7 MODELING FISH AND ITS MOVEMENT

Hisamudin Yusof, Norhaida Mohd Suaib

INTRODUCTION

Artificial life is a very interesting research area. It involves a multidisciplinary knowledge like mathematics, biology, genetics, physics, artificial intelligence and computer graphics. Modeling and animation of the artificial life has received a lot of attention from computer graphics researchers. It has been widely used in the development of computer games, animation films and smart robots, as well as in virtual reality applications. This chapter will discuss on the theory behind artificial fish and the emphasis on the computer graphics - modeling and animation of an artificial fish.

ARTIFICIAL FISH

One of the pioneers in artificial fish modeling is Tu (1996). Figure 7.1 shows different components that made up an artificial fish. These components are grouped into three main components – graphics display, biomechanical model and reasoning model.

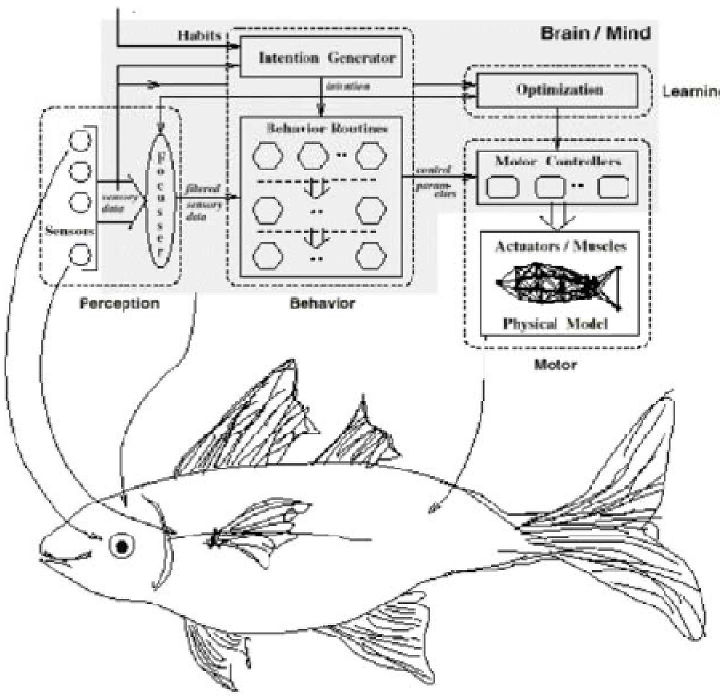


Figure 7.1 Main components, control and information flow in an artificial fish (Tu & Terzopoulos (1994); Tu (1999))

Captured data from the sensor is one of the input processed by intention generator. Intention generator will then send a suitable parameter to the behaviour routine so that appropriate behaviour can be selected. This in turns generates suitable signal to select the right movement. Based on selected movement, motor control actually sends control parameters to the 'muscles' structure in order to create certain movements like turn right or left, swim, glide and others. We will show the fish structure (physics-based model) and the graphics components that form a realistic artificial fish.

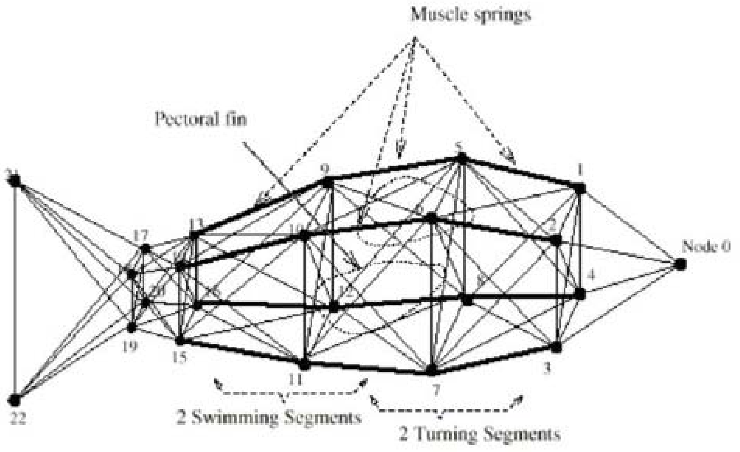


Figure 7.2 Physics-based fish model used by Tu & Terzopoulos (1994); Tu (1999)

Biomechanical Fish Model

A biomechanical model as shown in Figure 7.2 was used by Tu & Terzopoulos (1994) and Tu (1999) involves a physics-based model based on mass-spring system. It contributes to the fish's movements besides giving the general shape of the body. Nodes numbered from 0 to 22 are the mass that were connected by 91 springs that perform as the muscles for the fish.

