Wave Interaction In Pools

New Mexico
Supercomputing Challenge
Final Report
March 31, 2009

Team 103
Los Alamos Middle School

Team Members
Justine Yang
Lauren TenCate

Teacher
Robert Dryja

Project Mentors
Emily TenCate
James TenCate

Motivation: Why should anyone care?

Swimming is fun and healthy, but a lot of people don't participate in swimming because they feel it is harder to do a sport in the water than on land. This is because water, which is 800 times denser than air, acts as a wall. Also, making waves takes up a lot of energy. In fact, if speed is doubled, the energy spent on making waves will increase eightfold. *

Competitive swimming is an international sport, but waves slow even the fastest swimmers down. Perhaps if people could better control these waves that create turbulence making it harder for them to swim, more world records would be broken. Without people getting swamped by their own waves people such as Michael Phelps could go faster and faster because they would know what

kind of waves to expect. Pool directors could use this information by using it to decide what kind of lane-lines to buy to make their patrons have the most enjoyable experience. The general public could use this information to make their swimming experience more fun.

*"If I'm so fit, why is swimming so hard?" in Swimming Made Easy: The total immersion way for any swimmer to achieve fluency, ease, and speed in any stroke by Terry Laughlin, Total Immersion Swimming, Inc., New Paltz NY, 2001.

How we got interested in this project:

Both of us are competitive swimmers and we wanted to know how lane-lines and the waves bouncing off of them affected our speed. We've noticed that when swimming sometimes a really big wave swamps us when we try and travel against it. Other times we can just ride along in somebody's wave, but only if they're going the same direction

as us. This sparked our interest and we decided to do a project on it.

Problem:

Our team is trying to create a computer program that simulates the creation, reflection and damping out of waves that occur in competitive swimming. Our ultimate goal was to see if how waves interact with swimmers and ultimately if it affects the swimmer's speed.

The first task was to simulate waves created by a swimmer moving thru the water in a pool. Our second goal was to put the swimmer in a lane by adding lane lines and making the lane-lines and walls reflect and absorb waves that came off of the swimmer agent. The next consideration was how waves interact with the swimmers. Do waves help make people swim faster or slower? If you are

the fastest swimmer in the pool, will your waves slow down other swimmers or will it help them?

From observing ourselves and our teammates while swimming freestyle at a moderate pace in the pool, we determined that the average wake angle coming off a swimmer is about 45 degrees. Previous studies by one of our team members who did a series of experiments investigating wake angles and wave reflection patterns off different types of lane-lines confirmed this**, and gave us an idea of other elements that would be crucial in developing a realistic model. The most important observation was that waves only partially went through lane-lines; in other words, when a wave crossed through a lane line into the next lane, the wave height got smaller and sometimes the wave was completely damped out.

[**"Breaking Waves: Do Wave-Eating Lane Lines Really Work?" L. TenCate, Los Alamos County Science Fair 2008]

Method:

Neither of us had programmed before. Our team chose to learn and program in NetLogo, an agent-based program that seemed appropriate for this type of simulation. We used a layered approach to put together our project exactly as we had planned (making smaller sub-procedures and then putting them into a larger procedure to avoid confusion.) Our mentors had suggested this approach and we thought it was a good idea so we decided to use it.

Our first order of business was to create the pool. This consisted of a rectangular box with walls (the pool), lane-lines parallel to one set of walls of the pool, and swimmers inside the lanes. We made three

lanes with one swimmer in the middle of each lane. Using the middle swimmer (so we could experiment with various ideas more easily) we simulated the wake by creating little blue turtles coming off of the swimmer as it moved forward. The wake angle varied with the speed of the swimmer.

Next, we experimented with ways to have the wake partially reflect off the lane-lines, simulating what happens with real lane-lines. Of the different approaches, we found one acceptable solution that we used for the rest of the project.

