Realistic Autonomous Fish for Virtual Reality

Adele Lobb, Shaun Bangay
Email: A.Lobb@ru.ac.za, S.Bangay@ru.ac.za
Computer Science Department
Rhodes University, Grahamstown, 6139

Abstract

We create realistic autonomous fish for Virtual Reality systems. The fish are realistic in appearance, movement and behaviour: the swimming behaviour being non-scripted, within real time rendering.

The form of the fish is procedurally created. The size and shape of the form are controlled by a number of variables which are stored in a simple ASCII file. This allows efficient creation of different fish at run time.

The behaviour is obtained by implementing a flocking algorithm.

Keywords: Modeling, Rendering, Animating, Fish, Procedural Modeling

1 Introduction

The aim of this project is to create fish objects which can be used in virtual reality systems.The fish need to be realistic in appearance and behaviour, while being rendered in real time. Adding a variety of fish to a virtual reality system does not require vast amount of time or coding, this allows researchers to create interesting underwater environments with ease.

The autonomous virtual reality fish has been implemented in RHoVeR, the Rhodes virtual reality system.

This paper discusses the modeling of the fish, and the behaviour algorithms which are implemented.

2 Related Work

As the project had two main areas of interest, we looked at the related works in these two areas. Firstly the modeling of the actual fish and any animation of the fish form, and secondly the group behaviour of the fish.

2.1 Modeling and animation of 3D models

Fröhlich [2000] uses Softimage and Alias/Wavefront to create a textured polygonal geometry structure, which he animates by using the software tools and then storing a number of keyframes to be used by their system later on. The smooth animation of the individual fish is created using linear interpolation between corresponding vertices of each keyframe. He produces very realistic fish in the Virtual Oceanarium. One can also use data from models constructed by someone else. Proudfoot *et al*.[2001] used data of a fish form, created by Terzopolouz [1994], quite successfully, adding bump-mapping to the body and transparent bump-mapping to the fins to produce a static scene of a fish.

There are also many methods of constructing models with the ability to be deformed for animation. A physically based method is proposed by Miller [1988] to create legless creatures such as snakes and worms. The creatures are modeled using a mass-spring system. To simulate the contraction of the muscles, Miller animates the spring tensions. Chadwick *et. al.*[1989] propose a layered construction approach to create deformable animated creatures, where the creatures are built in layers with the relationship between the layers being specified by parametric constraints. Terzopoulos [1999] uses an image-based modeling techniques to convert photographs of real fish into 3D B-splines, which he then textures. To achieve the texturing he uses a snake-grid tool to obtain the nonuniform coordinate system for mapping the texture onto the spline surface. He then creates the muscles for the locomotion using 23 nodal point masses and 91 springs. The spring-mass model uses Lagrange Equations to control the movement.

2.2 Schooling behaviour

The flocking algorithm first created by Craig W Reynolds [1987] is the obvious one to base fish schooling behaviour on. To build a simulated flock, Reynolds starts with a boid model that supports geometric flight. He adds behaviours that correspond to the opposing forces of collision avoidance and the urge to join the flock. Stated briefly as rules, and in order of decreasing precedence, the behaviours that lead to simulated flocking are:

* Collision Avoidance: collisions are avoided with nearby flock-mates.

* Velocity Matching: an attempt is made to match velocity with nearby flock-mates.

* Flock Centering: an individual attempts to stay close to nearby flock-mates.

Each of the rules produces a suggested direction in which to steer the boid. Each rule has an associated fractional "strength". The boid has to collect the different suggested directions, combine, prioritize and arbitrate between potentially conflicting urges. One could use some artificial intelligence algorithm to do this, but an easier way is simply to average them.

Terzopoulos *et. al.* [1994] uses a much more complex behaviour algorithm for their artificial marine life. They use computer vision algorithms to enable their creatures to "see", and learning algorithms to allow them to gain complex motor skills similar to trained marine mammals.

3 Modeling Fish

The surface creating the body and the fins must be made out of triangles, to get effective lighting within OpenGL. The triangle points are calculated at program initialization so as not to slow down running speeds.

To create an individual type of fish, all parameters which determine the characteristics of that fish are stored in a simple ASCII text file, in a certain format, which is read to create information objects.

