EPCC-SS-97-10

Interactive Computer Simulation of the Principles Behind How Fish Swim

Iroro Fred Onome ORIFE

Abstract

The main objective of the project was to develop an online teaching tool to be used as a supplement for a second year course in fish biology. The HTML based tutorial firstly introduces the basic concepts behind fish locomotion. Then, along with the text, diagrams and fish pictures, interactive Java applets allow the students to enter data into simulations and view the results. Animation applets are also embedded in the hyper-text to demonstrate different locomotive techniques.

Contents

1 Introduction

Fish, from a purely biological standpoint, make interesting subjects of study due to their incredible diversity, physical features and habits. Their forms of locomotion are equally diverse, from the stingray to the flying fish to the seahorse. There are accordingly many different models from zoology to hydrodynamics that examine how fish swim.

There are many different factors governing these animals' ability to live, adapt and reproduce in their environments. As one studies fish and their different forms of locomotion, there are series of questions that become natural to ask, such as "Why do certain fish have a particular shape" or "Why don't certain species move as fast as other species".

The goal of this project was to develop an online (World Wide Web) software tutorial package for second year biology students in an effort to provide a novel way for them to discover and answer these questions for themselves. The tutorial will be a compliment to their course on fish biology at Heriot-Watt University, Edinburgh.

2 Some Background

2.1 The Tutorial's End-user

The main objective of the project was to develop an online tutorial which would be used as part of a second year course on fish biology. The students are taught the basic mathematics and physics behind fish swimming, but this can be rather dry. It can therefore be difficult for them to draw connections between the formulae and what is physically taking place.

The finished online tutorial enables students to browse a series of web pages, with introductory text and eye-catching fish images explaining the basic principles of fish locomotion. After completing a section, they can broaden their new knowledge by working with an interactive simulation or viewing a Java animation. For example, after having completed a section on drag forces, they can continue with a simulation, varying certain parameters associated with fish swimming through a medium to investigate the resultant power output and drag forces experienced by the fish.

There were several important points that had to be kept in mind during the design of the tool: Ease of use, ease of navigation and attractiveness.

Furthermore, Java was chosen as the implementation language for the simulations because of the "online nature" of the project and need for interaction between the tutorial and the students. In this way, Java applets demonstrating more dynamic ideas and principles could be imbedded in HTML documents that convey more static information such as fish facts and images.

An additional bonus of using Java is that the tutorial is platform independent. This means that it could be run from PCs, Macintoshes or Unix workstations. The only requirement to use the tutorial is a Java and Javascript enabled web browser, such as Netscape Navigator 3 or Microsoft Explorer.

2.2 The Contents

The tutorial starts by giving the general biological background on how fish generate thrust, the various forms of swimming employed by different species and how this relates to their lifestyles. This introduction is accompanied by fish related graphics and Java animations of the different locomotive styles.

After this introduction, the software goes on to introduce the ideas behind friction, drag, power and velocity and their inter-play. This discussion is followed by an interactive applet that allows the students to enter different velocity values for different shapes and look at the resulting drag forces experienced and the power needed to attain the chosen speed. There are links to an equation reference page for the more curious.

After dealing with the basic physics behind forces like drag and friction, the tutorial discusses different adaptations that fish have made in their evolution to overcome these obstacles. It deals further with the issue of size and its affect on swimming speed and the idea behind the ratio that is *Reynolds Number* is introduced.

The students are then presented with another simulation which gives them a chance to specify different values for the tail beat frequency and body length and find the resulting maximum swimming speed given those values.

The main section of the tutorial concludes with some discussion on the the design of a "fast fish", and why not all fish look like that. After completing the core section of the tutorial dealing with the basic physics of fish locomotion, the students are allowed to explore, a more specialised, " advanced topic", in this case fast starts. This is a specialised form of fish locomotion in which there is a rapid acceleration from a stand-still. It is employed by ambush predators to catch prey and by others to escape ending up as dinner.

Here the students can delve into a sub-field of biophysics that isn't as simple and is quite possibly more intellectually interesting. The section is prefaced by a quick introduction to different kinds of swim capacities and fast-starts in general. Then there is a visualisation applet. By clicking on the *Esox lucius* the students can watch it perform a quick get-away.

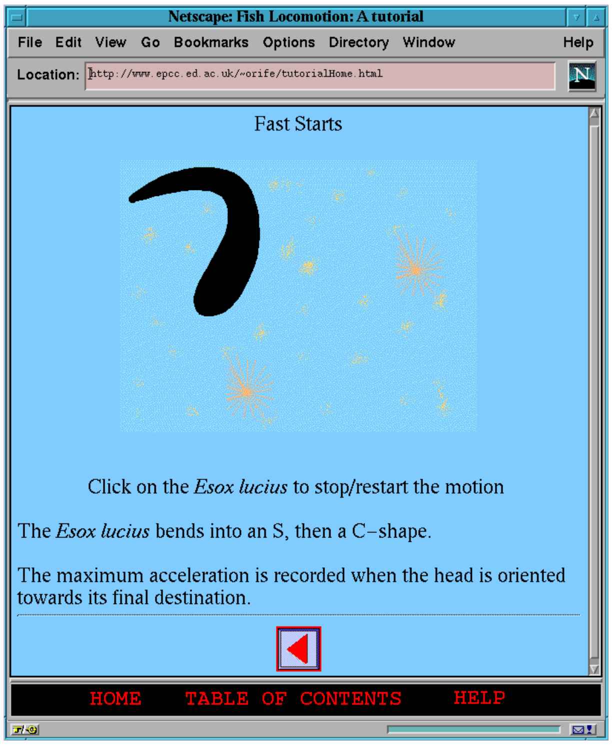


Figure 1: An applet demonstrating the fast-starts of an *Esox lucius* or northern pike

3 The development of the online tutorial package

3.1 AWT: GUI Development in Java

All of the simulations in the tutorial were written in Java. Java is an object oriented language closely related to C/C++. The main reason why developers are interested in Java is that is designed for programming on the Internet. What this means is that Java enables application designers to write completely new kinds of software, ones that can now be downloaded from an arbitrary site and run it on virtually any platform. This is important since the students using the tutorial could be on PCs, Macintoshes or Unix workstations.

