# Synthetic Motion Capture for Interactive Virtual Worlds

by

**Qinxin Yu**
余沁欣

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

**Synthetic Motion Capture for Interactive Virtual Worlds**

Qinxin Yu
Master of Science
Graduate Department of Computer Science
University of Toronto
1998

# Abstract

The numerical simulation of biomechanical models enables the behavioral animation of realistic artificial animals in virtual worlds. Unfortunately, even on high-end graphics workstations, the biomechanical simulation approach is at present computationally too demanding for the animation of numerous animals at interactive frame rates. We tackle this problem by replacing biomechanical animal models with fast kinematic replicas that reproduce the locomotion abilities of the original models with reasonable fidelity. Our technique is based on capturing motion data by systematically simulating the biomechanical models. We refer to it as synthetic motion capture, because of the similarity to natural motion capture applied to real animals. We compile the captured motion data into kinematic action repertoires that are sufficiently rich to support elaborate behavioral animation. Synthetic motion capture in conjunction with level-of-detail geometric modeling and object culling during rendering has enabled us to transform a system designed for the realistic, off-line biomechanical/behavioral animation of artificial fishes, into an interactive, stereoscopic, virtual undersea experience.

# Acknowledgements

First of all, I would like to thank my supervisor, Demetri Terzopoulos, for his excellent guidance. His breadth of knowledge and enthusiasm have been a constant inspiration for me. This thesis would not have been possible without his unfailing patience and encouragement.

Michiel van de Panne, my second reviewer, enthusiastically read my thesis on short notice and provided many insightful comments. His suggestions for improving the underlying mathematical formulation of the work were especially helpful.

My great appreciation also goes to Xiaoyuan Tu for making this research possible by providing her artificial fishes software on which our system was developed.

I am forever grateful to Victor Ng-Thow-Hing, Joe Laszlo, and Radek Grzeszczuk, for their invaluable suggestions and assistance.

Special thanks to Mark Deacon and the SMART Toronto organization for providing access to the Trimension/SGI Reality Theater as a development and test facility during Reality 98. Gary Schissler from SGI Canada was among the helpful people who provided technical support in the Reality Theater.

I really appreciate the good times I had with friends at DGP and DCS. They rescued me from the stress and pressure of research. You have made my experience at UofT an enjoyable one.

Finally, thanks to my parents, for their endless love and support. I would not be where I am without you.

# Contents

# List of Figures

# Chapter 1

# Introduction

Computer graphics and especially computer animation has boomed over the last decade, in large part due to the dramatic reduction in computer price/performance. This trend has also prompted an equally dramatic growth in the complexity of graphical models and the quality and speed of rendering. With the emergence of high-performance graphics computers, research in Virtual Reality (VR) has flourished. The applications of VR are many, including science and engineering, entertainment, and education. These applications demand the power of immersive visualization and interactivity.

## 1.1 Motivation

Artificial life (AL) modeling is an exciting new trend in computer graphics [32]. It has yielded impressive animations such as Tu and Terzopoulos' artificial fishes [35]. Our goal in this thesis, will be to further promote the convergence of AL modeling and VR technologies so that observing wild animals in their natural habitats will no longer be the privilege of biologists, nature cinematographers, and amateur adventurists. An ordinary user can enjoy interacting with exotic animals without encountering the danger associated with a "real life" experience. In this thesis, we will investigate the problem of producing animation which synthesizes the intricacy of motion and behavior evident in certain natural ecosystems, at interactive simulation and display rates.

Our work brings us closer to developing engaging virtual environments populated by lifelike characters. To this end, we have been motivated by Tu's work on artificial fishes. Seen in pre-recorded action, these lifelike virtual animals beckon active involvement. One feels compelled ultimately to interact with artificial fishes in their virtual marine environment as scuba divers would interact with the marine life inhabiting a coral reef. On the level of motion synthesis and control, however, the realistic, artificial life modeling of animals typically relies on biomechanical modeling techniques, which unfortunately require intensive numerical computation. In addition to those employed in artificial fishes, physics-based locomotion models also form the basis of Miller's snakes and worms [20], the virtual humans of Hodgins *et al.* [13], and other realistically self-animating characters.

## 1.2 Challenge

Our challenge is to develop a fast derivative of biomechanics based animation that is capable of supporting the interactive animation and rendering of virtual worlds inhabited by numerous lifelike creatures. We would like to achieve this goal without necessarily relying on costly, specialized virtual reality equipment, such as flight simulators [39, 24] or CAVE-like installations [8]. Unfortunately, in addition to the burden of photorealistic rendering, the dynamic simulation of biomechanical animal models of any complexity is usually too compute intensive to run at interactive rates on current desktop or deskside graphics workstations.

## 1.3 Methodology

Our solution is to replace computationally expensive biomechanical animal models with fast kinematic replicas that preserve as much as possible the lifelike appearances, locomotions, and behaviors of the fully dynamic originals. In particular, we capture motion data through the systematic biomechanical simulation of locomotion in the original models. We refer to this technique as *synthetic motion capture* since it is in principle not unlike natural motion capture applied to real animals, particularly human actors. We appropri-

Figure 1.1: We stop the submarine to watch a group of tropical fishes

ately process the recorded data and compile the captured actions into *action repertoires*. The action repertoire implements motion synthesis in a kinematic creature, and it is rich enough to support natural looking locomotion and complex behavior.

As a demonstration of our approach, we have transformed the non-realtime world of artificial fishes presented in [35] into an interactive virtual undersea experience. The user of our system pilots a submarine, navigating in a 3D virtual world populated by lifelike marine fauna (see Figs. 1.1, 1.2, and 1.3). The user may maneuver the submarine into a large school of fishes, chase a fleeing fish, or simply look around and observe colorful marine life unfold. Our interactive virtual marine world is inhabited by 60 artificial fishes of 7 different species. Each fish is an autonomous behavioral agent that interacts with other fishes. Our virtual marine world runs at interactive rates on a deskside graphics workstation and also in a large-scale "Reality Theater".

Figure 1.2: Let's approach the school to take a closer look

Figure 1.3: While on our way up towards the surface, we see a variety of fishes beneath us

## 1.4 Contributions

Our contributions in this thesis are as follows:

1. We have proposed and demonstrated synthetic motion capture, a novel technique that addresses the computational complexity of biomechanical simulation for the purposes of computer animation. The technique prescribes the capture of motion data through the systematic simulation of biomechanical models.

2. We have developed methods for compiling captured motion data into kinematic action repertoires that are sufficiently rich to support elaborate behavioral animation. We have demonstrated our methods with an appropriately modified version of Tu's artificial fishes model.

3. Synthetic motion capture in conjunction with level-of-detail geometric modeling and object culling during rendering has enabled us to transform Tu's system which was designed for the realistic, off-line biomechanical/behavioral animation of artificial fishes, into an interactive, stereoscopic, virtual undersea experience.