After getting partial reflections incorporated into the program, our main objective was to have each individual wake die off gradually. We made all three swimmers have their own independent wakes, and changed each swimmer's speed so that observing the effect of speed on the wake would be

easier. Our plan was to have the swimmer slow down once a wave hit them. To come up with realistic figures that would make our project accurate we compared it with real-world data. The data was measured from video of actual swimmers, which were taken during swim practices and swim meets at the Larry R. Walkup Aquatic Center in Los Alamos. From our experimental data we obtained essential information that we based our project and assumptions on. For example, our experimental research indicated that the typical wake-angle was about 45 degrees and that depending on the type of lane-line about 50-80 percent of the waves generated go through a lane-line. Then we chose variables that resulted in the above behavior and used those numbers to start our simulations. In our program we built in sliders so these parameters could be easily changed and resulting effects observed.

Details of Method: Description of Sub-Procedures

We chose to program a simple pool, three swimmers, lane-lines, and walls. The lane-line and wall color was chosen to be blue. The color was important for the reflection and bounce of the waves.

The setup of the pool and lane-lines was in the **draw-walls** sub-procedure. The Bounce routine for wave reflection was in the sub-procedure **bounce**. For experimental purposes our team started with a Netlogo example and built off of it to create the version of bounce we finally used.

The simulation starts with swimmers moving up and down the length of the pool in their lanes and turning at (or bounding off of) the walls. The speed

of each swimmer is an independent variable.

Each of the swimmers hatches turtles to represent the wake shedding off of them every step. This is described in the sub-procedure **move**. We used the color of the walls and the lane lines as a variable to signal the wave agents to bounce off of walls and lane lines only, not off the water. When the waves bounce off a boundary, they reflect in the opposite direction 180 degrees from the direction in which they came in. We can watch the development of the wave and its interaction with the lane lines and walls on the screen. Ultimately we will be able to watch the interaction of the waves and the swimmers too.

Occasionally, our program runs into a runtime error. This is probably because a wave turtle hits another wave turtle and we haven't told them what to do if

that happens. We have tried to fix it several different ways, but we haven't yet found a solution.

Original Goals

These were our original goals:

- ✓ Add waves to the all swimmers
- ✓ Make only some wave agents escape the lane
- ✓ Decide realistic rules for how to slow down a swimmer when encountering wave agents
- ✓ Create lane-lines and a pool
- ✓ Make waves realistically die out
- ✓ Have swimmers bounce off the walls

Achievements

Our actual achievements are as follows:

We have a running program with everything we set out to do except for deciding realistic rules to describe the slowing down of a swimmer when a

wave agent is encountered. We have simulated a pool with lane-lines and walls, three swimmers and wave-agents that represent the wake coming off a swimmer. The waves come off the swimmer with a wake-angle that depends on the speed of the swimmer. The simulated wake-angle matches our experimental. Some of the waves can escape through lane-lines and some reflect off, just like in real life. We took video of competitive swimmers, and are satisfied that our model is realistic because of this comparison of the behavior of our model to what is seen with actual swimmers.

Results:

We were able to program partial reflections and chose a reflection of about 50 - 70% based on our experience as swimmers and our videos of actual swimmers (our real world data). However, if we have too many waves going at the same time the

program slows down. The output of our program was more qualitative than quantitative. We didn't expect what we got because our project didn't turn out the way we wanted it to, mainly because we couldn't get the waves and swimmers to interact with each other. However, we did get some data even though it wasn't what we were expecting. In the model we noticed a pattern of waves coming off the swimmers at about a 45-degree angle. Also the faster a swimmer went the longer and thinner its wake got whereas the opposite was true when the swimmer went slower.

Discussion:

We can't think of a way to have the waves affect the swimmer. One reason for this is because sometimes the waves help people go faster. When swimming, we have noticed that it is easier to swim behind someone and use their wake helping us go

faster using less energy. This might be something to consider programming in the future, however.

Conclusions:

When we ran our code we noticed that wakes got longer and thinner the faster the swimmer went. This matches our experimental data, which means our code is somewhat realistic. People can use this result to try to make pools less turbulent which in turn may help competitors go faster.