Graphics Display

This model is a collection of geometric models that are mapped onto the main structure. Real fish image is then mapped onto Bezier surfaces that represent the left and the right side of the fish. Apart from the main body, Bezier surfaces are also used to represent the tails and fins. As the fish swims (or moves), control points will be displaced and the surface is updated accordingly.

Bezier Surface

Two sets of Bezier surfaces are modelled based on their own sets of control points. These control points will be updated as the fish moves, for example transformed points due to changes as the fish swims forward, turns left or turns right (Terzopoulos, Tu & Grzeszczuk (1994), Yu & Terzopoulos (1999), Tu (1999)).

The Cartesian Bezier blending function for the Bezier surface is given as:

$$P(u,v)=\sum_{j=0}^{m}\sum_{k=0}^{n}p_{j,k}BEZ_{j,m}(v)BEZ_{k,n}(u)$$
(1)

where the value of $p_{j,k}$ determines the position of control points $(m+1)$ and $(n+1)$. Patches that were made of Bezier surfaces is combined using suitable boundary constraints to ensure a smooth combination.

3D FISH MODEL

A 3D fish model that is consistent with the biomechanical model described by Tu & Terzopoulos (1994) and Tu (1999) (Figure 7.2) was used as a reference in our experiment. Since our research is focusing on the computer graphics, the following discussions will be more on the modeling and the visualization aspects of the artificial fish.

In accordance to different segments that has been implemented by Tu (1999), we described our model hierarchically. The hierarchy is illustrated as Figure 7.3.

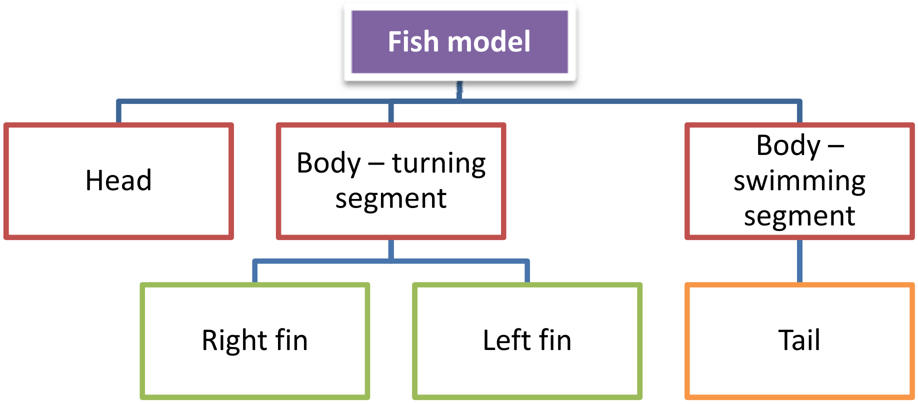
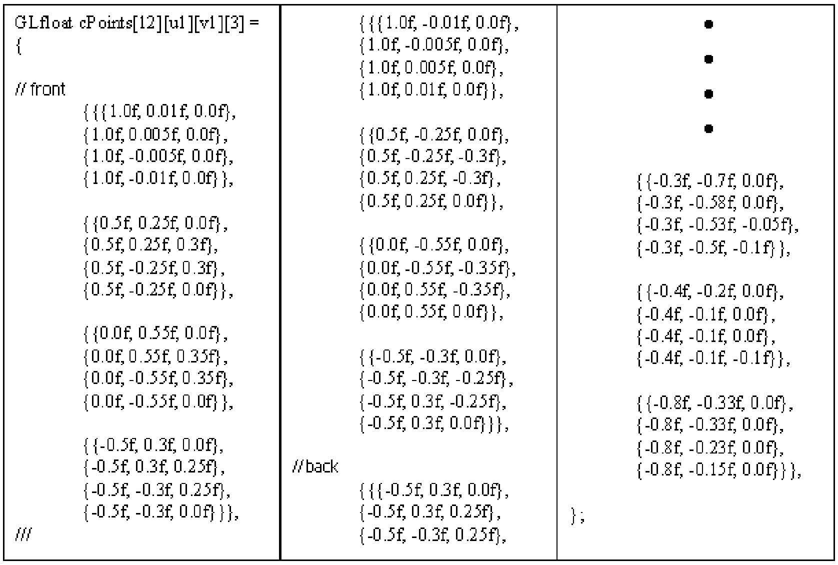