4. We have achieved a frame rate of 10–50 frames per second (FPS) for our virtual marine world, versus a frame rate of only 0.25 FPS for the original physically simulated version on the same computer. Excluding rendering, our fish motion synthesis proceeds three orders of magnitude faster compared to the original biomechanical simulation.

Thus, this thesis develops a methodology for building elaborate virtual environments in which a user may interact with realistic artificial animals, i.e., autonomous agents with the ability to act of their own accord in real time, which makes interactions with them interesting and compelling. Through synthetic motion capture of physics-based models, we eliminate the need for specialized motion capture equipment and avoid the necessity for filtering the data to reduce noise. By replacing the computation intensive dynamic simulation with our fully kinematic models, speed is increased dramatically while visual realism is preserved.

## 1.5 Thesis Overview

The remainder of this thesis is divided into 9 chapters. Chapter 2 reviews related work. Chapter 3 describes Tu's artificial fishes [34] which inspired us. Chapter 4 describes how we apply synthetic motion capture to artificial fishes. Tu's biomechanical fish models provided motion data for the action repertoire of our kinematic fishes. Techniques for processing the motion data to produce a functional motor control system are presented in Chapter 5. Chapter 6 discusses how we have adjusted Tu's artificial fish behavioral model to deal with the new, fully kinematic motor system. Chapter 7 reports on how we accelerate rendering by culling objects relative to the view frustum and geometrically modeling visible objects with a suitable level of detail based on their distance from the viewpoint, and how we model the virtual environment in which the artificial fishes reside. In Chapter 8 we describe the user interface that enables a user to maneuver the submarine. Chapter 9 discusses the performance that our approach achieves. Finally, we present our conclusions in Chapter 10 and list possible directions for future research.

# Chapter 2

# Background

In this chapter, we review previous work in computer graphics related to our research. Starting from an introduction to several commonly used techniques in creating computer animations, we proceed to survey current technology in building virtual environments and how some other systems obtain a fast frame rate. We conclude by putting our work in perspective with prior research in computer graphics.

## 2.1 Animation Techniques

Several categories of techniques have been developed to assist in making 3-D character animation. They include keyframing, kinematics, physics-based animation and motion capture.

### 2.1.1 Keyframing

First introduced by Burtnyk and Wein [4] in 1971, keyframing is the simplest technique for animating an object. It allows animators to supply a set of keyframes and automatically generates the intermediate frames, known as the in-betweens. There are generally two ways to produce these in-betweens: image-based and parametric keyframe animations. With image-based keyframe animations, in-betweens are often created using linear interpolation of the keyframe images. With parametric keyframe animations, the keyframes are specified with an appropriate set of parameter values at a given time.

These parameters are interpolated and images are constructed from the interpolated parameters. Splines are often used for this purpose, and many methods have been proposed to improve the quality [5], [27], [16]. Despite these efforts, this method remains tedious, although it offers the animator direct control over the motion.

### 2.1.2 Kinematics

The kinematic approach to animation describes motion with positions, velocities and accelerations without examining the forces that produce the motion. It falls into two categories, forward kinematics and inverse kinematics. Forward kinematics is used to determine positions, velocities and accelerations of all the links in an articulated model given the position, velocity and acceleration of the root of the model and all the transformations between links and their relative velocities and accelerations. It is a necessity for skeletal keyframed animation. Inverse kinematics aids in positioning and orienting a body model which consists of joints and segments [1]. Given an arbitrary chain of joints and a position in space, inverse kinematics finds the joint angles such that the distal end of the chain reaches that position and orientation in space. For example, if we want to have the arm of an actor reach a certain position in space at a certain time, inverse kinematics can be used to solve the velocity and joint angles so that the arm can move according to the specification.

### 2.1.3 Physics-Based Animation

Physics-based animation typically uses dynamics. It differs from kinematics in that it involves masses and forces and the motion that they produce. Because object motions are governed by the laws of physics, the animation produced normally appears more realistic than that with a kinematic approach. The physics-based approach frees the animator from having to worry about low-level details of the motion at the cost of having less control over the movements. It involves a physical simulation that may require solving a system of equations depending on how complex the physical model is.

Physics-based control techniques are used to direct the physical models. They can be classified into two categories: the constraint-based approach and the motion synthesis approach.

### Constraint-based Approach

The constraint-based approach exerts control by imposing kinematic constraints on the motions of an animated object [26]. There are two ways to calculate motions that satisfy constraints: the inverse dynamics technique and the constrained optimization technique.

With inverse dynamics, the motion of an animated body is specified by solving a set of equations of motion. The resulting motions are physically correct, i.e., they obey the Newtonian laws of physics, but they may still look unnatural with respect to any specific form of animal locomotion. For example, the generated motion for a human may resemble that of a robot. This is because an animal's movement must also obey biomechanical principles. Contrary to the assumption of the inverse dynamics technique, a real animal's muscles cannot generate arbitrary forces and torques in order to move along any specific path. This leads to the constrained optimization technique that casts motion control as an optimization problem by defining an objective function. The objective function evaluates the resulting motion, and it balances between the user defined constraints (which may include both hard and soft constraints) and the constraints induced by the laws of physics (which are often treated as soft constraints). However, the compromised solution may not lead to visually realistic motion.

### Motion Synthesis Approach

The motion synthesis approach ([20], [23], [36] and many others) synthesizes the muscles of animals as a set of actuators that are capable of driving the dynamic model of a character to produce locomotion. It guarantees that the laws of physics are never violated while taking the limitations of natural muscles into consideration. This allows the simulation of various locomotion modes found in real animals, but it offers less direct animator control than the constraint-based approach. For instance, it is difficult to produce the motions of an animated character that precisely follows a pre-defined trajectory.

### 2.1.4 Motion Capture

In creating action repertoires for virtual creatures, we were motivated by motion capture techniques. Also known as performance animation, motion capture has been used for computer character animation since the late 1970s, but it has recently become more popular due to hardware improvements. As Maiocchi [18] pointed out:

> *Motion capture is the measurement and recording of the direct actions of an actor for immediate or delayed analysis and playback. The information captured can be as simple as the value given by an input device such as a joystick or as complex as the deformations of the face of a mime. Motion capture for computer animation involves the mapping of such measurements onto the motion of the digital character.*

There are three major types of motion capture systems: mechanical systems, optical motion capture systems and magnetic systems. *Mechanical systems* are based on the technique which connects the parameters controlling the motion of the character to some input devices. These input devices can be as simple as the mouse, the joystick and the keyboard, or they can be more sophisticated such as the data glove using optical fibers. With these systems, the characters can be directly acted, allowing the equivalent of puppeteering. *Optical motion capture systems* obtain the locations for some key points in an animated model by tracking the positions of the retro-reflective markers attached to the key points on an actor's body with high contrast video images. The 3-D position of each key point can be reconstructed using data recorded from several different directions. These systems can track more points without wires in a large active area, but they are sensitive to light and reflections. Since a marker must be seen by at least two cameras (for 3D data), the total or partial occlusion caused by the subject or other markers can result in lost, noisy, displaced, or swapped markers. *Magnetic systems* use the position and orientation of a set of sensors that are capable of measuring their spatial relationship to a centrally located magnetic transmitter. They are much cheaper than the optical systems, but they are sensitive to metal interferences. Physical connections to the control units also make these systems cumbersome. Recent research has lead to a vision based motion capture technique that recovers high degree-of-freedom articulated human body configurations in complex video sequences [2].