Although the above result is interesting, we were hoping to investigate how waves interacted with swimmers too. We have not yet been able to program the aforementioned phenomena.

Acknowledgments:

We are very grateful to our mentors Emily TenCate and Jim TenCate

We are also thankful to the people who helped research:
Phil Coe and Michael Yang.

We are grateful to people who helped us edit our report:
Emily TenCate and Deb Summa

References

- www.seafriends.org.na/oceano/wves.htm
- www.onr.navy.mil/Focus/ocean/motio/waves1.htm
- www.spaceandmotion.com/science-phusics-wsm-wave-diagram.htm
- "If I'm so fit, why is swimming so hard?" in Swimming Made Easy: The total immersion way for any swimmer to achieve fluency, ease, and

speed in any stroke by Terry Laughlin, Total Immersion Swimming, Inc., New Paltz NY, 2001.

- science.howstuffworks.com/wave-pool1.htm

- Philipp Coe, USA-S swim coach and certified pool technician, maintenance director, Larry R. Walkup Aquatic Center, Los Alamos NM

Appendix 1: Graphics

Figure 1: Creation of wake and reflection of waves. This is a picture of our NetLogo simulation screen. There are three lanes, each with a swimmer (in red)in the center. As the swimmers move down the lane (towards the bottom of the picture), wake agents are hatched (blue dots). The wake agents travel (blue lines), eventually dyingout or hitting a lane line or wall. Some waves reflect off these boundaries, others die out.

