Dangerous Drivers

New Mexico

Supercomputing Challenge

April 4, 2007

Team 22

Bosque School

Team members

Calin Popa

Reed Sanchez

Jeremy Adkins

Teacher

Thomas Allen

Mentor

Jim Hicks

Dangerous Drivers

Our project for the New Mexico Supercomputing Challenge involved making a traffic simulator to represent crashes at an intersection. Our goal was to answer the question: "What types of driving behaviors – such as shifting, braking, turning and running red lights – cause accidents?" In order to do this, we used StarLogo agent-based modeling. We created a cross-street intersection with a stoplight in the middle that gives cars the signal to go, stop, and turn. The number of "cars" – or turtles in the case of our program – in the intersection can vary from 2 to 99. They are programmed to shift and make turns, which greatly increases the chances of a crash. The program uses "checks" to determine if the turtles are likely to "crash" or not. These checks work by assigning a random number for each turtle that determines if the turtle "drives" safely or not – in other words, how aware the turtle is of its surroundings and if it checks for other turtle before it shifts lanes or makes turns. When two turtles "crash," or collide, a blue square appears meaning a crash has occurred.

The first step we took of developing our model to answer the question was to gather information on traffic accidents. For instance, "in 2002 43,005 people died due to traffic accidents in America." and "in 2004 42,836 people died due to traffic accidents in America."(MADD). "The injury rate per 100 million vehicle miles of travel decreased by 6.0 percent from 2003 to 2004 according to Reference: 2004 Annual Report and Fatality Analysis Reporting System" (FARS). After gathering the data, we started to program an interaction simulation with realistic driving actions. Our programmer, Reed Sanchez, finished the model, we started to look for patterns in the crashes. We ran the program 6 times for approximately 5 minutes each time and recorded the data. The program would

freeze when the number of turtles crashed (stacked up) in the intersection got too big. In order to get better statistics, we need to run the program more times, yet we ran out of time. The data presented below was gathered from our set of 6 runs.

Our data showed that crashes resulting from right turns and t-bone crashes, resulting from running a red light, are uncommon. However, crashes resulting from shifting lanes and braking are very common. We all believed that left turns would cause the most crashes, yet we were mistaken. At this time, our model doesn't take the severity of the crash into account. "Rear ending" a car due to braking is not likely to be as bad a crash a T-bone crash resulting from running a red light. From our model, we conclude that most crashes occur while shifting lanes or braking. This makes some sense because changing lanes is complicated and requires that the driver look and think about blind spots. Yet this computer model doesn't take the effect the full variability that humans and their reaction time and driving skill would have on the results of this experiment.

We believe that our number one original achievement is making a realistic simulation of a street intersection that has some aspects of realistic driving behavior included – for instance, checking to see if a car is next to you before you shift lanes. Also, we included a factor to make some drivers "worse" than others (the random number check) in order to begin to account for human error and variability. In the future, we want to explore the effect that the age of the driver may have on crashes in the intersection. We want to make the car colors represent the age of the driver, and include another variability factor, to see if the results are affected.

ACKNOWLEDGEMENTS

We used StarLogo for our agent-based modeling, Microsoft Word to generate this report, and Microsoft Powerpoint to create our interim presentation. We used data from the MADD (Mothers Against Drunk Driving) website and FARS (Fatality Analysis Reporting) for the information on crashes. We would like to thank our teacher Mr. Thomas Allen for helping us, Mr. Jim Hicks who served as our mentor, our school Bosque School and the New Mexico Supercomputing Challenge program for the help that they gave us to do this project.