The first phase in using a performance animation system is system set-up. The system must be calibrated to be ready for the actual motion capture session. For magnetic systems, the capture area must be free from any potential interference that results in erroneous reading of the data, such as metal sources, magnetic and electrical fields. Then the performance has to be carefully planned. The sensors must be placed on an actor's body in correspondence to the key points one wants to measure. These sensors vary in size from system to system, and many of them require cabling. Both factors impose limitations in the range of movements of the actor. The sensor data can now be recorded at a certain sampling rate that the motion capture system operates. The collected data needs to be filtered properly to reduce any noise that can affect the data value but leave the nuances that characterize the movements intact. While the recording is done in real time, the availability of the data for animating a 3D character may require a non real-time process. The captured raw data is often finessed and further animated using inverse kinematics.

Recent research involves developing techniques to process the captured data. Bruderlin and Williams [3] addressed the problem of reusing and adapting existing data. They implemented a motion editor which allows the user to manipulate the data with various signal processing techniques applicable to animated motion. Rose et al. [30] applied space time constraints to create smooth transitions between segments of human body motion.

## 2.2 Virtual Environments

Researchers and developers apply state-of-the-art computer animation techniques to create various virtual environments. Our virtual marine world is an example of a virtual reality system, which is defined by Manetta and Blade [19] as:

> Virtual Reality: A computer system used to create an artificial world in which the user has the impression of being in that world and with the ability

to navigate through the world and manipulate objects in the world.

Partially derived from the flight simulators that the Air Force and commercial airlines use to train pilots, VR technology has seen rapid development in the last couple of decades. Ivan Sutherland is widely considered the founder of VR technology due to his work on the ultimate display [31] in 1965. Many virtual reality systems have been built since, including the CAVE automatic virtual environment [8], and Disney's Aladdin [25]. In this section, we describe two of these well known VR environments briefly.

### 2.2.1 The Flight Simulator

The flight simulator was one of the driving forces behind the emergence of VR technology. Its history dates back to 1929 when Edwin A. Link patented the first ground-based flight trainer. It has grown more and more sophisticated over time. By 1986, the simulator included helmets, used half-silvered mirrors to create virtual overlays on the physical cockpit, fiberoptics to provide expensive but effective high-resolution visual displays, three-dimensional acoustic systems were developed, and a glove that used piezoelectric vibrotactile actuators was designed.

### 2.2.2 The CAVE

In 1991, Carolina et al. [8] at the University of Illinois developed the CAVE to enhance scientific visualization. The CAVE is a $10' \times 10' \times 10'$ theater consisting of three rear-projection screens for walls and a down-projection screen for the floor. Full color workstation images ($1280 \times 512$ stereo) were projected onto the screens at 120Hz, giving 2,000 to 4,000 linear pixel resolution to the composite image. There was computer-controlled audio and the display was flicker-free. With viewer-centered head-tracking perspective and a large angle of view, interactive control and binocular display, the CAVE is appealing to leading-edge computational scientists for visualization purposes. A user's head and hands were tracked with Polhemus or Ascension tethered electromagnetic sensors.

The great visual effect achieved with the CAVE comes at a high cost. Four Silicon Graphics high-end workstations were used to create the imagery (one for each screen), and they were connected to a fifth for serial communications to input devices and synchronization.

Because the CAVE is designed for the purpose of scientific visualization, the biggest challenge is the display. Hence, most of the effort was concentrated on obtaining as accurate tracking of the user as possible and on providing the correct projection and stereo.

### 2.2.3 Discussion

Currently, output devices for existing virtual environments range from a simple monitor to much more sophisticated and costly projection systems such as a reality room. The content varies from simple scenarios such as Calson and Hodgins' virtual world of one-legged creatures [6], to much more complicated ones such as those seen in military flight simulators. It is often costly to produce a sophisticated virtual environment, such as Disney's Aladdin, since the hardware required to guarantee interactive speed for the simulation and rendering increases tremendously. We propose an approach that brings us closer to developing engaging virtual environments that are elaborate in nature without reliance on costly, specialized equipment.

## 2.3 Real-time Techniques

Researchers and VR system developers have employed various clever techniques to increase the complexity of the models in virtual environments while maintaining realism and fast frame rates.

Carlson and Hodgins [6] used simulation levels of detail for the real-time animation of single-legged hoppers. They built a virtual world where hoppers try to escape from a puck. Three levels of detail were incorporated: rigid-body dynamics, a hybrid kinematic/dynamic model, and a point-mass simulation. They identified the criteria for switching among these levels seamlessly. At 30 frames per second, their system can simulate 8 creatures using dynamics only, or 24 creatures using hybrid simulation only. We too exploit multiple levels of detail in animation, but the intrinsically higher biomechanical complexity of artificial fishes makes it infeasible to maintain an acceptable frame rate using any reasonable dynamic model. Instead, we propose synthetic motion capture as a means of achieving interactive frame rates without excessively compromising the quality of the animation.

Granieri et al. [10] used an off-line production process to record posture graphs for a human model. The recorded motions were played back in real-time to animate human figures in a distributed simulation. A typical distributed simulation contains many simulation hosts, each is responsible for simulating a portion, or subset, of all the objects and processes involved in the simulation. They also used motion levels of detail but concentrated more on procedurally generated motion.

Van de Panne [37] used footprints as basis for generating locomotion for bipedal figures at interactive rates. He employed a planning algorithm that dynamically generates new footprints and the associated new motions to produce autonomous motions.

Lamouret and van de Panne [17] discussed various problems associated with using motion databases to create novel animations. They implemented a prototype system for a planar three-link articulated figure performing hopping behavior. We have addressed some of the same problems outlined in their paper and successfully built a much more elaborate system.

The computer game development industry has also applied various techniques to generate real-time animations for characters in interactive games. Many impressive games have been produced to date, including Doom, Quake, Duke Nukem 3D, etc.

## 2.4 Our Proposed Method

We propose a technique that avoids the expense of dynamic simulation, while preserving the natural looking motions that it can generate. Our technique is based on the use of motion capture data. However, instead of attaching sensors to a real human or animal, we use physics-based modeling as the source of our data. This eliminates the need for specialized motion capture equipment, and also guarantees that the recorded data would be noise-free. We refer to our technique as *synthetic motion capture*.