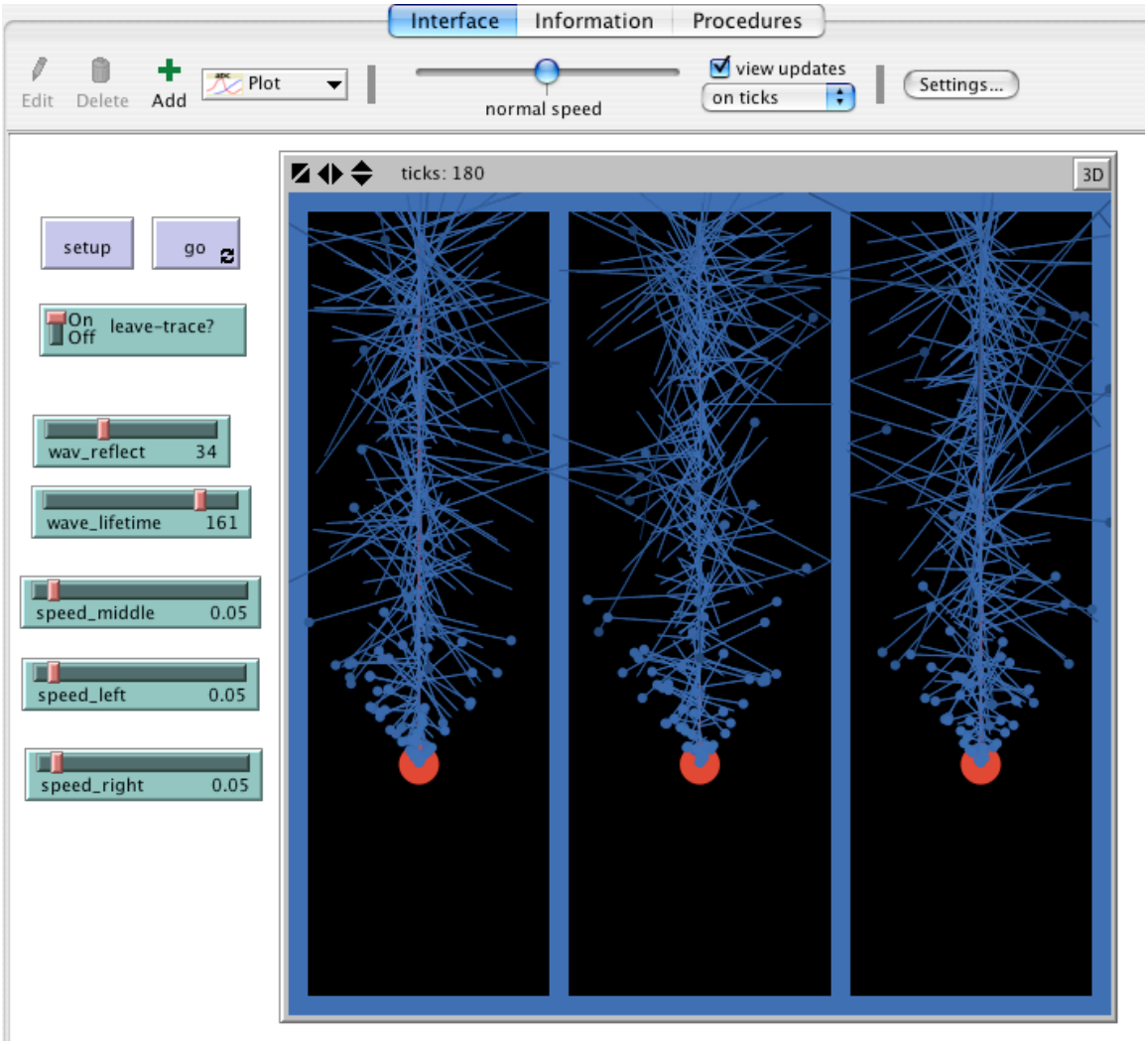


Figure 2: Effect of swimmer speed
Changing the speed of a swimmer changes the wake angle, just like in real life. As you can see in this picture the fastest swimmer (in the center) has a long, narrow wake (wakes outlined in yellow for illustration purposes). The slower swimmers (on the outsides) have a shorter and wider wakes, with the slowest swimmer (on the right) have the largest wake-angl.