Figure: Crash Results

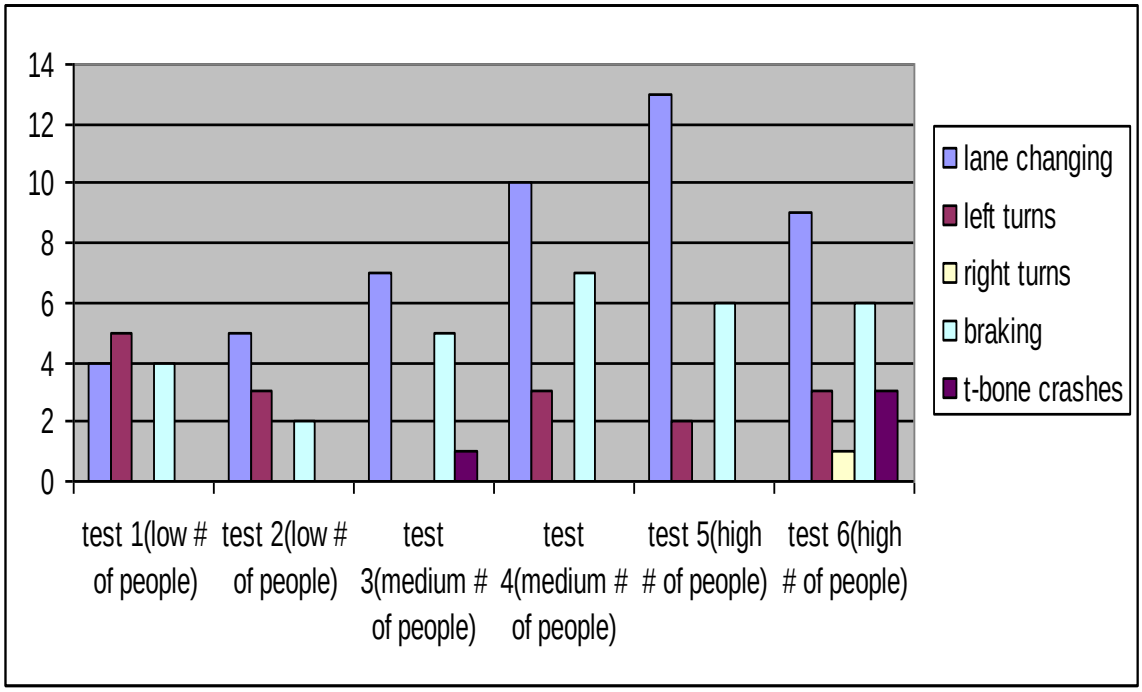