We build an elaborate, engaging underwater world without relying on specialized motion capture equipment. Like other virtual reality systems, we provide user-centered perspective view into our virtual world, interactive control, and a stereoscopic display.

# Chapter 3

# Artificial Fishes

The system described in this thesis is a real-time version of the artificial fishes created by Tu [34]. This chapter presents Tu's animation system. The artificial fish model consists of three submodels: a graphical display model, a biomechanical model, and a brain model.

## 3.1 Biomechanical Model

The biomechanical model captures some of the physical structures of the fish's body. It is also responsible for locomotion. Figure 3.1 shows a system overview of an artificial fish. The motor system adopts the motion synthesis approach of physics-based modeling to simulate motion. Each fish is approximated by a mass-spring-damper model consisting of 23 nodal point masses and 91 spring-damper units as illustrated in Figure 4.1. Each spring-damper pair forms a uniaxial Voigt viscoelastic unit that approximates the viscoelasticity properties of biological tissue. Applying Hooke's law and Newtonian mechanics, a system of intrinsically stiff second-order differential equations is established to govern the motion. This system of $69 \times 69$ equations must be integrated over time to simulate the dynamics of the fish model. A numerically stable, semi-implicit Euler method is employed to solve the system. The biomechanical model is also equipped with nine motor controllers responsible for the detailed muscle actions and pectoral fin motions required for locomotion:

1. swim MC — produces forward caudal locomotion

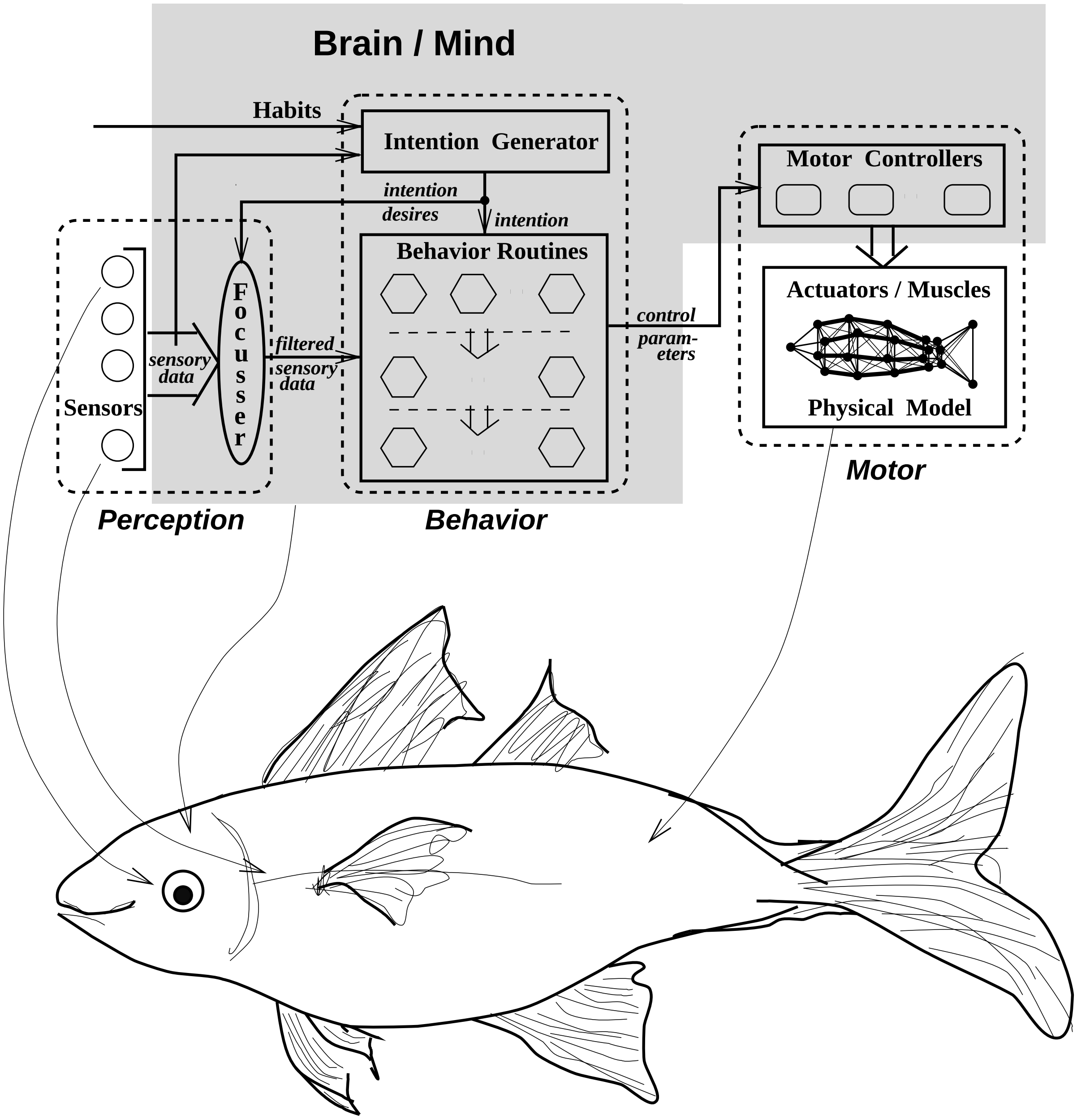



Figure 3.1: System overview of the artificial fish

2. left-turn MC — executes a left turn
3. right-turn MC — executes a right turn
4. glide MC — provides a smooth transition from forward swimming to turning and vice versa
5. ascend MC — lifts towards the surface
6. descend MC — dives towards the seabed
7. balance MC — maintains the balance of the body
8. brake MC — slows the forward velocity
9. backward MC — retreats

These muscle motor controllers execute the designated motor functions by controlling the contractions of the muscles. A set of control parameters are used to coordinate the motion. The pectoral fins on most fish control pitching (up-and-down motion of the body), yawing (the side-to-side motion) and rolling. For our purposes, the detailed movements of the pectoral fins are less important than the motion of the fish's body. Therefore, the pectoral oaring motions are simulated kinematically while the dynamic forces that the pectoral fins exert on the body of the fish are approximated.

### 3.1.1 Graphical Display Model

The graphical display model captures the form and appearance of any specific real fish. The geometric fish models were constructed manually using the Alias$^{TM}$ 3D Modeler. Two juxtaposed B-spline surfaces, one for the left half and the other for the right half of the fish body are employed. B-spline surfaces are also used to represent the dorsal and ventral fins, while the pectoral fins are modeled as polygonal surfaces. After obtaining the texture coordinates with the aid of active deformable contours (popularly known as "snakes") introduced by Kass, Witkin and Terzopoulos [14], the fishes are represented by texture-mapped geometric body models. The control points of the geometric surface model are then associated with the faces of the dynamic model to couple the two together.