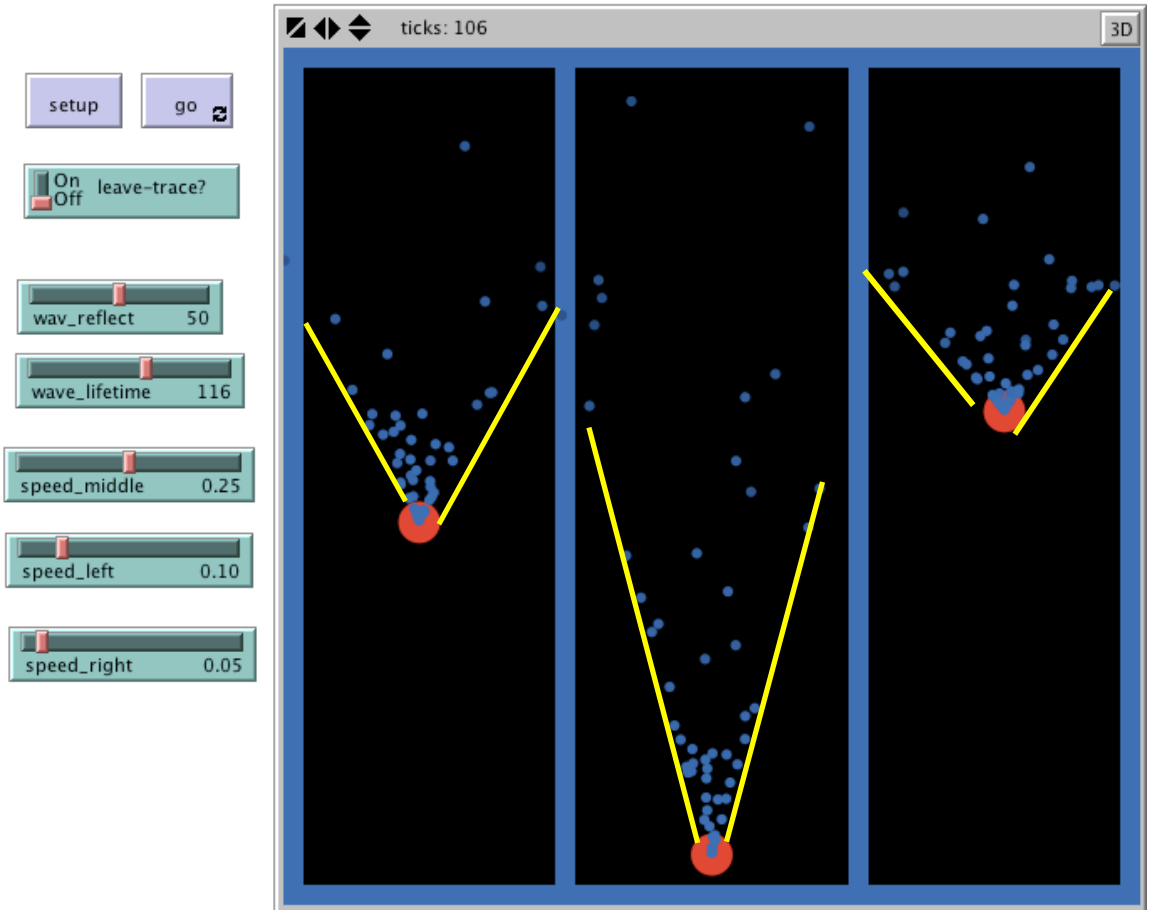


Figure 3: Experimental data-Wake angle from a real swimmer

This is picture of a wake angle that is created by a competitive swimmer swimming freestyle at a moderate pace. The wake angle measures about 45 degrees.