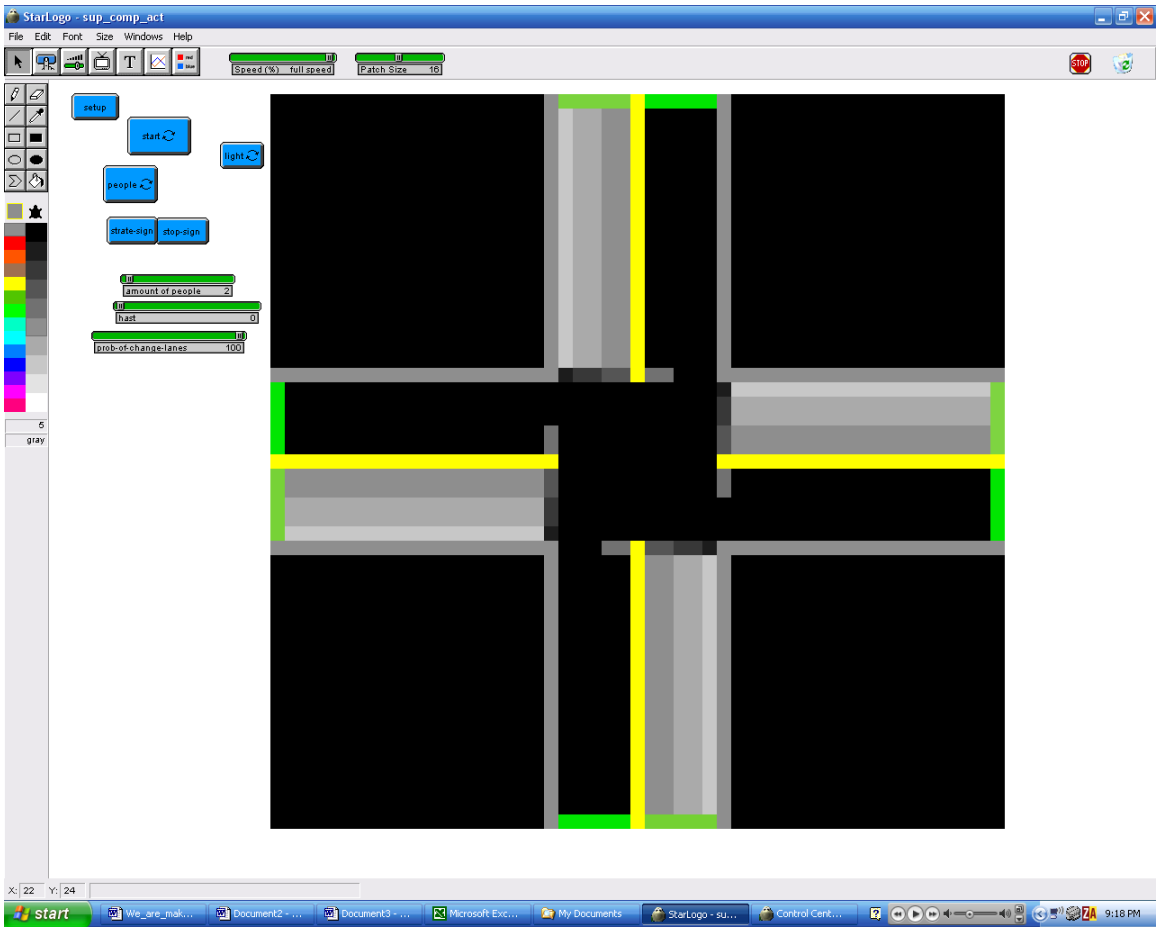