Figure 7.3 Hierarchy of fish model

The head, body, tail and fins are modelled using Bezier surfaces that are generated based on a set of control points using Equation 1. These Bezier patches (left and right patch) are arranged side by side to generate each segments. Every patches for the head, body and tail is based on u x v = 4 x 7, while the fins (which are relatively smaller) are made of u x v = 4 x 4. We used pre-defined control points for our fish model. An algorithm for Bezier surface generation is given as Figure 7.4, while a definition of control points is illustrated as Figure 7.5.

```
Begin
        for i = 0 to n = 0 do
                for j = 0 to m do
                        input control points, $cp_{ij}$
                next j
        next i
// u curve
        for u = 0.0 to 1.0 insteps of 0.05 do
                move_flag = true
                for v = 0.0 to 1.0 insteps of 0.05 do
                        $P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} p_{j,k} BEZ_{j,m}(v) BEZ_{k,n}(u) * CP_{j,k}$
                        if $P(u,v)$ = starting point then
                                move_to( $P(u,v)$ )
                                move_flag = false
                        else
                                draw_to( $P(u,v)$ )end if
                next v
        next u
// v curve        for v = 0.0 to 1.0 insteps of 0.05 do
                move_flag = true
                for u = 0.0 to 1.0 insteps of 0.05 do
                        $P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} p_{j,k} BEZ_{j,m}(v) BEZ_{k,n}(u) * CP_{j,k}$
                        if $P(u,v)$ = starting point then
                                move_to( $P(u,v)$ )
                                move_flag = false
                        else
                                draw_to( $P(u,v)$ )end if
                next u
        next v
        return
end
```

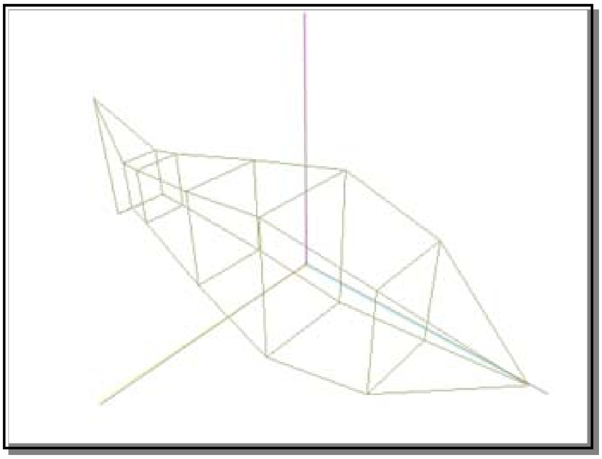
Figure 7.4 Algorithm to generate a Bezier surface



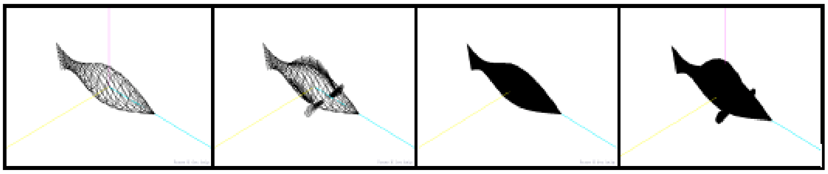
```
GLfloat cPoints[12][u1][v1][3] =
{

// front
        {{{1.0f, 0.01f, 0.0f},
        {1.0f, 0.005f, 0.0f},
        {1.0f, -0.005f, 0.0f},
        {1.0f, -0.01f, 0.0f} },

        {{0.5f, 0.25f, 0.0f},
        {0.5f, 0.25f, 0.3f},
        {0.5f, -0.25f, 0.3f},
        {0.5f, -0.25f, 0.0f} },

        {{0.0f, 0.55f, 0.0f},
        {0.0f, 0.55f, 0.35f},
        {0.0f, -0.55f, 0.35f},
        {0.0f, -0.55f, 0.0f} },

        {{-0.5f, 0.3f, 0.0f},
        {-0.5f, 0.3f, 0.25f},
        {-0.5f, -0.3f, 0.25f},
        {-0.5f, -0.3f, 0.0f} }},
///
        {{{1.0f, -0.01f, 0.0f},
        {1.0f, -0.005f, 0.0f},
        {1.0f, 0.005f, 0.0f},
        {1.0f, 0.01f, 0.0f}},

        {{0.5f, -0.25f, 0.0f},
        {0.5f, -0.25f, -0.3f},
        {0.5f, 0.25f, -0.3f},
        {0.5f, 0.25f, 0.0f}},

        {{0.0f, -0.55f, 0.0f},
        {0.0f, -0.55f, -0.35f},
        {0.0f, 0.55f, -0.35f},
        {0.0f, 0.55f, 0.0f}},

        {{-0.5f, -0.3f, 0.0f},
        {-0.5f, -0.3f, -0.25f},
        {-0.5f, 0.3f, -0.25f},
        {-0.5f, 0.3f, 0.0f}}},

//back
        {{{-0.5f, 0.3f, 0.0f},
        {-0.5f, 0.3f, 0.25f},
        {-0.5f, -0.3f, 0.25f},

        •
        •
        •
        •

        {{-0.3f, -0.7f, 0.0f},
        {-0.3f, -0.58f, 0.0f},
        {-0.3f, -0.53f, -0.05f},
        {-0.3f, -0.5f, -0.1f} },

        {{-0.4f, -0.2f, 0.0f},
        {-0.4f, -0.1f, 0.0f},
        {-0.4f, -0.1f, 0.0f},
        {-0.4f, -0.1f, -0.1f}},

        {{-0.8f, -0.33f, 0.0f},
        {-0.8f, -0.33f, 0.0f},
        {-0.8f, -0.23f, 0.0f},
        {-0.8f, -0.15f, 0.0f}}},

};
```