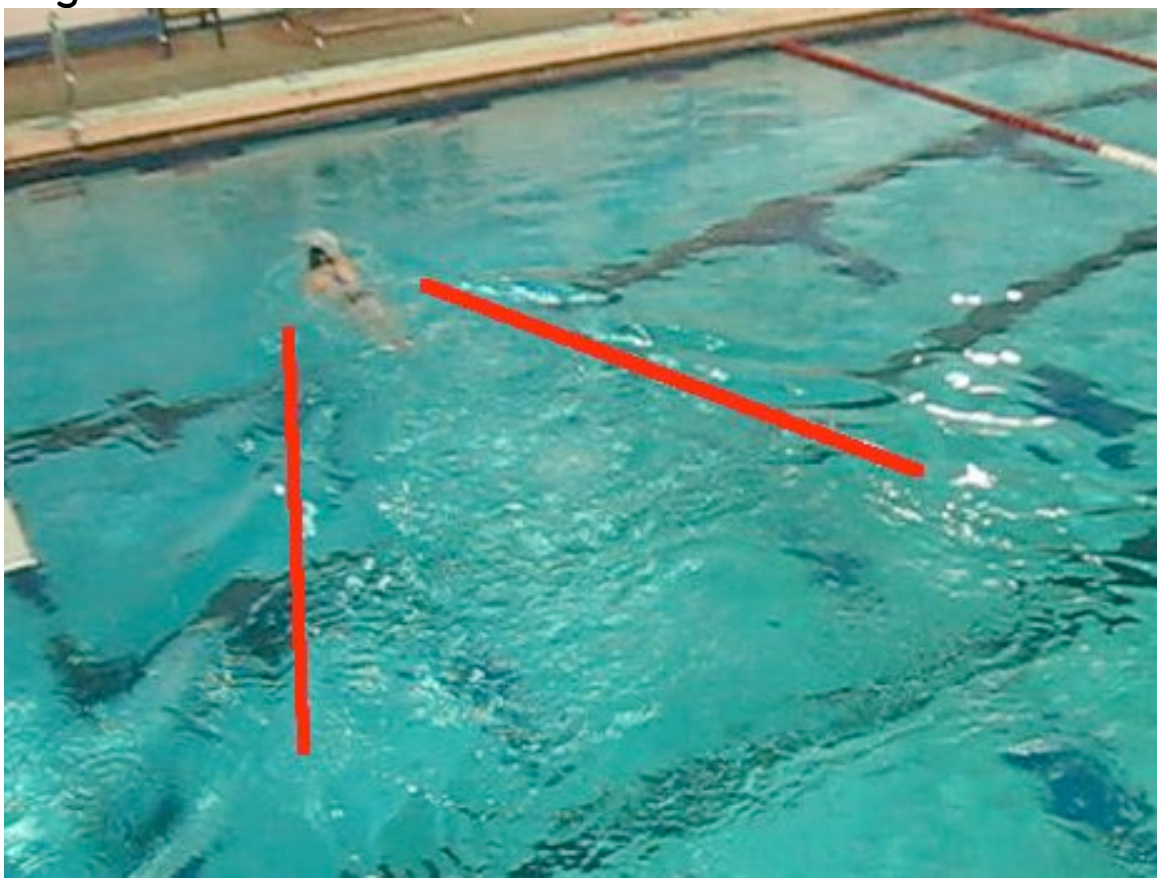


Figure 4: A comprehensive picture of the full simulation.

Here is the front console for our simulation. The pool is in black. The blue outline represents the outer wall, and the blue lines within the pool are the lane lines. Swimmers are in red. The blue dots trailing behind each swimmer are the wave agents. The collection of wave agents make up the developing wake. Sliders on the left of the pool are used to change swimmer speed, the percentage of wave agents that get thru the lane line (transmission factor), and how long a wave agent lives.