All of the simulations and animations were Java applets. An *applet* is a essentially a Java program that can be embedded within another application. Applets adhere to a set of conventions that lets them run within a Java-compatible browser. The simulation applets were included in HTML documents and provided interactive content to enrich the static fish locomotion text and images.

3.1.1 The Abstract Windowing Toolkit

The AWT (Abstract Windowing Toolkit) is a Java package that is part of the standard JDK1.0.2 API (Java Development Kit Version 1.0.2 Application Programmer's Interface) that permits the development of graphical user interfaces. This toolkit provides a number of classes, each providing a graphical interface component (buttons, menus, windows, etc.) which can be combined with other components to build a complete Graphical User Interface (GUI).

The toolkit is named as an abstract toolkit as it provides only the essential components and functionality which are common to all major windowing systems. For example, the Microsoft Windows environment, the Apple Macintosh environment and the various X Windows environments all supply button components. However, the physical appearance of the button, its precise behaviour and API differ significantly between all three environments. Java, on the other hand abstracts the essential behaviour of a push button and provides an API which is presented to the developer in the AWT. When an instance of the Java Button class is realised, the actual button that the user sees is obtained from the native environment. [**?**]

3.1.2 bagOButtons

In spite of the many inherently good features of AWT, some of the components available are quite bland. This is to say, that the standard buttons provided by the `Button` class aren't very flashy or eye-catching. Furthermore, you can't load images onto them, as they are purely text-buttons. A web search was performed and an independent package downloaded and imported into the applet. The package, called bagOButtons, implemented several different button classes that allow images (GIFs, JPEGs) to be loaded onto the buttons, to have a "sticky" effect (where a pressed button remains depressed until pressed again) and a variety of other goodies that enhance the functionality and appearance of the button.

3.2 Java and Event Handling

After setting up the simulation GUI with buttons, text fields, scrollbars and the like, the next programming task was to implement what is called the "event handling". Events are essentially what allows the Java simulation applet to interact with the students. When the user clicks on a button or enters something into a text field or types return on the keyboard, the applet must be able to receive these forms of user input and use the data to calculate and output the necessary results.

3.2.1 The Event Queue

The nature of the data input is asynchronous because the users can often select menu items, type text, click the mouse, or otherwise provide input at almost any time. To deal with this, GUI programs use an event approach. Each user action, a keystroke, mouse click or anything else is placed in an *event queue* as it occurs. An infinite while loop then reads events from the queue and executes the appropriate code to handle them.

Normally, it is the responsibility of the operating system to manage the event queue and, depending on the architecture of the system, there may be only one system event queue or each application may have its own event queue.

In Java applications/applets, each virtual machine has one main AWT event queue. There's almost certainly a native event queue, but that won't be discussed.

3.2.2 The handleEvent() Method

So how exactly does one implement an event handler to perform some calculations when the user clicks on a button or presses Enter? This is controlled by the method called `handleEvent()` with the prototype:

```
protected boolean handleEvent(Event e)
```

The body of this method takes the events from the event queue and examines them by a switch statement on the `Event.target` and performs the necessary computations depending on the target in question. The target is the target component and it simply indicates the component over which the event occurred or with which the event is associated.

In plain English, if a button is clicked, three events would be generated, one for the mouse move, the mouse down and the mouse up events. Each of these events has the button name as its target and is passed in sequence to `handleEvent()` which performs calculations based on the event id.

3.3 Implementing Animation Sequences in Java

Aside from implementing simulation applets that allowed some level of student interaction with the software, animation applets were also an important part of the visualisation section, which followed the discussion on different modes of fish locomotion. The five animation applets that were produced brought to life the Anguiliform, Sub-carangiform, Carangiform and Thunniform modes of locomotion, as well as the fast start techniques used by ambush predators and prey alike.

3.3.1 The animation loop

Every animation program has an animation loop, which displays consecutive images in the animation sequence. Generally, the loop has its own thread. It is never implemented in the `paint()` or `update()` methods since that would take over the main AWT thread, which is in charge of all the drawing and event handling.

The animation loop does the following over and over again:

1. Calls the `repaint()` method to request that the `paint()` method be called. (The `paint()` method draws the graphic that corresponds to the current frame number)
2. Sleeps for up to `delay` milliseconds.
3. Advances the frame number.