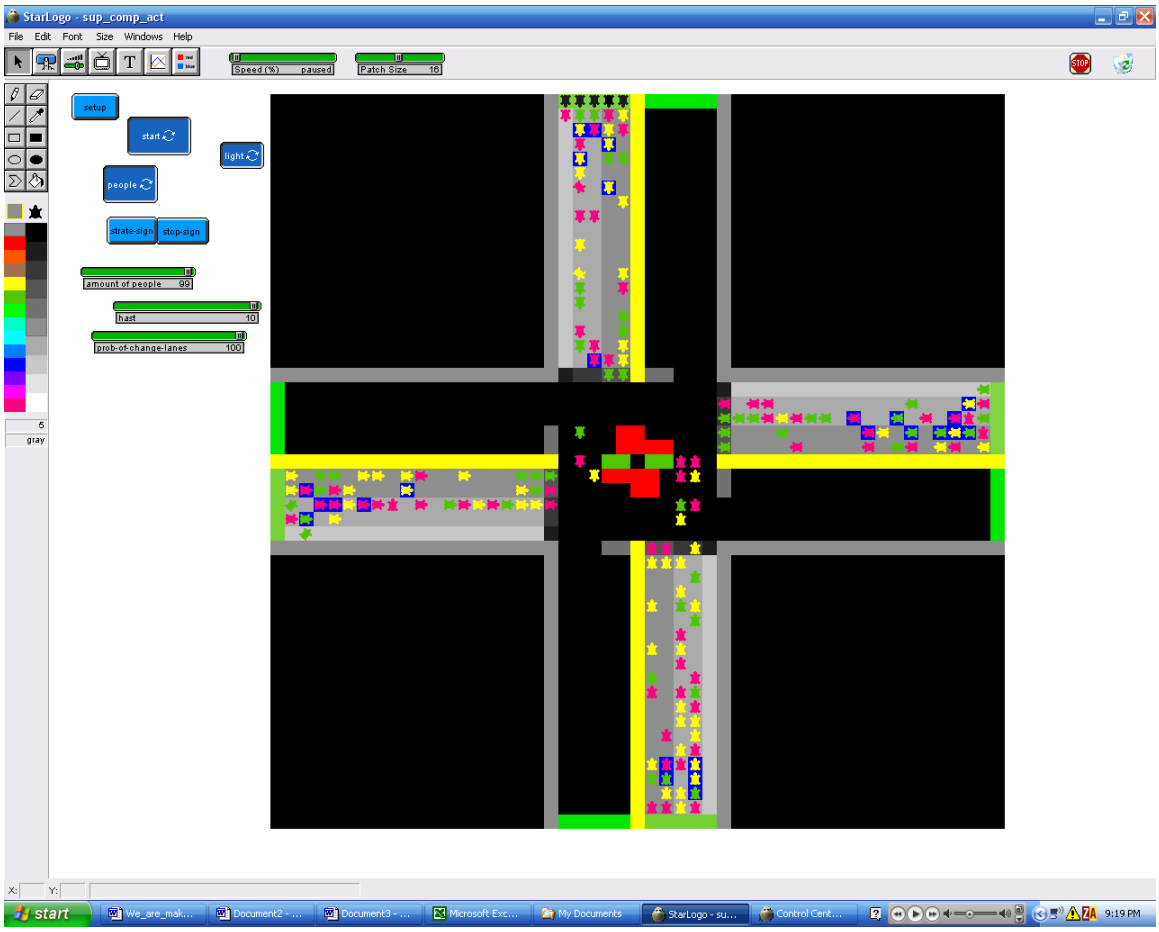
Table: Crash Data

	test 1(low # of people)		test 2(low # of people)
lane changing	4	lane changing	5
left turns	5	left turns	3
right turns	0	right turns	0
braking	4	braking	2
t-bone crashes	0	t-bone crashes	0
	test 3(medium # of people)		test 4(medium # of people)
lane changing	7	lane changing	10
left turns	0	left turns	3
right turns	0	right turns	0
braking	5	braking	7
t-bone crashes	1	t-bone crashes	0
	test 5(high # of people)		test 6(high # of people)
lane changing	13	lane changing	9
left turns	2	left turns	3
right turns	0	right turns	1
braking	6	braking	6
t-bone crashes	0	t-bone crashes	3