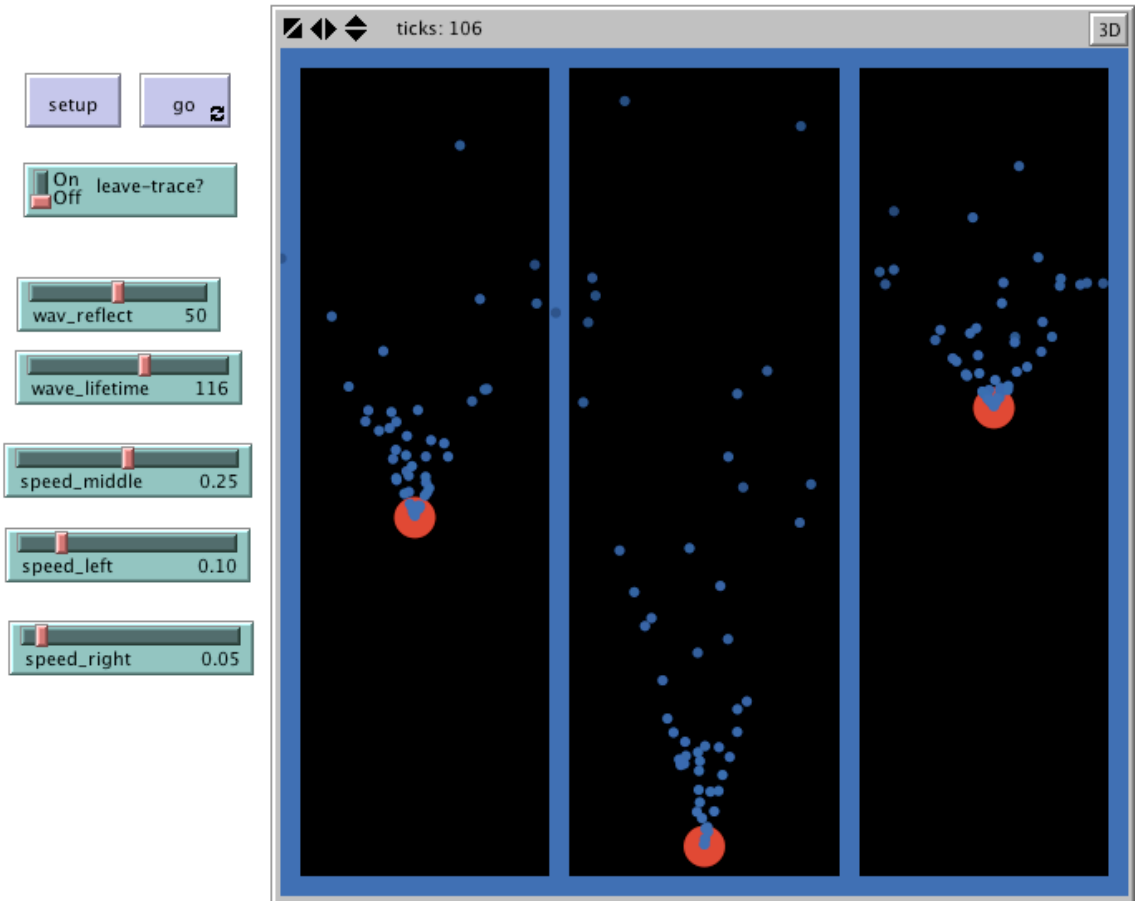
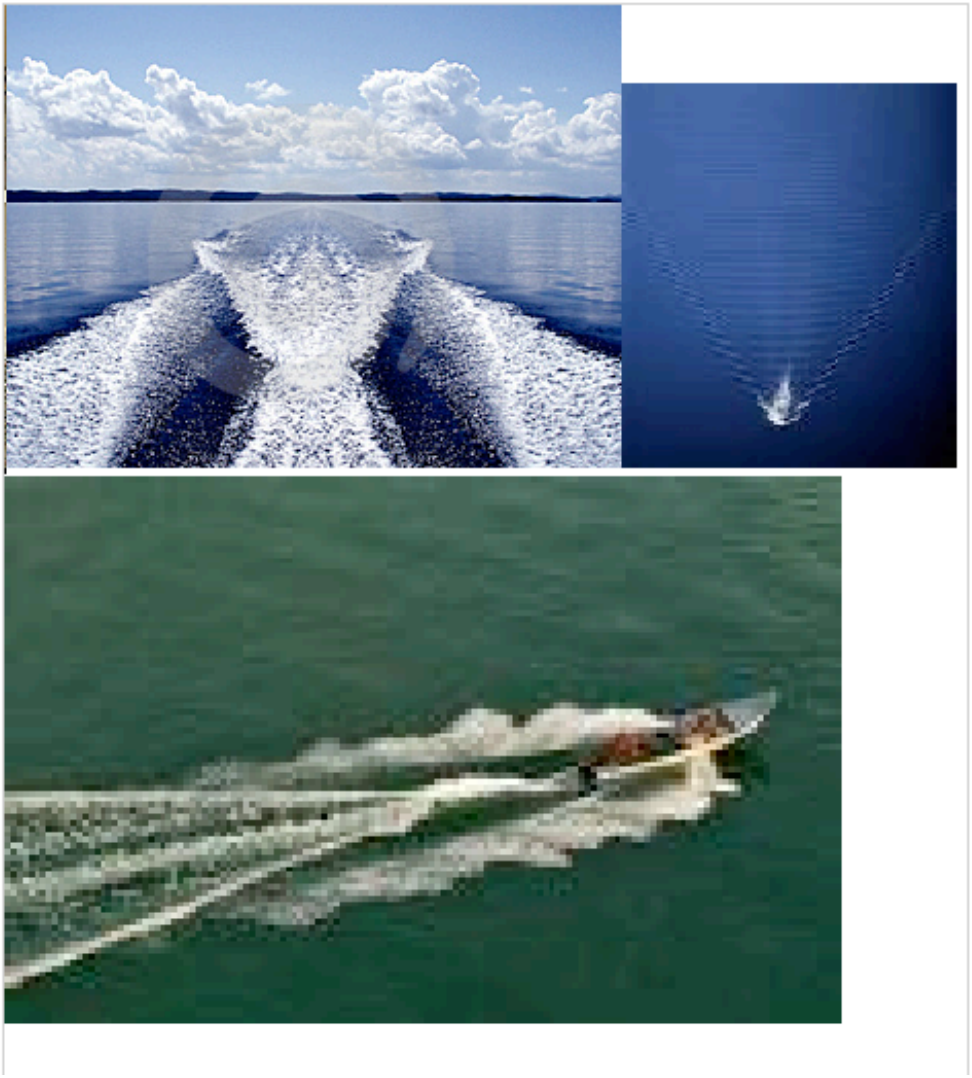


Figure 5: Real world data- Boat wakes

These pictures show boat wake angles which are similar to a swimmer's wake. Again, the speed boat (top left hand corner, going the fastest) has a longer and narrower wake than the slower boats.