3.3.2 Additional Features

A couple more nice features of online animations that were kept in mind during development were:

1. The ability to suspend the applet's execution when the user leaves the page. This was done by overriding the `stop()` method so that it prevents the animation thread from performing. When the user revisits the page, the `start()` method restarts the animation thread.
2. Allowing the user to explicitly stop and restart the animation, without leaving the page. Animation can sometimes be distracting, especially if there is important information below, so it's important to give the user the power to stop the motion, to be better able to concentrate on something else. This feature is implemented by overriding the `mouseDown()` (a close relation of `handleEvent()`) method to catch the mouse clicks and call the `start()` or `stop()` method, depending on the thread's current state. [**?**]

3.3.3 Frames

All of the animation frames that are used were drawn manually, or scanned in and then seriously butchered and manipulated in `xpaint` or `xfig` until they took on their final form. This is arguably the most difficult part of the whole animation task, as both `xpaint` and `xfig` are relatively primitive. An effort was made to use some three dimensional modelling software. At the end of three days of futile mouse shuffling, despair took over, and with nothing really exceptional to look at, one dimension was cut from the plan of action.

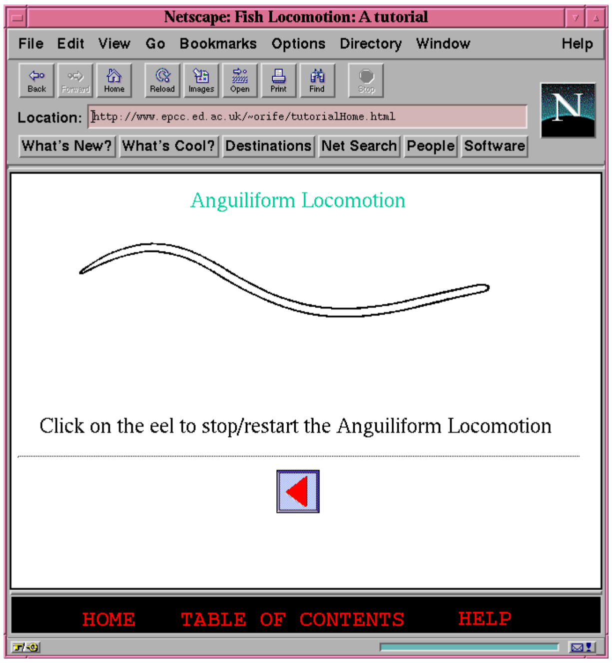


Figure 2: A Java applet animating Anguiliform locomotion

3.4 Javascript: Spicing up the GUI

To supplement the layout of the HTML pages and Java applets, some Javascript features were added to make the pages more visually appealing.

3.4.1 Javascript Basics

Javascript is a compact, object-based scripting language for developing client and server Internet applications. A Javascript enabled browser interprets Javascript statements embedded directly in an HTML page. In a client application for Netscape Navigator, Javascript statements embedded in a HTML page can recognise and respond to user events such as mouse clicks, form input, and page navigation.

One can write a Javascript method, for example, to verify that users enter valid information into a form requesting a fax number or postal code. Without any network transmission, an HTML page with embedded Javascript can interpret the entered text and alert the user with a message dialog if the input is invalid. Javascript can also be used to perform an action (such as play an audio file, execute an applet, or communicate with a plug-in) in response to the user opening or exiting a page.

“The Javascript language resembles Java, but without Java’s static typing and strong type checking. Javascript supports most of Java’s expression syntax and basic control flow constructs. In contrast to Java’s compile-time system of classes built by declarations, Javascript supports a run-time system based on a small number of data types representing numeric, Boolean, and string values. Javascript has a simple instance-based object model that still provides significant capabilities.

Javascript also supports functions, again without any special declarative requirements. Functions can be properties of objects, executing as loosely typed methods.

Javascript complements Java by exposing useful properties of Java applets to script authors. Javascript statements can get and set exposed properties to query the state or alter the performance of an applet or plug-in.

Javascript descends in spirit from a line of smaller, dynamically typed languages like HyperTalk and dBASE. These scripting languages offer programming tools to a much wider audience because of their easier syntax, specialised built-in functionality, and minimal requirements for object creation." [**?**]

3.4.2 Cool Event Handling in Javascript

Javascript was mostly used to handle mouse events in the HTML pages. In the table of contents page, for example, when the user passes the mouse over the different hyperlink elements in the table of contents, they light up, but as soon the mouse has passed over them, they go back to their original colour.

The way this was accomplished in Javascript was to have the items in the table of contents stored as images (GIFs), but each item has a duplicate for when the mouse is over it. So when the table of contents page is first loaded, all the item images are normal, but as the mouse passes over the various items, the browser calls a Javascript method `raise()` to swop the particular item image with its duplicate:

```
onMouseOver="raise('img03', 'forms_mouseIn')"
```

When the mouse has moved on past that particular item, the duplicate image is switched out and the original item image is restored

```
onMouseout="raise('img02', 'thrust_mouseOut')"
```

3.5 Javascript and Animation: Some Technical Difficulties

After implementing some trial animations in Java, the possibility of doing the same in Javascript was explored. The idea was to write some methods in Javascript that could simply perform the animation loop as described above in Java. The possible benefit of this strategy is that Javascript is interpreted by the browser, and so it might be faster than having to load up the class files to run an equivalent Java applet.

After trying out several animations, this method was eventually abandoned because the browser used (Netscape Navigator 3.01) was able to animate the images for only a short period. Afterwards, the browser would thrash about, before dying. This experience was quite negative, and compared to the smooth animations provided in Java, animation in Javascript was abandoned, and all dynamic visualisations were written as applets.

4 The resulting software package: Clips and Screen Shots

Since there aren't really any "results" in the traditional sense of the word as it pertains to research papers, I will simply present some screen captures of the various HTML pages, Java applets and figures from the tutorial.