### 3.1.2 Brain Model

The brain model is responsible for motor control, perception control and behavior control. It consists of three control centers: the motor center which was described in section 3.1, the perception center, and the behavior center.

#### Perception Center

Realistic perception modeling is as important as locomotion modeling. It is essential to model the perceptual capabilities of the animals as well as the limitations of natural perception. Moreover, the animals must also have some mechanism to filter through a relatively large amount of sensory information to focus on a particular set of information at any particular time. This process is often referred to as attention.

As illustrated in Figure 3.1, the perception system of a fish comprises of a set of virtual sensors and a perceptual focusser. Each fish is equipped with two sensors, a temperature sensor that measures the virtual water temperature and a vision sensor.

The vision sensor enables a fish to detect objects in the environment that are relevant to its survival, such as food, predators and mates. The cyclopean vision sensor is limited to a 300 degree spherical angle extending to a radius, which reflects the visibility of the translucent water. The spherical angle and the visual range define a view volume within which objects can be seen. The vision sensor can interrogate the world model database to identify objects in sight and interrogate the physical simulation to obtain information such as the position and velocity of objects of interest.

The task of the focusser is to extract the appropriate aspects of the world that is relevant to the particular tasks with which the animal is concerned. To obtain the focus of attention, the focusser follows the intention generated by the intention generator in the artificial fish's brain. For example, if the intention is to avoid collision, the focusser is activated to locate the position of the most dangerous obstacle.

### Behavior Center

Comprised of the habits and mental state of the fish, an intention generator, and a set of behavior routines, the behavior system (Figure 3.1) controls action selection. At each time step, the intention generator issues an intention based on the fish's habits, mental state, and incoming sensory information. The corresponding behavior routine is then selected and executed, which in turn causes the appropriate motor controllers to run.

### Action Selection Mechanisms

Priorities must be established among different behaviors. For example, avoiding life-threatening situations such as escaping from a predator should always take precedence over other behaviors. The intention generator is therefore based on a priority hierarchy of behaviors. Also some persistence in behavior should be evident. This is achieved by giving each fish a single-item short term memory that is used to store the current intention and some associated information that may be used to resume a behavior when it is interrupted.

### Behavior Routines

There are nine behavior routines listed below:

1. avoiding-static-obstacle
2. avoiding-fish
3. chasing-target
4. eating-food
5. mating
6. leaving
7. wandering
8. escaping

9. schooling

Each behavior routine uses the focused perceptual data to select appropriate motor controllers and provide them with the proper motor control parameters.

## 3.2 Artificial Fish Types

Three types of artificial fishes were implemented in the original system: predators, prey, and pacifists. The predators are hunters and are not preyed upon by other predators. The prey fish form a school when a predator is far and the school should scatter when the predator gets closer. The pacifists are indifferent to predators and their primary intention is to eat and mate.

## 3.3 Modeling the Marine Environment

Physics-based animate models of seaweeds and plankton have been created to enhance visual realism. The virtual water is translucent and it is rendered with a bluish fog effect. The water current is simulated as a simple fluid flow field, while the dynamic seaweeds respond to simulated water currents. Edible plankton are modeled as floating mass points under the influence of the water current.

## 3.4 Discussion

In our implementation of artificial fishes, described in detail in the remaining chapters of this dissertation, we replace the motor system of the original artificial fish with a kinematic action repertoire complied using our synthetic motion capture approach. Some of the behavior routines in the original artificial fish brain model must be adjusted to accommodate the somewhat weakened, fully kinematic motor system. Furthermore, we also modify the graphical display model to reduce the rendering time.

# Chapter 4

# Compiling Action Repertoires

In this chapter, we explain the application of synthetic motion capture to artificial fishes. The artificial fish model is described in [35]. Additional details are available in [34].

The original biomechanical fish model is a dynamic mass-spring-damper system consisting of 23 nodal point masses as illustrated in Figure 4.1. Under the action of the 12 contractile muscle springs, at each time frame $t$ the numerical simulator performs an implicit time integration of the system of 69 coupled, second-order ordinary differential equations of motion that govern the biomechanical model. This involves first computing the external hydrodynamic forces at time $t$, then solving a sparse $69 \times 69$ system of linear algebraic equations for the 23 nodal velocities at $t + \Delta t$, and finally integrating explicitly in time to obtain the 23 nodal positions $\mathbf{n}_i$, for $i = 0, \ldots, 22$, at the next time frame $t + \Delta t$.

## 4.1 Motion Data Capture and Processing

To eliminate this computationally intensive numerical simulation, we capture and compile into action repertoires the nodal positions computed over sequences of time frames.

The numerical simulator computes nodal positions $\mathbf{n}_i$ with respect to a fixed world coordinate system. To compile an action repertoire and facilitate multiple level-of-detail modeling, we express these nodal positions with respect to a body-centered coordinate system $\mathbf{B}$, illustrated in Figure 4.2, that translates and rotates in accordance with the