Figure 7.5 Sample codes to define the control points

Resulting model is shown in Figure 7.6. The upper image (Figure 7.6 (a)) shows the main structure used while the lower images (Figure 7.6 (b)) show the generated Bezier patches that represent the surface of the fish.



(a)



(b)

Figure 7.6 Resulting model (a) basic structure, (b) wireframe and solid models – without and with the fins

ANIMATION OF FISH MODEL

There are a few options that can be adapted to animate the fish model. In the artificial fish model by Tu (1999), a fully physics-based model was implemented. Intention generator activates certain behaviour that in turns, stimulates the motor action and causes the fish's muscles generate movement. Referring to the biomechanical model, movement is caused by contraction and expansion of the spring. It is done harmoniously so that the tail and the swimming segments could swing back and forth. The accumulated force causes the fish to thrust. Pectorial fins (the left and the right fins) are used to stabilize the movement – by controlling the pitch and yaw.

Another option is by using a set of pre-defined path/positions. User will not be able to interact with the fish. We took a moderate approach by applying 3D transformation on the control points to simulate head, fins and tail movement as the fish swims. At the same time users are allowed to control the turning angle of the fish.

Animating the swinging motion of the fish's tail and fins, or the turning of its head involves rotation about certain axis. This involves a series of equation a follows:

$$\left.\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \\ z' &= z \end{aligned}\right\} \text{rotation about the } z-axis\,,$$

$$\left.\begin{aligned} y' &= y\cos\theta - z\sin\theta \\ z' &= y\sin\theta + z\cos\theta \\ x' &= x \end{aligned}\right\} \text{rotation about the } x-axis\,,$$

$$\left.\begin{aligned} z' &= z\cos\theta - x\sin\theta \\ x' &= z\sin\theta + x\cos\theta \\ y' &= y \end{aligned}\right\} \text{rotation about the } y-axis\ .$$

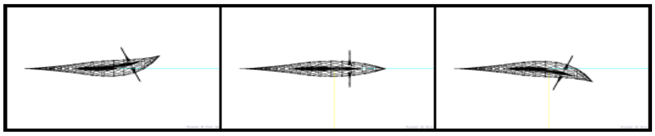
```
void glMoveBody(double degree, int a, int b, int c, int d, int e, int f, int g, int h)
{
            int i;
            theta = degree * pi / 180;                              //rotation degree

            //rotation point
            korZ = mybezier[a].anchors[b][0].z;//c_point[a][b][0][2];
            korX = mybezier[a].anchors[b][0].x;//c_point[a][b][0][0];
            korZ1 = mybezier[c].anchors[d][0].z;//c_point[c][d][3][2];
            korX1 = mybezier[c].anchors[d][0].x;//c_point[c][d][3][0];

            //rotated point
            for(i = 0; i < 4; i++)
            {
                        pZ[i] = mybezier[e].anchors[f][i].z;//c_point[e][f][i][2];
                        pX[i] = mybezier[e].anchors[f][i].x;//c_point[e][f][i][0];
                        pZ1[i] = mybezier[g].anchors[h][i].z;//c_point[g][h][i][2];
                        pX1[i] = mybezier[g].anchors[h][i].x;//c_point[g][h][i][0];
            }
            //rotation
            for(i = 0; i < 4; i++)
            {mybezier[e].anchors[f][i].z /*c_point[e][f][i][2]*/ = (korZ + (((pZ[i] - korZ) *
cos(theta)) - ((pX[i] - korX) * sin(theta))));
                        mybezier[e].anchors[f][i].x /*c_point[e][f][i][0]*/ = (korX + (((pZ[i] -
                                    korZ) * sin(theta)) + ((pX[i] - korX) * cos(theta))));
                        mybezier[g].anchors[h][i].z /*c_point[g][h][i][2]*/ = (korZ1 +
                                    (((pZ1[i] - korZ1) * cos(theta)) - ((pX1[i] - korX1) *
                                    sin(theta))));
                        mybezier[g].anchors[h][i].x /*c_point[g][h][i][0]*/ = (korX1 +
                                    (((pZ1[i] - korZ1) * sin(theta)) + ((pX1[i] - korX1) *
                                    cos(theta))));
            }
}
```