Picture of Setup



Picture of Running Program



TURTLE CODE

```
Turtles-own [haste place]

to start
if pc = green + 1 [ifelse ((count-turtles-at 0 1) not= 0) [die] [fd 1]]
if pc = green + 1.1 [ifelse ((count-turtles-at 1 0) not= 0)[die][fd 1]]
if pc = green + 1.2 [ifelse ((count-turtles-at 0 -1) not= 0)[die][fd 1]]
if pc = green + 1.3 [ifelse ((count-turtles-at -1 0) not= 0)[die][fd 1]]
findage
findhast
findwhere
gtwhere
end

to findage
if color = 1 [setc green]
if color = 2 [setc pink]
if color = 3 [setc orange]
if color = 0 [setc yellow]
end

to findhast
if ((Random 1000) > W)
        [sethaste 1]
end

to findwhere
let [:ran (random 2)]
ifelse ((random 100) < prob-of-change-lanes)
        [
        if :ran = 0 [setplace 0]
        if :ran = 1 [setplace 1]
        if :ran = 2 [setplace 2]
        ]
        [
        if pc-ahead = black + 5 [setplace 0]
        if pc-ahead = black + 6 [setplace 1]
        if pc-ahead = black + 7 [setplace 2]
        ]

end

to gtwhere
loop
        [
```

```
if place = 0
        [
        if pc-ahead = 7 [cll]
        if pc-ahead = 6 [cll]
        if pc-ahead = 5 [watchlights if pc = 5 [fd 1]]
        if pc-ahead = 3 [care fd 1 watchlights]
        ]

if place = 1
        [
        if pc-ahead = 7 [cll]
        if pc-ahead = 5 [clr]
        if pc-ahead = 6 [watchlights if pc = 6 [fd 1]]
        if pc-ahead = 2 [care fd 1 watchlights]
        ]

if place = 2
        [
        if pc-ahead = 6 [clr]
        if pc-ahead = 5 [clr]
        if pc-ahead = 7 [watchlights if pc = 7 [fd 1]]
        if pc-ahead = 1 [care fd 1 watchlights]
        ]
if pc = 0 [care fd 1]
if pc = green + 9 [die]
if pc = 4 [lt 45 fd 1]
if (count-turtles-here not= 1)
        [
        crash 0
        crash 1
        crash 2
        crash 3
        crash 4
        crash 5
        crash 6
        crash 7
        crash yellow
        crash 56
        crash 56.1
        crash 56.2
        crash 56.3
        if color = red [die]
        ]
if color = red [die]
if pc-ahead = blue [rt 90 fd 2 lt 90 fd 2 lt 90 fd 2 rt 90]
]
```

```
end


to straight-all
if pc = green + 9 [die]
if pc = yellow [die]
if (count-turtles-here not= 1) [stamp blue die]
if pc-ahead = blue [rt 90 fd 2 lt 90 fd 2 lt 90 fd 2 rt 90]
if ((Random 1000) > w)
        [
        if pc = green [fd 1]
        if pc = red [fd 1]
        if pc = black [fd 1]
        if pc = black + 2 [fd 1]
        if pc = black + 3 [lt 45 fd 1]
        if pc = black + 4 [lt 45 fd 1]
        ]
end

to crash :colr
if pc = :colr
        [
        stamp blue
        wait 30
        stamp :colr
        grab one-of-turtles-here
        [setc blue setc-of partner red]
        ]
end


to watchlights
care
if pc = black [fd 1]
if pc = black + 2
        [
        waitchangelights 2 -6
        waitchangelights -2 6
        waitchangelights -6 -2
        waitchangelights 6 2
        ]
if pc = black + 3
        [
        waitchangelightl 3 -5
        waitchangelightl -3 5
        waitchangelightl -5 -3
```

```
        waitchangelightl 5 3
        ]
if pc = black + 1 [fd 1 rt 90]
if pc = black + 4 [lt 45 fd 1]
end

To waitchangelights :x :y
ifelse (pc-at :x :y) = red
        [
        wait-until [(pc-at :x :y) = green ] fd 1
        ]
        [
        if (pc-at :x :y) = green [fd 1]
        ]
end

To waitchangelightl :x :y
ifelse (pc-at :x :y) = red
        [
        wait-until [(pc-at :x :y) = green] lt 45 care fd 1
        ]
        [
        if (pc-at :x :y) = green [lt 45 fd 1]
        ]
end

to care
check 180 0 -1
check 90 1 0
check 270 -1 0
check 0 0 1
check 45 1 1
check (270 + 45) -1 1
check (45 + 90)1 -1
check (45 + 180)-1 -1

end

to check :heading :x :y
if heading = (:heading)
        [
        if ((count-turtles-at :x :y) not= 0)
        [wait-until [(count-turtles-at :x :y) = 0]]]
end

to clr
```

```
rt 45
clsr 45 1 0
clsr 315 0 1
clsr 225 -1 0
clsr 135 0 -1
end

to clsr :heading :x :y
if heading = (:heading)
        [ifelse ((count-turtles-at :x :y) not= 0)
        [lt 45 care fd 1 clr] [fd 1 lt 45]]
end

to cll
lt 45
clsl 45 1 0
clsl 315 0 1
clsl 225 -1 0
clsl 135 0 -1
end

to clsl :heading :x :y
if heading = (:heading)
        [ifelse ((count-turtles-at :x :y) not= 0)
        [rt 45 care fd 1 cll] [fd 1 rt 45]]

End
```