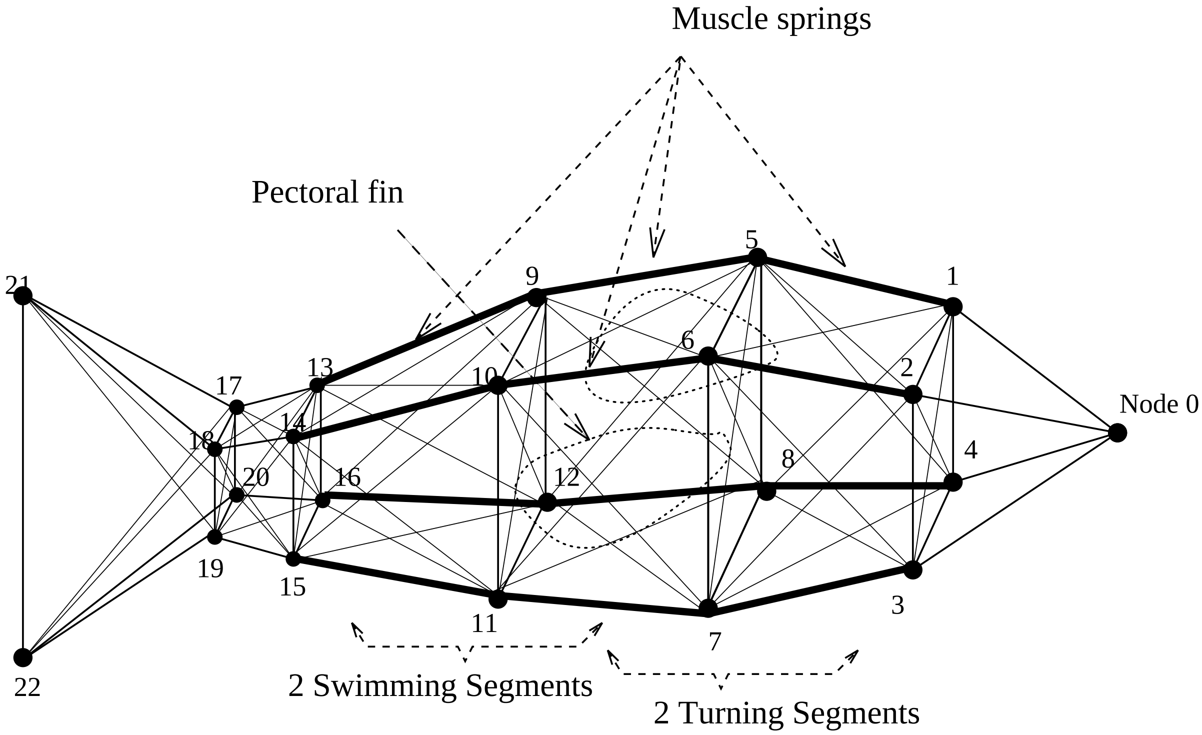



Figure 4.1: The artificial fish biomechanical model

dynamic fish model. At each time frame, we record the incremental translation (i.e., the change in position) and rotation (i.e., the change in orientation) of $\mathbf{B}$, as well as the "body deformation", or the nodal positions with respect to this body coordinate system.

Referring to Figure 4.2, the origin $\mathbf{o} = [o_1\ o_1\ o_3]^T$ (center point of the fish) and the three unit vectors that define the body coordinate system $\mathbf{B}$ are computed as follows:

$$\mathbf{o} = \frac{1}{2}(\mathbf{n}_5 + \mathbf{n}_7) \tag{4.1}$$

$$\mathbf{x} = \frac{\mathbf{n}_0 - \mathbf{o}}{\|\mathbf{n}_0 - \mathbf{o}\|} \tag{4.2}$$

$$\mathbf{y} = \mathbf{x} \times \frac{\mathbf{n}_5 - \mathbf{n}_6}{\|\mathbf{n}_5 - \mathbf{n}_6\|} \tag{4.3}$$

$$\mathbf{z} = \mathbf{x} \times \mathbf{y}. \tag{4.4}$$

The $\mathbf{x} = [x_1\ x_2\ x_3]^T$ axis points to the anterior of the fish, the $\mathbf{y} = [y_1\ y_2\ y_3]^T$ axis points in the dorsal direction, and the axis $\mathbf{z} = [z_1\ z_2\ z_3]^T$ points in the right lateral direction.

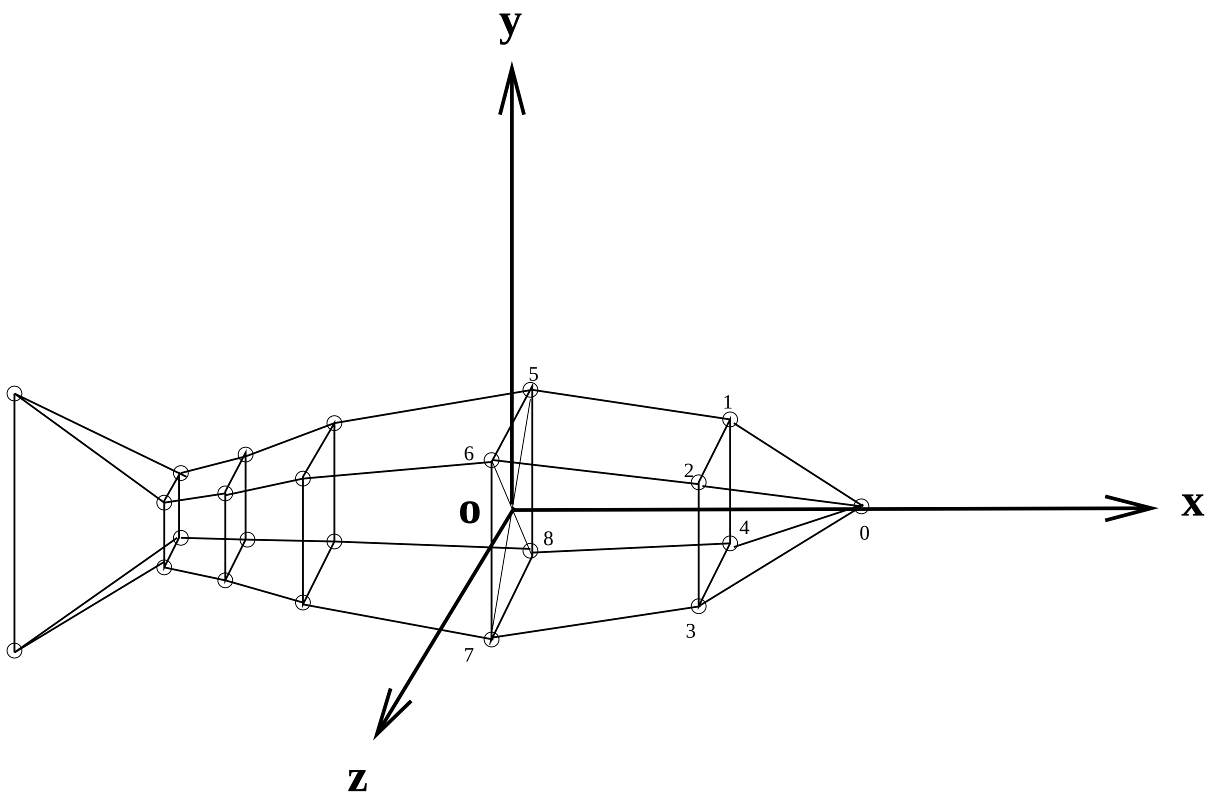



Figure 4.2: The body coordinate system of a fish

This body coordinate system can be represented by the homogeneous matrix

$$\mathbf{B} = \begin{bmatrix} x_1 & y_1 & z_1 & o_1 \\ x_2 & y_2 & z_2 & o_2 \\ x_3 & y_3 & z_3 & o_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.5}$$

within which the upper-left $3 \times 3$ submatrix $\mathbf{R} = \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} \end{bmatrix}$ indicates the rotation required to transform a point in the body coordinate system to a point in world coordinate system.

