turn(-Pi); # turn around
    move(1);
    move(1);
    turn(Pi/2); # face to the east/right again
    move(1); # The car turns gold because it facing and touching a target.
end:
How about automating the moves? -- Loops

Question: Simulate this animation

Key idea:

-- WRITE SOME LOOPS!

- for i from 1 to 5 do
  - move(1);
- end do;
- turn(Pi/2);
TurnAbsolute and Nested Loops

Some more cases:

Question: Animate this configuration →

Key Idea:

1) For complete turn:
   \[
   \text{turnAbsolute}(\pi); \text{ #face west}
   \]

2) Use nested loops:
   \[
   \text{for …. Move(1); turn}(\pi/2);
   \]
Using parameters – a steps right, b steps up!

Here's a demo program that moves forward \( a \) steps and then turns left and moves forward another \( b \) steps.

> run3a := proc( a::posint, b::posint) #move a steps forward, make a 90 degree turn to the left, and then b steps forward.
>     local i;
>     for i from 1 to a do move(1); end do;
>     turn(Pi/2);
>     for i from 1 to b do move(1); end do;
> end:
> run(run3a,4,5); #move 4 steps forward, turn, then move 5 steps forward.
> carMovie(stateTable);
We have a Procedure:

```
isTouching(direction, 'barrierType')
```

Checks if we reached a barrier or not → return value:

- True == touching,
- False == not touching!

Note: “barrierType” is : BARRIER or TARGET
Q:::Assume you want to write a procedure that given this configuration the car moves once and come back between the two walls (barriers)

Idea:::::::::::

while (getCURRENTLOCATION()[4] <> CARBUMPED) do
    move(1);
    end do;
    turn(PI);
Finding a gap and going through it

You can use a while loop to test for whether the car is touching the wall.

- If it is, then move forward 1.
- Otherwise, leave the while loop and proceed into the actions that turn the car and move it forward to the target.

```plaintext
runc5b := proc()
   while ....
   #Now turn right

   #now move until you hit the target
   while ...
   end:
```

#Now turn right

#now move until you hit the target

#end:
Detecting things at a distance

The car in the simulator can also use "radar" to look to the front and to the side to see if there's something within a certain range.

>scanDistance == gives us info about space

>example:
>if scanDistance(cl[1],cl[2],angleRad, d) <> NULL then

get report:

>printScanDistanceReport(d, 0);
General advice

- To create a car control program, look at the situation and the demo if provided.
- Have your group create a plan. You can “pseudo code” the plan into the program as comments:
  ```
  run3a := proc(a::posint)
      #Move forward 1 step
      #Turn left
      # Move forward a steps
  end;
  ```
- Then fill in the comments with actual control statements. The important thing is to have a plan first!
More general advice

- Put print or printf statements into your program to see what it is doing

- trace(run3a);
  trace(isTouching); etc.
  will also print out information. To turn off tracing do untrace(run3a);
Some technical notes about the lab

Because the lab has a lot of animations, the file is very large. This means autosaving will take a while. So to avoid problems it may be helpful to increase your autosave interval.

Also, to make the file smaller, it will significantly speed up your save time if you delete any tutorial animations after you view them. You can always “regenerate” them later if you need them by hitting enter on the command that created the animation.

Finally, the help files may have trouble loading on the Macs in the lab. If that is the case, there are PDF versions of the help files in the zip folder with the lab. Please refer to those if the help files don’t load properly on the Macs.
Lab structure

- This lab is divided into the following sections:
  - **Required Reading**: Introduction & Overview, Tutorials 1-5, Wrap-Up
  - **Required Problems**: Problems 1-5
  - **Optional Problems**: Problems 6,7

  *Work in groups as always !!*
  *Do not forget to CHECK IN!*
Maple talk, Thursday May 29

- Location: Bossone Auditorium
- Time: 10am-11am
- Date: Thursday May 29
- Speaker: Robert Lopez, Professor Emeritus, Rose-Hulman Institute
  “Clickable Math in Maple, with Engineering Overtones”
Why you should go to this talk

- To learn about Maple’s use in more advanced engineering work.
- Because you will get credit equivalent to half a quiz for attending the (entire) lecture.
- Bring your Drexel ID – attendance will be taken.
Talk abstract

- Clickable Math in Maple, with Engineering Overtones
- The ease-of-use features in Maple coalesce into a syntax-free paradigm for implementing calculations via a point-and-click approach. The robustness of this paradigm will be illustrated by examples taken from the math courses taught to science, engineering, and math students in the college undergraduate curriculum. Several distinctly engineering examples will also be included to show the utility of Maple as a tool outside the math classroom.
Can’t make the lecture?

- The lecture will be video taped. There will be a replay of the lecture:
  - Bossone Auditorium, Friday May 30, 10:30am.
- Send email to your instructor if you wish to attend the alternative time.
What to turn in?!

- Lab 4, Quiz 4
- **DO NOT FORGET PROFICIENCY EXAM ON WEEK 10!**
- Submit your lab via Bb Vista
- We have office hours in UC147 next week, if you have any problems on Quiz 4 or the lab feel free to stop by and ask questions!
- Good Luck!