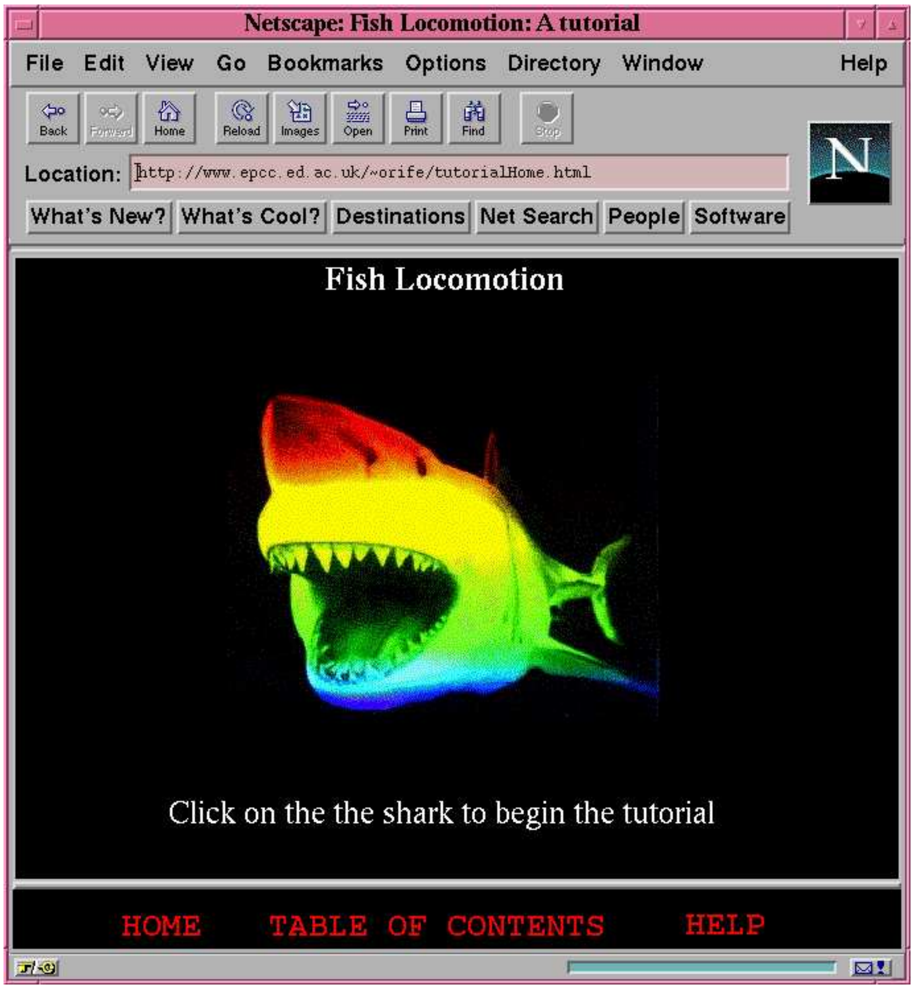
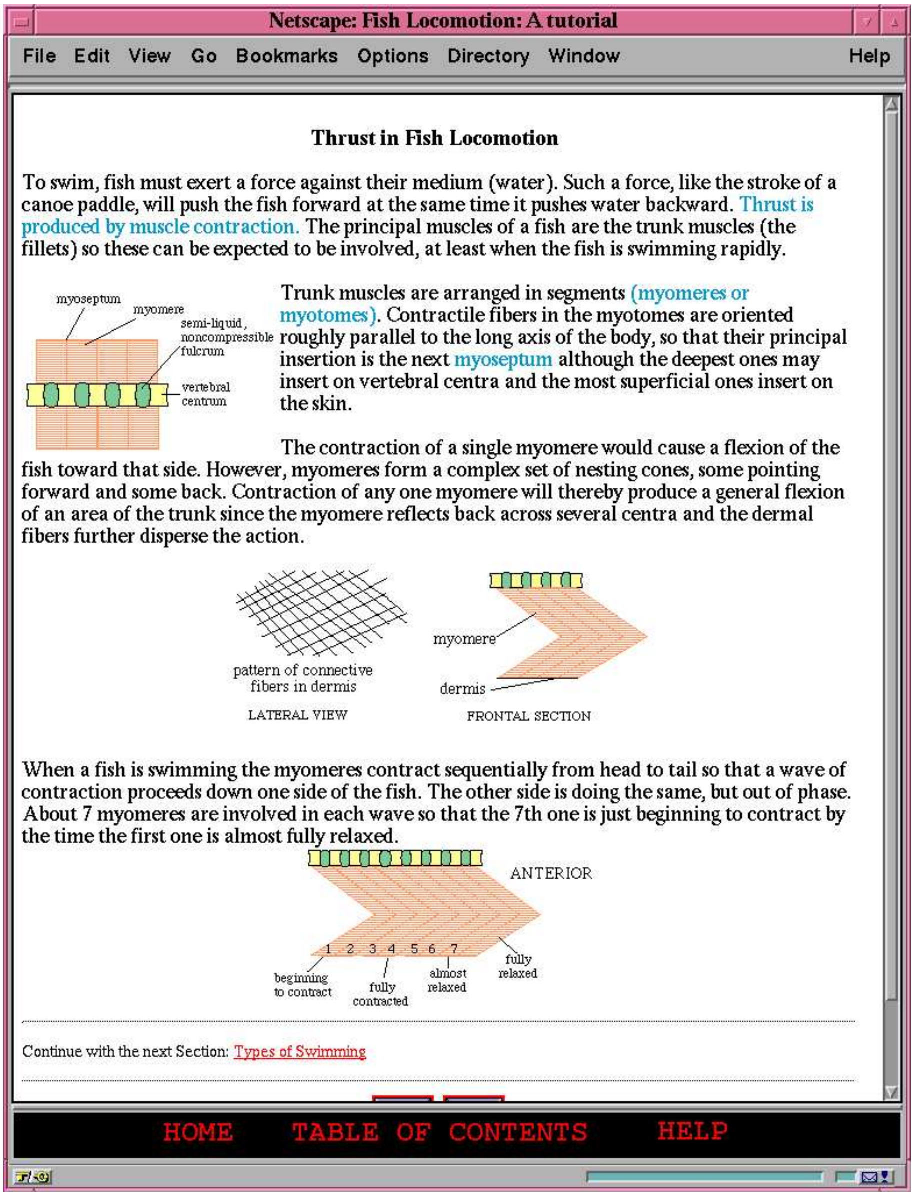