At each time frame $t$, we record the change in orientation and position. The orientation change is recorded in the form of a $3 \times 3$ rotation matrix $\mathbf{M}^t$ that transforms $\mathbf{R}^t$ into $\mathbf{R}^{t+\Delta t}$:

$$\mathbf{M}^t = \mathbf{R}^{t+\Delta t}(\mathbf{R}^t)^{-1}, \tag{4.6}$$

where $(\mathbf{R}^t)^{-1} = (\mathbf{R}^t)^T$, since it is an orthonormal matrix. This rotation matrix $\mathbf{M}^t$ captures the three orientation degrees of freedom. The change of position is recorded as the translation of the center point with respect to the orientation of the body coordinate system:

$$\mathbf{t}^t = (\mathbf{R}^t)^{-1}(\mathbf{o}^{t+\Delta t} - \mathbf{o}^t). \tag{4.7}$$

Let $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{22}$ denote the deformation data, where $\mathbf{d}_i$ is a vector indicating the position for the $i$th node in the fish model. The deformation data are recorded with respect to the body coordinate system as follows:

$$\mathbf{d}_i^t = (\mathbf{R}^{t+\Delta t})^{-1}(\mathbf{n}_i^{t+\Delta t} - \mathbf{o}^{t+\Delta t}), \quad i = 0, 1, \ldots, 22. \tag{4.8}$$

Figures 4.3 and 4.4 show examples of recorded locomotion segments, a forward swim segment and a right turn segment, respectively. For each frame, the trajectory of the local $\mathbf{x}$ and $\mathbf{z}$ axis shows (in top view) the evolving sequence of position and orientation changes up to the current frame, while the fish body shows the deformation relative to the body coordinate system in the current frame.

The artificial fish geometric display model uses B-spline surfaces. To display the original artificial fish, the B-spline control points are computed relatively inexpensively

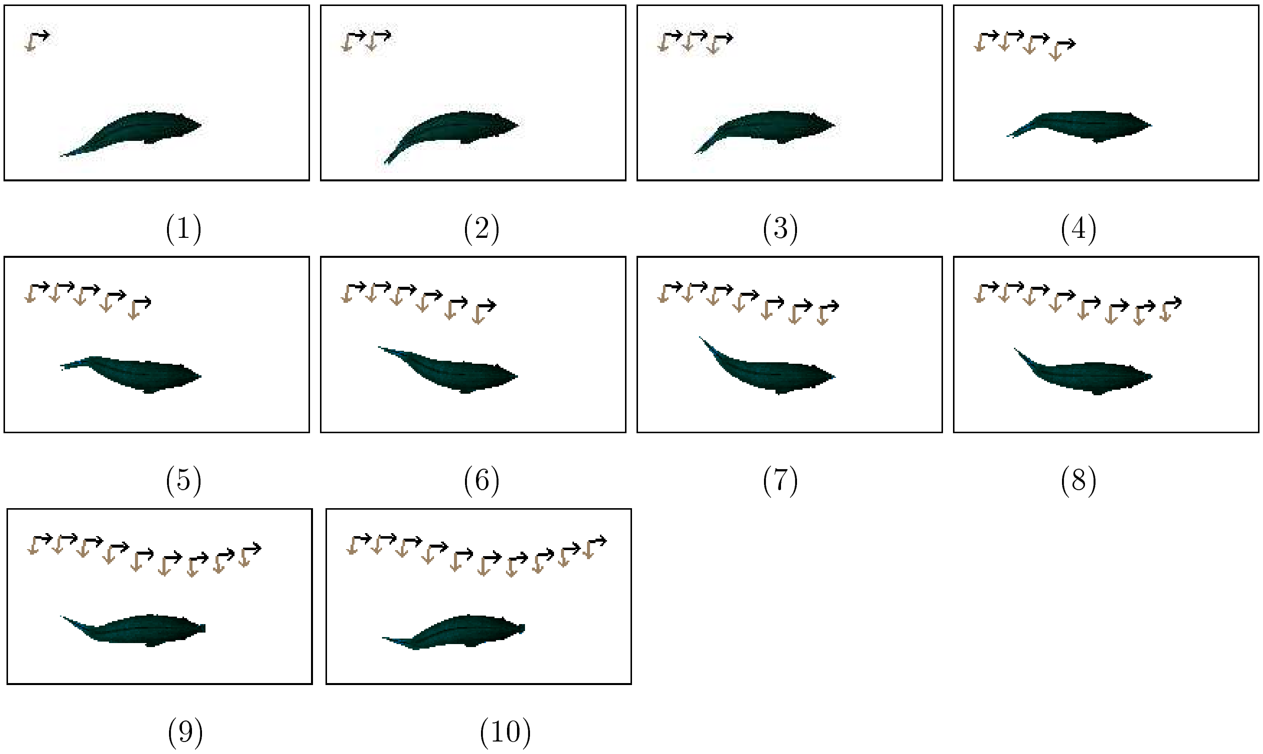
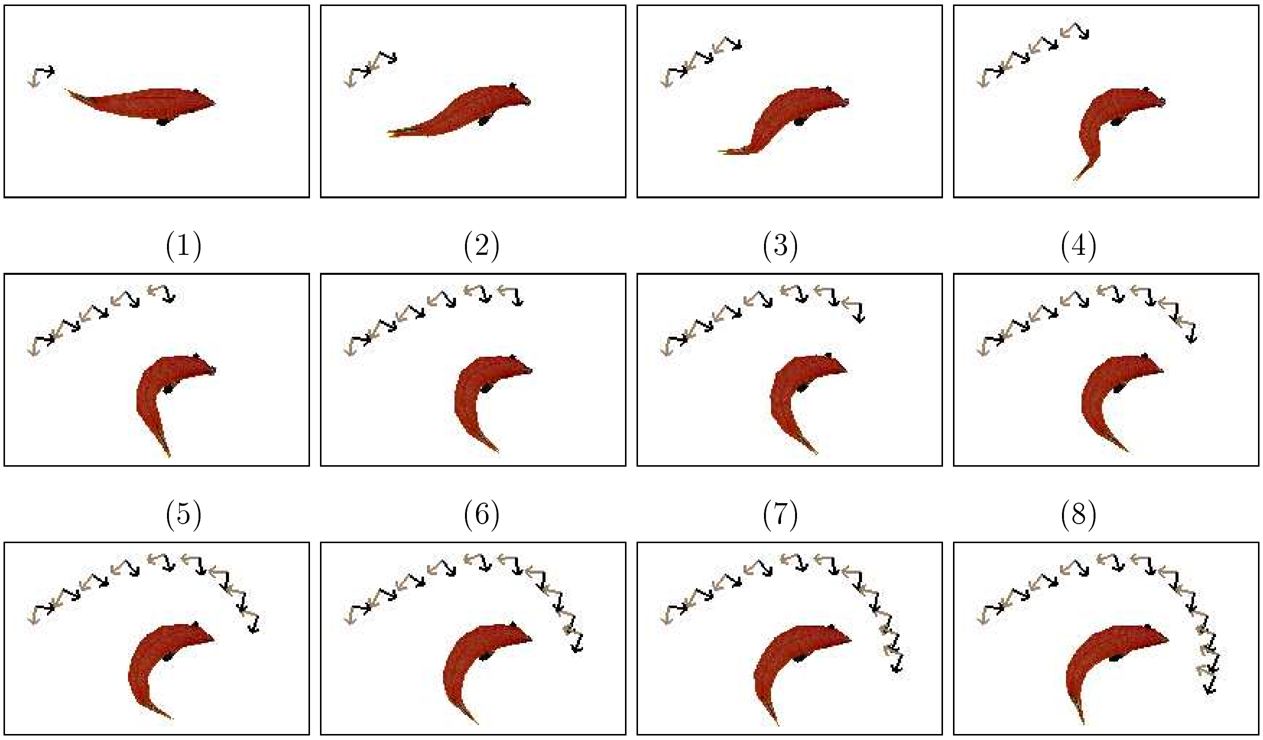