3.1 Modeling the Body

The contour of the body is controlled using a mathematical function. This function is required to be closed in the interval x=0, and x=1. (i.e. goes through y=0 at these points). This function is then rotated around the x axis to create the surface of the body. To aid future extensions the body shape class FishShapeFunction.java is an abstract class, that can be extended with the implementation of any mathematical function. For VRFish we use a sine function. The function is divided into a number of body segments, each of which are rotated around the x-axis to form a circle. Each circle is divided into a number of points. These points are grouped together to form triangles, which together create the surface of the fish body. (See **Figure 8**)

3.2 Modeling the Fins

There are 4 main categories of fins, namely Pelvic, Pectoral, Caudal and Vertical fins. Vertical fins include Dorsal, Adipose and Anal fins. The fins can be seen in **Figure 1** .

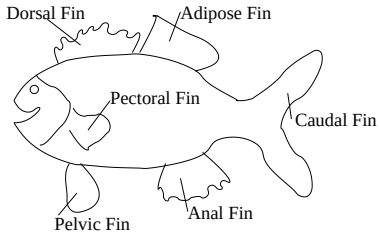


Figure 1: Generic Fins

In nature there is a wide variation in shape and placement of these fins, as can be seen in **Figures 9, 10 [Smith 1986]**

To allow for these variations there are a number of variables which are used in our fin geometry. These control the fin placement, shape and size. The number of variables differ depending on the fin category (dorsal, adipose etc.).

Each of the fins have a slightly different creation algorithm, but they are all rendered in the same way. VRFin.java is an abstract class with 2 abstract methods, createFin, and moveFin. Its third method drawFin is implemented in VRFin.java. All other fin classes extend VRFin.java.

3.2.1 Vertical Fins

The vertical fins include the dorsal, adipose, and anal fin. These fins are in the x-axis-plane and have no swimming animation. The shape and and size of these fins are controlled by the number of body segments the fin spans, (given by start segment, and end segment), the start length, the start angle, the end length and the proportion. The proportion determines the position of the top vertex of the adjacent triangle. (see **Figure 2**)

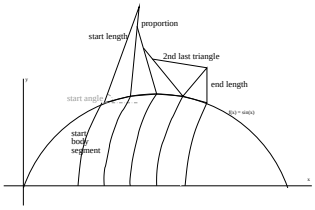


Figure 2: Construction of Vertical Fins

3.2.2 Caudal Fin

The caudal fin is constructed using two halves, each consisting of a number of triangles. The size and shape of the caudal fin is determined by the top length, middle length, bottom length, top angle (which is the angle between the top line and middle line) the bottom angle and the number of triangles in each half. (see **Figure 3**)

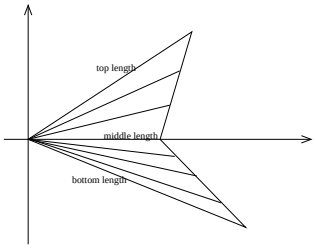


Figure 3: Construction of Caudal fin

3.2.3 Pelvic Fins

The pelvic fins are created so that they are placed in a vertical position below the body. The shape and size of the fins are controlled by the starting body segment, the end body segment, the point on the segment the fin is positioned at, the start length, the end length and the start angle. (see **Figure 4**)

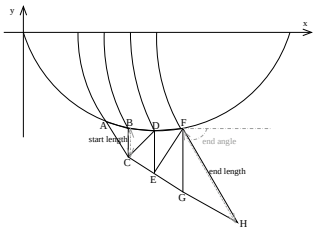


Figure 4: Construction of the Pelvic Fin

3.2.4 Pectoral Fins

The pectoral fin is constructed with a similar algorithm to the vertical fins, with the exception that it is a horizontal fin rather

than a vertical fin.

4 Rendering

After creation the body and each of the fins consist of a set of triangles, which can be drawn using OpenGL to create the fish form. To make it look realistic texturing is required. For this a texture file is required, a picture of a fish in gif format is sufficient.

4.1 Fixel - a Utility program

Each point of the fish has to have a corresponding point on the texture. To aid in determining these points we created a utility program in which the user can click on certain points, and the rest of the texture points are calculated and then saved into a text file. From there the texture points can be copied and pasted into the file containing all the other parameters. (**Figure 11** shows the calculated texture points for the body.)

4.2 Non-triangular fins

Even with the texture added, the fins are very angular and not realistic at all. To rectify this we implement masking. This requires a black and white version of the texture file. The fish is white and the background black.

4.3 Alpha Test

This is a two pass process. A black and white version of the texture is drawn first. During this pass the colour mask is disabled, and the depth mask and alpha testing are enabled and the alpha function is set to LESS. For the second pass, the colour mask is enabled, the depth mask is disabled, and the alpha function to set to GREATER. The colour texture is then drawn. [Neider et al. 1993] **See Algorithm 4.3.**

Algorithm 1 Pseudo code for Alpha Test

Pseudo code for alpha testing

```
disable ColorMask
set texture environmental variables using
   TexEnvf (GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_MODULATE)
enable DepthMask and Alpha Testing
set the AlphaFunction to LESS
Enable 2D texturing, use the black and white picture
Draw the points. Setting texture coordinates,& normals first
disable 2D texturing
enable ColorMask
set the AlpaFunction to GREATER.
enable 2D texturing, use the texture file
Draw the points. Set the texture coordinates, and normals
disable texturing, and Alpha Testing
```

This obtained satisfactory results.