Figure 3: The tutorial home page



Netscape: Fish Locomotion: A tutorial

File Edit View Go Bookmarks Options Directory Window Help

Thrust in Fish Locomotion

To swim, fish must exert a force against their medium (water). Such a force, like the stroke of a canoe paddle, will push the fish forward at the same time it pushes water backward. Thrust is produced by muscle contraction. The principal muscles of a fish are the trunk muscles (the fillets) so these can be expected to be involved, at least when the fish is swimming rapidly.

Trunk muscles are arranged in segments (myomeres or myotomes). Contractile fibers in the myotomes are oriented roughly parallel to the long axis of the body, so that their principal insertion is the next myoseptum although the deepest ones may insert on vertebral centra and the most superficial ones insert on the skin.

The contraction of a single myomere would cause a flexion of the fish toward that side. However, myomeres form a complex set of nesting cones, some pointing forward and some back. Contraction of any one myomere will thereby produce a general flexion of an area of the trunk since the myomere reflects back across several centra and the dermal fibers further disperse the action.

When a fish is swimming the myomeres contract sequentially from head to tail so that a wave of contraction proceeds down one side of the fish. The other side is doing the same, but out of phase. About 7 myomeres are involved in each wave so that the 7th one is just beginning to contract by the time the first one is almost fully relaxed.

Continue with the next Section: Types of Swimming

HOME TABLE OF CONTENTS HELP

Figure 4: Part of the Thrust in Fish Locomotion section of the tutorial

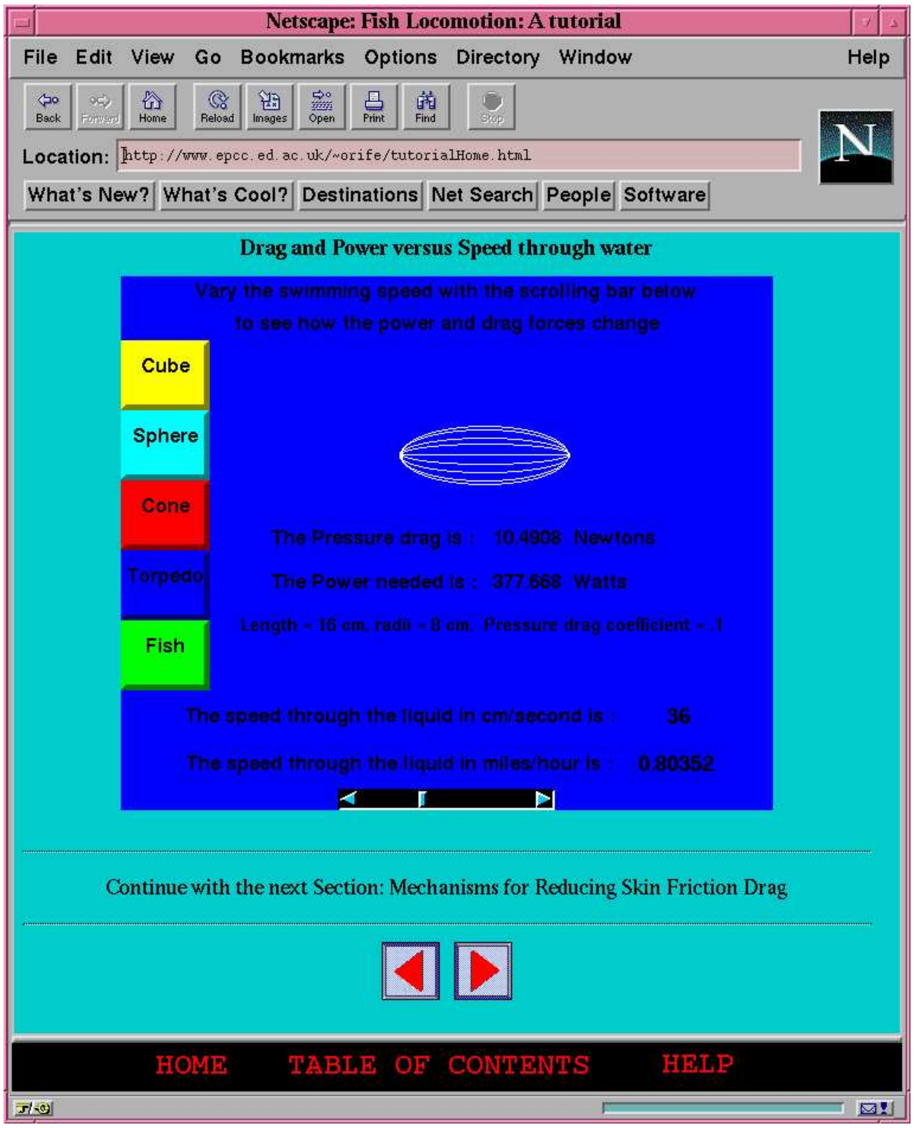


Figure 5: A Java applet used to investigate the drag experienced and power needed to attain different speeds for different shapes