Appendix 2: Commented NetLogo Code Listing

```
globals [countsteps] ; makes "countsteps" a variable
turtles-own [wav-number] ; makes "wav_number" a variable




to setup ; creates pool boundaries and sets initial swimmer
conditions

  ca
  draw-walls ; creates  pool boundaries

  set-default-shape turtles "circle"
  crt 3 ; creates 3 swimmer agents


  ask turtles [

     set heading 180 ; tells swimmers which direction to move
     set color red ; makes swimmers red
     set size 2 ; makes swimmers large

    ; this block sets initial coordinates for the swimmer agents
    if who = 2
         [set xcor 14
          set ycor 19]
     if who = 1
          [set xcor -14
          set ycor 19]
     if who = 0
          [ set ycor 19]

  ]
```

```
end

to move ; allows turtles to move and make waves

  if who = 0; specifies Agent #0

    [ fd speed_middle; sets the agent's speed to the value
specified on the slider
       hatch 2 [; creates 2 new agents (waves) for each step taken
            set size 0.5 ; wave agents are small
            set color blue ; wave agents are blue
            randomize ; sets a random wave agent heading
            set countsteps 0 ; sets the number of steps that new
wave agents have taken to 0
            set wav-number random 100 ; sets the % of wave agents
that can pass through lane-lines, according to the "wav-number"
variable
        ]
     ]

; the following blocks do the same as described above for other
swimmer agents
  if who = 1
    [fd speed_left
    hatch 2
      [ set size 0.5
        set color blue
        randomize
        set countsteps 0
        set wav-number random 100]]

  if who = 2
    [fd speed_right
    hatch 2
       [ set size 0.5
```

```
        set color blue
        randomize
        set countsteps 0
        set wav-number random 100]]
end



to draw-walls ; creates pool boundaries and lane-lines

; creates pool boundaries
    ask patches with [abs pxcor = max-pxcor]
      [ set pcolor blue ]
   ask patches with [abs pycor = max-pycor]
      [ set pcolor blue ]


; creates lane-lanes
  ask patches with [abs pxcor = -7]
    [ set pcolor blue ]
  ask patches with [abs pxcor = 7]
    [ set pcolor blue ]


 end



to randomize ; sets a random heading for any agent

      set heading random 360

end
```

```
to go ; the main running loop of the program

  ask turtles [
    ifelse leave-trace?  ; determines whether agents leave visible
paths
      [ pd ]
      [ pu ]


   move ; subprocedure "move"
   set countsteps countsteps + 1 ; agents' "countsteps" count
increases by 1
   bounce ; subprocedure "bounce"
   fd 0.1 ; makes wave agents move forward 0.1 steps

; if any wave agent's "countsteps" count is above the specified
variable (wave lifetime), then the agent dies
   if ( color != red) and (countsteps >= wave_lifetime)
      [ die ]

 ; wave agents grow darker as they age
   if color != red
       [ set color (color - 0.02) ]
 ]

  tick; sets graphics window ahead another step

end


to bounce ; waves and swimmers bounce off of lane-lines and
boundaries
```

```
; makes swimmers bounce off pool boundaries (walls)
if (color = red) or (wav-number >= wav_reflect) [

   if abs [pxcor] of (patch-ahead 0.01) >= max-pxcor
      [ set heading (- heading) ]
   if abs [pycor] of (patch-ahead 0.01) >= max-pycor
      [ set heading ( 180 - heading) ]


; makes agents bounce off lane-lines, according to the
"wav_reflect" variable specified in the slider
   if pcolor = blue
      [ set heading ( 180 + heading )
        set heading ( 180 - heading) ]

 ]

end
```