5 Animation

5.1 Fin Motion

To create the impression that the fish is swimming the pelvic, pectoral and caudal fins move. They are simply rotated back and forth around a vector. This is a very simplistic movement model, but creates enough of a swimming motion to convince most viewers.

5.2 Behaviour

To include schooling behaviour we implemented a basic flocking algorithm based on Reynolds flocking algorithm [Neider et al. 1993]. According to Reynolds each Boid (bird object) has three main behaviours which influence the way it moves:

1.Boids try to steer the same way as other Boids

2.Boids try not to bump into other Boids

3.Boids try to move to the center of the flock of Boids.

These behaviours all depend on the location and behaviour of the rest of the flock.

Conrad Parker [1985] extends Reynolds flocking algorithm and suggests that one can add other rules to improve and create a more complex behavior.

The main rules specified by Reynolds are implemented in VR-Fish, as well as a 4th rule which involves following a random path, using a spline to go from one point to the next. This allows a single fish to swim on a random path in the virtual reality environment, but as soon as there are other fish of the same type it will initiate the schooling behaviour. Each rule is weighted differently to create the final direction the fish moves in. These weightings are stored in the ASCII file.

The VRSchool class, reads in the text file once, and instantiates multiple fish, changing some of the parameters slightly for each fish. At this stage the only variation is the initial position. [1]

6 Results

6.1 Performance

We have created a number different fish. (This means that we have a number of different ASCII files which the system can read in.) For these experiments all fish are made up of 16 body sections, each of which are divided into 12 points. The size, shape and placement of the fins, weightings for schooling behaviours and initial velocity differ between fish types. In these experiments there are no other virtual objects within the system besides the fish.

The calculations for the creation of the fish occur before running, so it is the rendering and fish interaction which influence the frame rate.

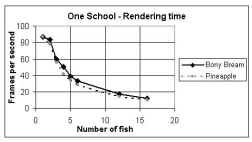
All tests are done on a Pentium III, 500MHz, 128MB RAM, with a Voodoo 5 graphics card, running Redhat 7.2 Linux.

Figure 5 shows the frame rates for the number of fish within the system, using only 1 type of fish in the system at a time. As expected, as the number of fish within the system increases the frame rate decreases. There is a slight difference between the 2 types of fish - which is as expected due to the pineapple fish having large fins, to accommodate the shape of the fins.

Even with 16 fish within the system, it is above the 10 frames per second aim, to allow for real time rendering.

The graph in Figure **6** shows the rendering time when we have two different schools of fish in the system, namely a school of Bony Bream and a school of Pineapple fish. A fish will school only with fish of the same type, so there is no interaction between the different schools. As expected as the number of fish in each school increase so the frame rate decreases.

[1] Animation: http://www.cs.ru.ac.za/research/g90f2972/Overview.html



(a) OnesSchool at a time

Figure 5: One school in the environment at a time

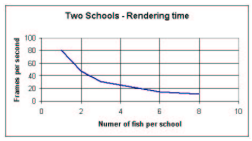


Figure 6: Two schools in the environment

6.2 Appearance

Figure 12 is a scanned in photo of a*Monocentris Japonicus* (Pineapple fish) from J.L.B. Smith's Sea Fishes. Used as the texture file.

Figure 13 is a black and white version of the texture file.

And **Figure 14** is the end result of of the fish created using VRFish.

7 Future Extensions

A more complex schooling behaviour can be implemented by adding more rules, such as hunger, fear and object avoidance.

Different body shapes can be added to allow for differently shaped fish bodies. For example: Extend FishShapeFunction.java with a mathematical function which will produce a graph such as the graph in **Figure 7** in order to get the shape for the Angle fish **Figure 15.**

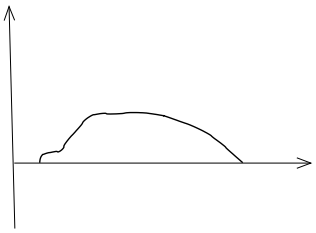


Figure 7: Graph for Angelfish

8 Conclusions

A fish library object has been created, which fulfills all required aims. Different types of fish can now be inserted into real time virtual reality systems. They have sufficient realism in their appearances, movement and behaviour to create an interesting undersea virtual environment .