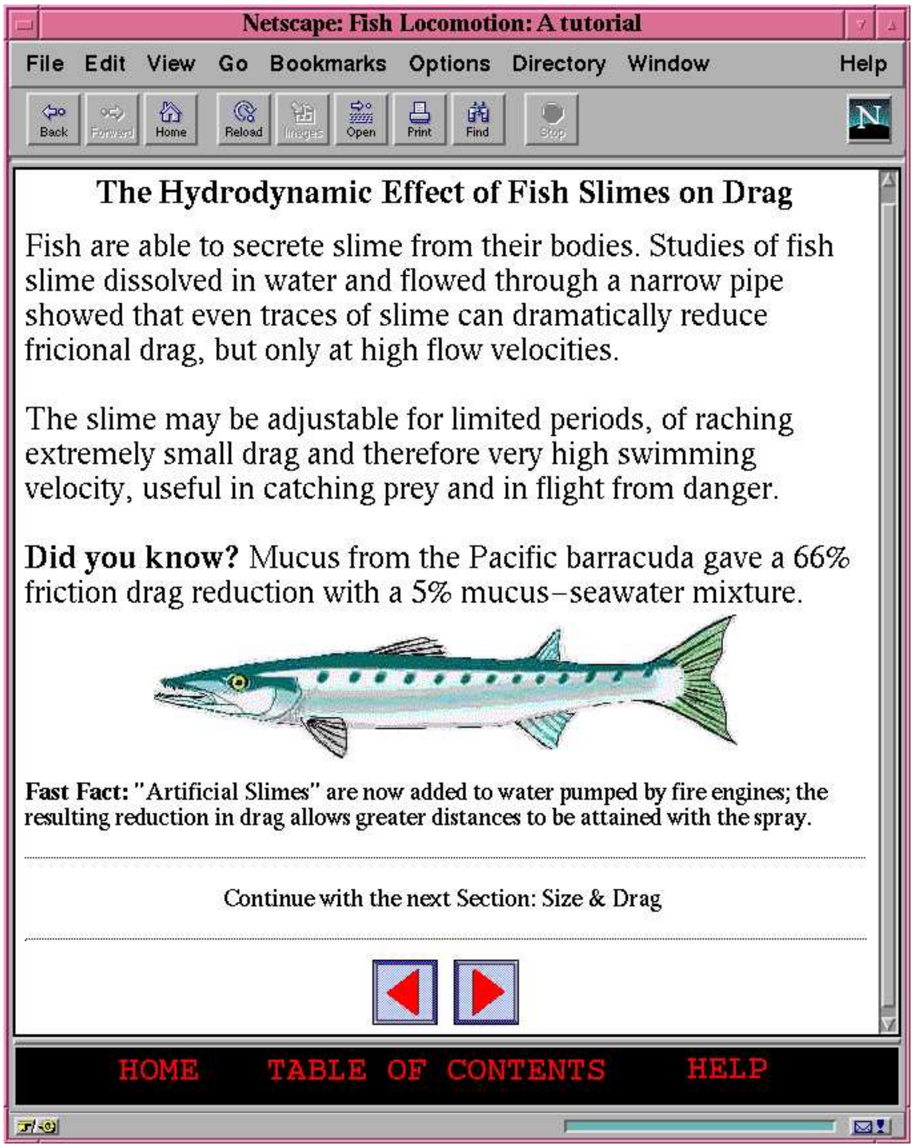


Figure 6: Part of the Drag Reduction Mechanisms section of the tutorial

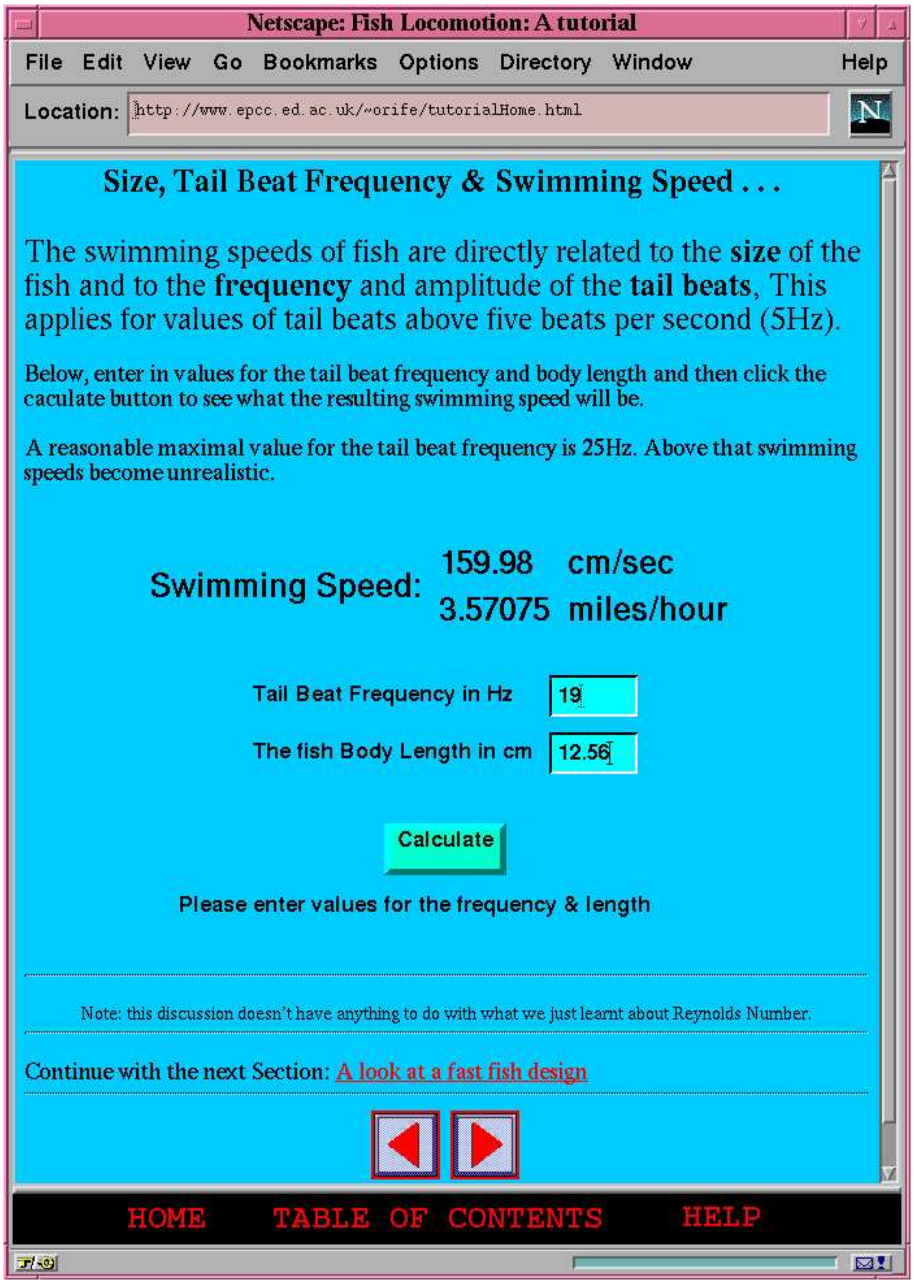


Figure 7: A Java applet demonstrating the relationship between body length, tail beat frequency and swimming speed