Figure 7.7 Sample code to generate body movements

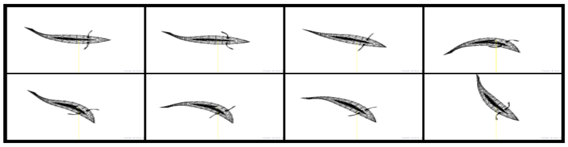
Homogeneous Coordinate System could also be used for calculation. There are also OpenGL functions like ***glRotatef*** (***GLfloat*** *angle,* ***GLfloat*** *x,* ***GLfloat*** *y,* ***GLfloat*** *z*) and ***glTranslatef*** (***GLfloat*** *x,* ***GLfloat*** *y,* ***GLfloat*** *z*) to ease the implementation.

Turning Motion

Most of the fish turns using the front muscles until a maximum level is met. Turning degree is proportional to the curvature of the fish model's turning segment (Webb 1989). Similar to swimming motion, motor control that handles turning actions will generate appropriate parameters so that the fish could turn. An example of the turning action that we managed to produce is given as Figure 7.8 and the sample code for turning motion is given as Figure 7.9.



(a) Turn to left and right



(b) A complete body turn

Figure 7.8 Results on turning motion from our experiment

```
void turn_head(double a)
{
            //head
            glMoveBody (a, 0, 1, 1, 1, 0, 0, 1, 0);
            //top fin
            glMoveBody (a/2, 0, 2, 1, 2, 8, 0, 9, 0);
            //left-right fin
            glMoveBody (a/2, 8, 1, 9, 1, 4, 0, 5, 0);
            glMoveBody (a/2, 8, 1, 9, 1, 6, 0, 7, 0);
            glMoveBody (a/2, 8, 1, 9, 1, 4, 1, 5, 1);
            glMoveBody (a/2, 8, 1, 9, 1, 6, 1, 7, 1);
            glMoveBody (a/2, 8, 1, 9, 1, 4, 2, 5, 2);
            glMoveBody (a/2, 8, 1, 9, 1, 6, 2, 7, 2);
            glMoveBody (a/2, 8, 1, 9, 1, 4, 3, 5, 3);
            glMoveBody (a/2, 8, 1, 9, 1, 6, 3, 7, 3);
}
```

Figure 7.9 Sample code for turning motion

Swimming Motion

Previous researchers like Blake (1983) and Tu & Terzopoulos (1994) recorded real fish movement as it swims. These can be seen from Figure 7.10 and Figure 7.11.

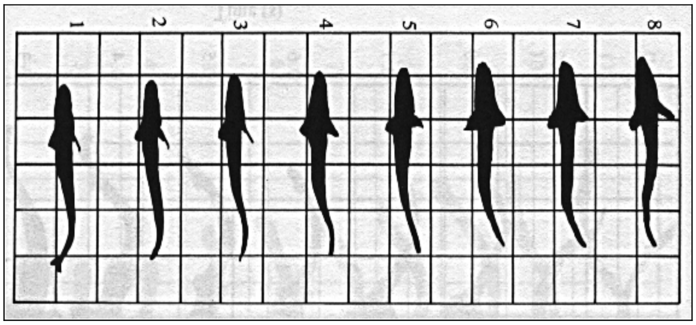


Figure 7.10 Real fish swimming movement (Blake 1983)