OBSERVER CODE

```
to setup
ca
build
end

to build
crt 4
ask-turtles
        [
        fd 6 setc yellow pd fd 19 rt 90 pu fd 6 rt 90
        setc grey pd fd 19 pu fd 1 pd
        ]

ask-turtle 0
```

```
        [
        setc black + 1 fd 1
        setc black + 2 fd 2
        setc black + 3 fd 1 pu fd 2 pd
        setc black + 4 fd 1 pu fd 4 pd
        setc grey fd 19 rt 90
        setc green + 1 pu fd 1 pd
        fd 4 pu fd 2
        setc green + 9 pd fd 4
        ]
ask-turtle 1
        [
        setc black + 1 fd 1
        setc black + 2 fd 2
        setc black + 3 fd 1 pu fd 2 pd
        setc black + 4 fd 1 pu fd 4 pd
        setc grey fd 19 rt 90
        setc green + 1.1 pu fd 1 pd
        fd 4 pu fd 2
        setc green + 9 pd fd 4
        ]
ask-turtle 2
        [
        setc black + 1 fd 1
        setc black + 2 fd 2
        setc black + 3 fd 1 pu fd 2 pd
        setc black + 4 fd 1 pu fd 4 pd
        setc grey fd 19 rt 90
        setc green + 1.2 pu fd 1 pd
        fd 4 pu fd 2
        setc green + 9 pd fd 4
        ]
ask-turtle 3
        [
        setc black + 1 fd 1
        setc black + 2 fd 2
        setc black + 3 fd 1 pu fd 2 pd
        setc black + 4 fd 1 pu fd 4 pd
        setc grey fd 19 rt 90
        setc green + 1.3 pu fd 1 pd
        fd 4 pu fd 2
        setc green + 9 pd fd 4
        ]
ask-turtles
        [
        pu rt 180 fd 6 lt 90 fd 1
```

```
        setc black + 5 pd fd 17 pu rt 90 fd 1 rt 90 pd fd 17 pu
        setc black + 6 lt 90 fd 1 lt 90 pd fd 17 pu rt 90 fd 1 rt 90 pd fd 17 pu
        setc black + 7 lt 90 fd 1 lt 90 pd fd 17
        ]
ct
end

to people
ask-patches-with [pc = green + 1]
        [if ((Random 100) < n)
                [sprout [setc (random 3)]]]
ask-patches-with [pc = green + 1.1]
        [if ((Random 100) < n)
                [sprout [setc (random 3)rt 90]]]
ask-patches-with [pc = green + 1.2]
        [if ((Random 100) < n)
                [sprout [setc (random 3)rt 180]]]
ask-patches-with [pc = green + 1.3]
        [if ((Random 100) < n)
                [sprout [setc (random 3) lt 90 ]]]
end

to light
stop-sign
wait 10
go-vert-strait
wait 30
stop-sign
wait 10
go-hors-strait
wait 30
stop-sign
wait 10
go-vert-left
wait 30
stop-sign
wait 10
go-hors-left
wait 30
end

to go-vert-strait
ask-patch-at 1 0 [setpc green]
ask-patch-at -1 0 [setpc green]
ask-patch-at -2 0 [setpc green]
ask-patch-at 2 0 [setpc green]
```

```
end

to go-hors-strait
ask-patch-at 0 1 [setpc green]
ask-patch-at 0 -1 [setpc green]
ask-patch-at 0 2 [setpc green]
ask-patch-at 0 -2 [setpc green]
end

to go-vert-left
ask-patch-at -1 -1 [setpc green]
ask-patch-at 1 1 [setpc green]
ask-patch-at -2 -1 [setpc green]
ask-patch-at 2 1 [setpc green]
end

to go-hors-left
ask-patch-at -1 1 [setpc green]
ask-patch-at 1 -1 [setpc green]
ask-patch-at -1 2 [setpc green]
ask-patch-at 1 -2 [setpc green]
end

to strate-sign
ask-patch-at -1 0 [setpc green]
ask-patch-at 1 0 [setpc green]
ask-patch-at 0 1 [setpc green]
ask-patch-at 0 -1 [setpc green]
ask-patch-at -2 0 [setpc green]
ask-patch-at 2 0 [setpc green]
ask-patch-at 0 2 [setpc green]
ask-patch-at 0 -2 [setpc green]
ask-patch-at -1 -1 [setpc green]
ask-patch-at 1 1 [setpc green]
ask-patch-at -2 -1 [setpc green]
ask-patch-at 2 1 [setpc green]
ask-patch-at -1 1 [setpc green]
ask-patch-at 1 -1 [setpc green]
ask-patch-at -1 2 [setpc green]
ask-patch-at 1 -2 [setpc green]
end

to stop-sign
ask-patch-at -1 0 [setpc red]
ask-patch-at 1 0 [setpc red]
ask-patch-at 0 1 [setpc red]
```

```
ask-patch-at 0 -1 [setpc red]
ask-patch-at -2 0 [setpc red]
ask-patch-at 2 0 [setpc red]
ask-patch-at 0 2 [setpc red]
ask-patch-at 0 -2 [setpc red]
ask-patch-at -1 1 [setpc red]
ask-patch-at 1 -1 [setpc red]
ask-patch-at -1 2 [setpc red]
ask-patch-at 1 -2 [setpc red]
ask-patch-at -1 -1 [setpc red]
ask-patch-at 1 1 [setpc red]
ask-patch-at -2 -1 [setpc red]
ask-patch-at 2 1 [setpc red]
end
```