The fish are procedurally created, the size, shape and behaviour are controlled from a text file, which can be changed and adapted with ease.

The design of VRFish allows for future extensions and modifications without the need to rewrite the entire system.

References

CHADWICK J. E., HAUMANN D. R., PARENT R. E., 1989. Layered Construction for Deformable Animated Characters, *ACM SIGGRAPH 1989, Computer Graphics*, Volume 23, Number 3, pp 243-252

FRÔHLICH T, 2000. The Virtual Oceanarium, *Communications of the ACM*, Volume 43, Number 7, pp 95-101

NEIDER J. DAVIS T. WOO M., 1993, *OpenGL Programming Guide*, Addison-Wesley Publishing Company, pp 291-324.

MILLER G. S. P., 1988. The Motion Dynamics of Snakes and Worms, *Computer Graphics*, Volume 22, Number 4, pp 169-178

MOLOFEE J.2000. Lesson 20, [On-line]
Available: http://nehe.gamedev.net/tutorials/Lesson20.asp?1=20 [Accessed on 12 December 2001].

PARKER C. 1985, Boids, [On-line].
Available : http://www.vergenet.net/~conrad/boids/pseudocode.html [Accessed on 2 November 2001].

PROUDFOOT K., MARK W. R., TZVETKOV S., HANRAHAN P., 2001. A Real-Time Procedural Shading System for Programmable Graphics Hardware, *ACM SIGGRAPH 2001*,pp159-170

PLATT J. C., BARR A. H., 1988. Constraint Methods for Flexible Models, *ACM SIGGRAPH 1988, Computer Graphics*, Volume 22, Number 4, pp 279-288

REYNOLDS C. W., *1987.*Flocks, Herds, and Schools: A Distributed Behavioural Model, *ACM SIGGRAPH 1987, Computer Graphics*, Volume 21, Number 4.

SMITH J.L.B, 1986, *Smiths' Sea Fishes*, ed. Smith M.M. Heemstra P.C., South Book Publishers (Pty) Ltd.

TERZOPOULOS D., 1999. Visual Modeling for Computer Animation: Graphics with a Vision, *Computer Graphics*, pp 42-45

TERZOPOULOS D., TU X., GRZESZCZUK R., 1994. Artificial Fishes: Autonomous Locomotion, Perception, Behavior, and Learning in a Simulated Physical World, *Artificial Life*, Volume 4, Number 1, pp 327-351.

Travel Egypt, 2000.[On-line].
Available: http://www.touregypt.net/vdc/Rsfish.html, [Accessed on 3 September 2002].

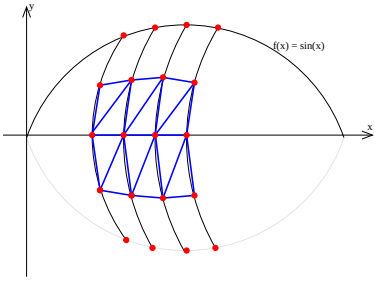


Figure 8: Modelling the Body

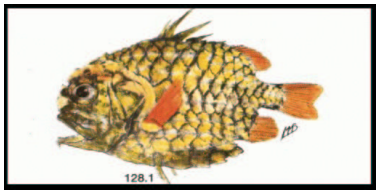


Figure 9: *Monocentris Japonicus* (Pineapple fish)

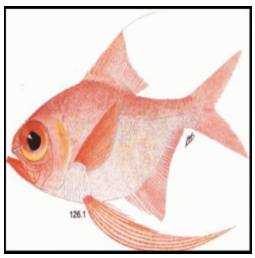


Figure 10: *Beryx decadactylus* (Beryx)

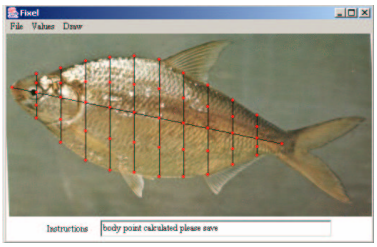


Figure 11: Fixel - Utility Program for calculating texture points

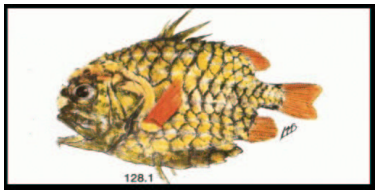


Figure 12: *Monocentris Japonicus*

Figure 13: Black and White version of *Monocentris Japonicus*

Figure 14: *Monocentris Japonicus* created for Virtual Reality

Figure 15: Angelfish