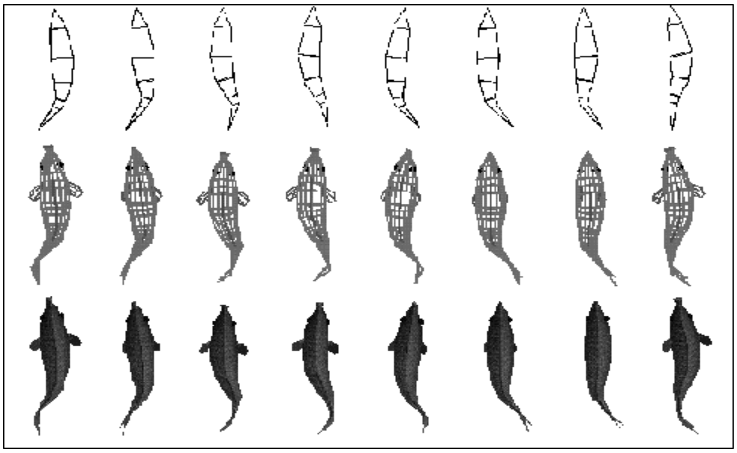
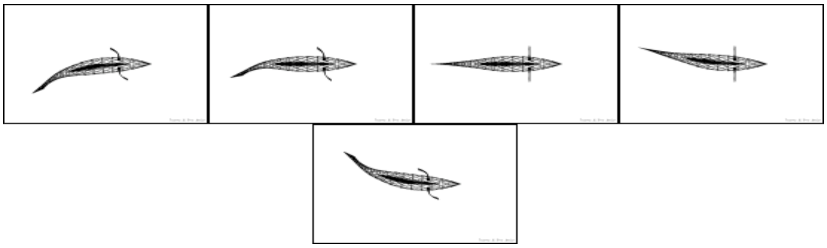
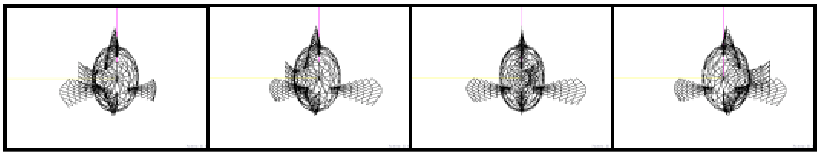


Figure 7.11 Swimming movement by Tu & Terzopoulos (1994)



(a) Top view



(b) Front view

Figure 7.12 Results from our experiments

TEXTURE MAPPING

Texture mapping makes the fish model more realistic. In our experiment, a bitmap file from a real fish photograph was used together with OpenGL function ***glTexCoord2f(…)***. The final result is shown as Figure 7.13.

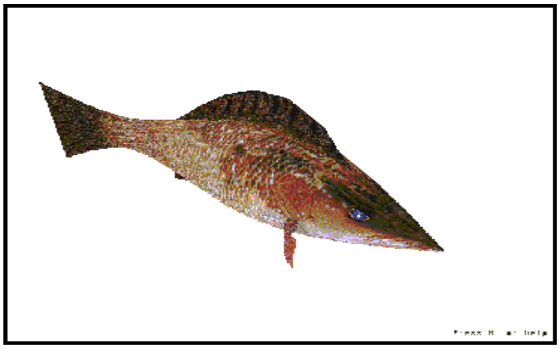


Figure 7.13 A texture-mapped fish

CONCLUSION

Topics on how to create a 3D model of a fish and to generate its movement has been presented in this chapter. A brief overview of an artificial fish that was developed by previous researchers has been discussed. In our research, the modelling and animation aspect of a fish were emphasized. Although physics-based model and motion control component was not involved in this research, results show that the model that we produce could be used as a basis for further research in games and virtual environment applications.

REFERENCE

D. Terzopoulos, X. Tu & R. Grzeszczuk, "Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world," *Journal of Artificial Life,* 1**,** 4, (1994).

P.W. Webb, Form and function in fish swimming. *Scientific American*, 251, pp. 58-68 (1984).

Q. Yu & D. Terzopoulos, "Synthetic motion capture: Implementing an interactive virtual marine world", The Visual Computer 15, pp. 377-394 (1999).

R. W. Blake, Fish Locomotion (Cambridge University Press, 1983).

X. Tu and D. Terzopoulos, "Artificial Fishes: Physics, Locomotion, Perception, Behavior", **Computer Graphics**, SIGGRAPH 94 Conference Proceedings, pp. 43-50, July, 1994

X. Tu, "Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception, and Behavior ", *ACM Distinguished Ph.D Dissertation Series,* (Lecture Notes in Computer Science. Eds.: G. Goos, J. Hartmanis, J.van Leeuwen. Vol. 1635, (Springer-Verlag, 1999)).