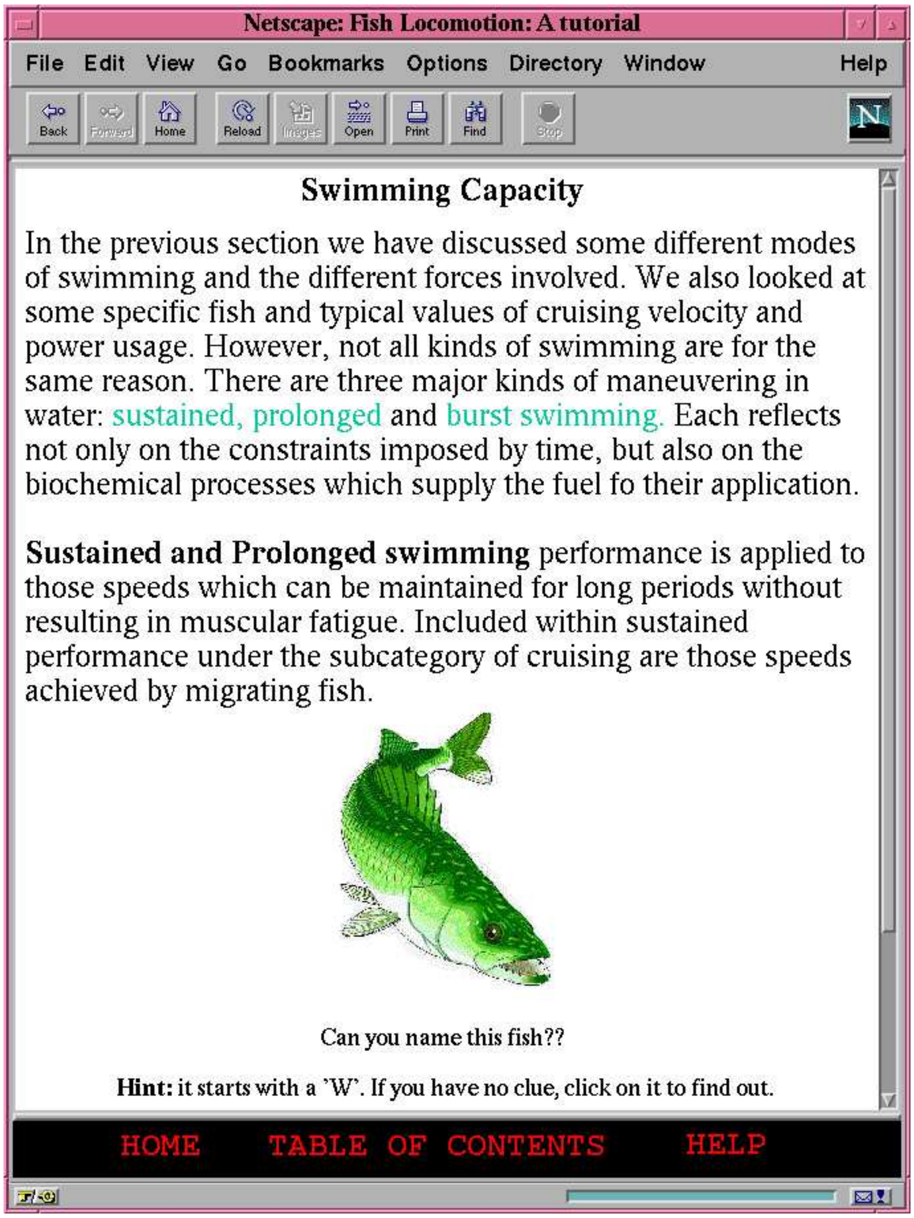


Figure 8: The section detailing fish swimming capacities

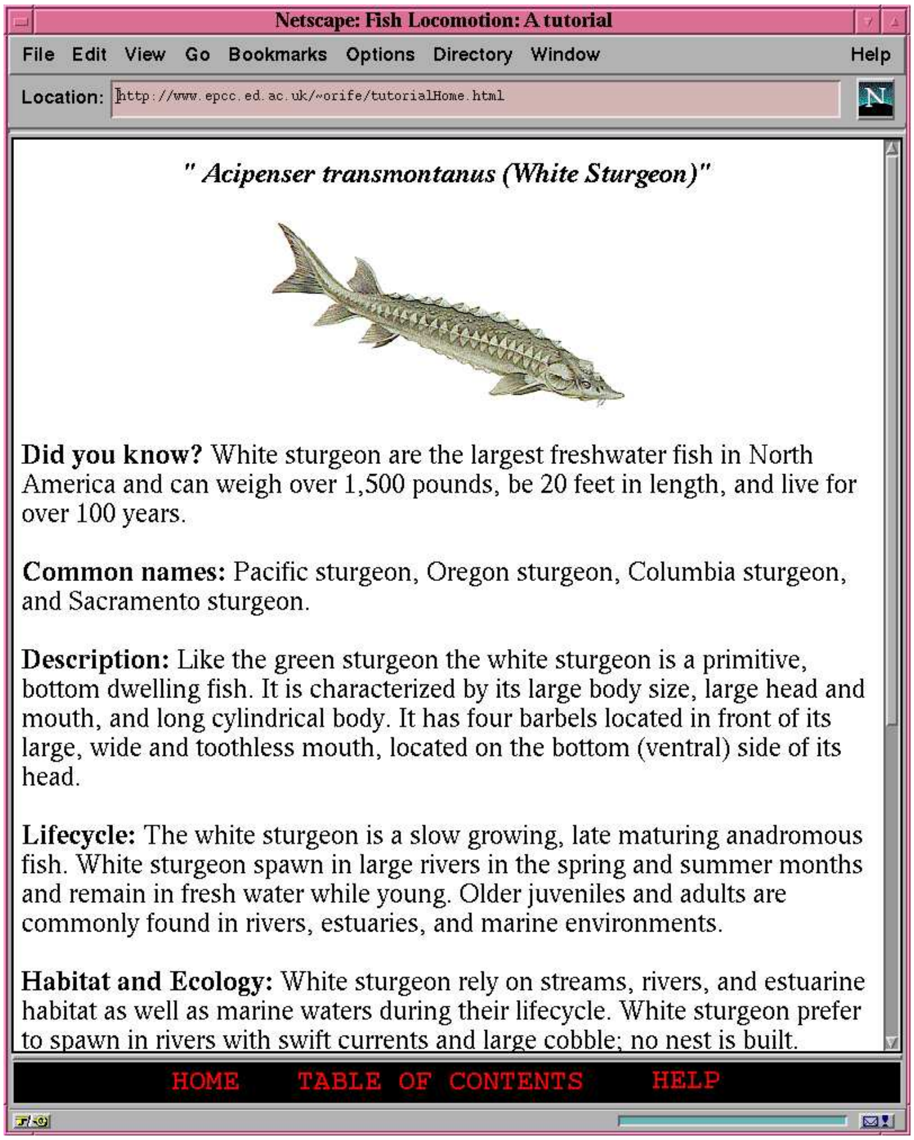


Figure 9: Clicking on the various fish images scattered throughout the tutorial brings up a page like this giving more information on the species shown

5 Conclusion

Most of the more interesting work, such as the simulations and animations, were produced as Java applets. With more time, more complex and sophisticated simulations/animations could be performed. In the current tutorial, the complexity of the animations that were produced was constrained by the time scale of the project.

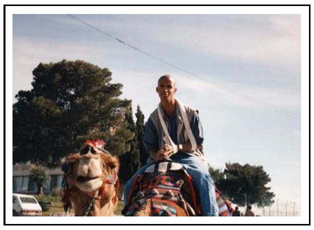
Further work could simulate fish locomotion using 3-D modelling software to create the animation frames, and the same Java code could be reused to perform the animation. Since the general nature of the tutorial was introductory, there are numerous possibilities for add-ons treating the more specialised sub areas of fish biophysics. Computer–aided learning and simulations in biophysics aren't widely used, but this project demonstrates that the only real limits lie within the imagination of the developers and the time available.

References

[1] A Java/AWT Programmer's Primer, "www.scism.sbu.ac.uk/jfl/jicontents.html"

[2] The Java Tutorial, Object-Oriented Programming for the Internet, Mary Campione and Kathy Walrath, "www.aw.com/cp/campione-walrath.html"

[3] Javascript Authoring Guide, Netscape Communications Corporation, "home.netscape.com/eng/mozilla/Gold/handbook/javascript/"



Iroro Orife graduated in May 1997, with a Bachelor of Science double major in Electrical and Computer Engineering and Math/Computer Science, from Carnegie Mellon University. He likes playing tennis, his synthesiser keyboard, travelling and chillin' on the beach.

This project was supervised by Dr. Alastair Lyndon, Dept of Biology, Heriot-Watt University and Dr. Stephen Breuer, EPCC.