Figure 4.3: A forward swim action segment

from the nodal positions of the biomechanical model. We have the option of explicitly storing control points as part of the synthetic motion capture process. Since there are many more control points (426) than there are nodes (23), there is a tradeoff between the memory required to store control points explicitly versus the time required to compute them on the fly from the nodal points. As the storage requirements grow, memory paging is exacerbated, tending to slow down the animation. Thus, the recording of control points for a particular species of fish can be justified only when its population in the animation is substantial. In our marine world, only the schooling fish, whose population totals 51, satisfies this condition. We record the control points $\mathbf{S}$ in body system coordinates for the schooling fish as

$$\mathbf{s}_i^t = (\mathbf{R}^{t+\Delta t})^{-1}(\mathbf{c}_i^{t+\Delta t} - \mathbf{o}^{t+\Delta t}), \quad i = 0, 1, \ldots, 425, \tag{4.9}$$

where the $\mathbf{c}_i$ denote the control point positions in global world coordinates.



(1) (2) (3) (4)
(5) (6) (7) (8)
(9) (10) (11) (12)

Figure 4.4: A right turn action segment

## 4.2 Pattern Abstraction

The dynamic simulation of the physics-based artificial fishes uses 9 motor controllers to generate coordinated muscle actions [35, 34]. The 9 basic swimming patterns—forward swimming, left turning, right turning, gliding, ascending, descending, balancing, braking, and retreating—can be combined to synthesize continuous locomotion. To reduce the size of the action repertoire, the following considerations lead us to abstract from these a smaller set of fundamental motion types.

Gliding serves as a transition action between forward swimming and turning. For example, it is used to switch smoothly from swimming forward to turning left, or from turning left to turning right, etc. Significant storage space is required for this action to be stored in the repertoire, since it assumes different forms in different transitions. Fortunately, motion warping [38] serves as a good replacement for gliding, and it requires almost no storage. Ascending and descending can be easily represented by a forward swim heading upwards and downwards respectively. The balancing action helps a fish to maintain its balance, so it does not go belly-up, and it must be done with care when we generate animations. Sections 5.2 and 5.3 will describe gliding and balancing in more detail. Braking slows the forward velocity and it can be approximated by a forward swim with a negative acceleration. Similarly, retreating can be accomplished by a forward swim with a negative velocity. The pectoral fin movements that play a vital role in animating a life-like swimming fish continue to be computed using a kinematic model as in the original system.

Hence, the action repertoires consist of three fundamental motion types—forward swimming, left turning, and right turning. Furthermore, fishes from different species exhibit different muscle actions for any given swim pattern because of differences in body shapes and mass distributions. As a result, we must compile a different action repertoire for each species. Fish in the same species may vary in size, but they display similar muscle actions for the same swim pattern. In this case, we scale the stored data to accommodate the variability in size. Consequently, the action repertoires contain data for three swim patterns for each species.

## 4.3 Action Segments

Our goal is to select the minimal set of action segments that can best represent each swim pattern. Since extended swimming motion is cyclic in nature, we record one cycle for each selected segment. For a forward swim, the locomotion speed is approximately proportional to the contraction amplitudes and frequencies of the muscles, and a fish can swim no faster than a certain top speed. Hence, we select three segments with three speeds—slow, medium and fast—which serve adequately to approximate the full range of possible speeds. Figure 4.3 shows a sample forward swim segment.

The turn angle of a fish is also proportional to the contraction amplitudes and frequencies of the muscles. We categorize turns into sharp and gradual turns. Hence, segments are chosen according to these categories for each species. Figure 4.4 shows a right turn segment being recorded.

For a forward swim, a cycle begins with the tail swinging to one side of the body and ends with it swinging to the other. Since it is possible for the fish to switch to a forward swim with its tail on either side, we marked two entry points for each segment. The entry points are:

1. Left - the tail is on the left side of the fish's body and begins to swing towards the right.

2. Right - the tail is on the right side of the fish's body and begins to swing towards the left.

For a left turn, the tail stays mostly on the left side, and for a right turn, the tail stays mostly on the right side. However, a fish may decide to begin making a turn when its tail is on either side of its body. Hence, for each turn pattern, we recorded two segments, beginning with the tail on either side of the body.

To summarize, for each species, we recorded three segments for a forward swim and four segments, including two for each pattern (gradual or sharp) for each of the two turning motions.

## 4.4 Recording

Within the physical simulation, each fish acts as an intelligent agent, reacting to its environment and interacting with other fishes. The possible actions encompass a rich set of movements from which we can select to form an action repertoire.

To perform the actual recording, we picked a fish from each species, which has a medium size as a representative of that species. We then monitored these representatives and selected the segments that satisfy the conditions outlined in Section 4.3. Once this step was completed, the physical simulation was put into action and data were recorded for these selected segments as was explained in Section 4.1.

# Chapter 5

# Motor System

Each fish navigates around the virtual world autonomously and its motor system is responsible for locomotion. In this chapter, we describe how we use the action repertoire to synthesize fast, kinematic locomotion.

## 5.1 Action Reconstruction

When we generate a new animation, the recorded data must be adapted to the current position and orientation of a fish. At any time step $t$, we need to determine the nodal positions based on the data stored in the repertoire. First, we apply the orientation and position changes to establish where the fish should be by updating the body coordinate system as follows:

$$\mathbf{R}^t = \mathbf{M}^{t-\Delta t}\mathbf{R}^{t-\Delta t}, \tag{5.1}$$

and

$$\mathbf{o}^t = \mathbf{o}^{t-\Delta t} + \mathbf{R}^{t-\Delta t}\mathbf{t}^{t-\Delta t}. \tag{5.2}$$

The nodal positions can then be computed according to the deformation data:

$$\mathbf{n}_i^t = \mathbf{o}^t + \mathbf{R}^t(\alpha \mathbf{d}_i^{t-\Delta t}), \quad i = 0, 1, \dots, 22, \tag{5.3}$$

where $\alpha$ is a scaling factor for applying the motion data to fish of the same species, but of varying sizes, and it is computed as the ratio of the size of the animated fish to the

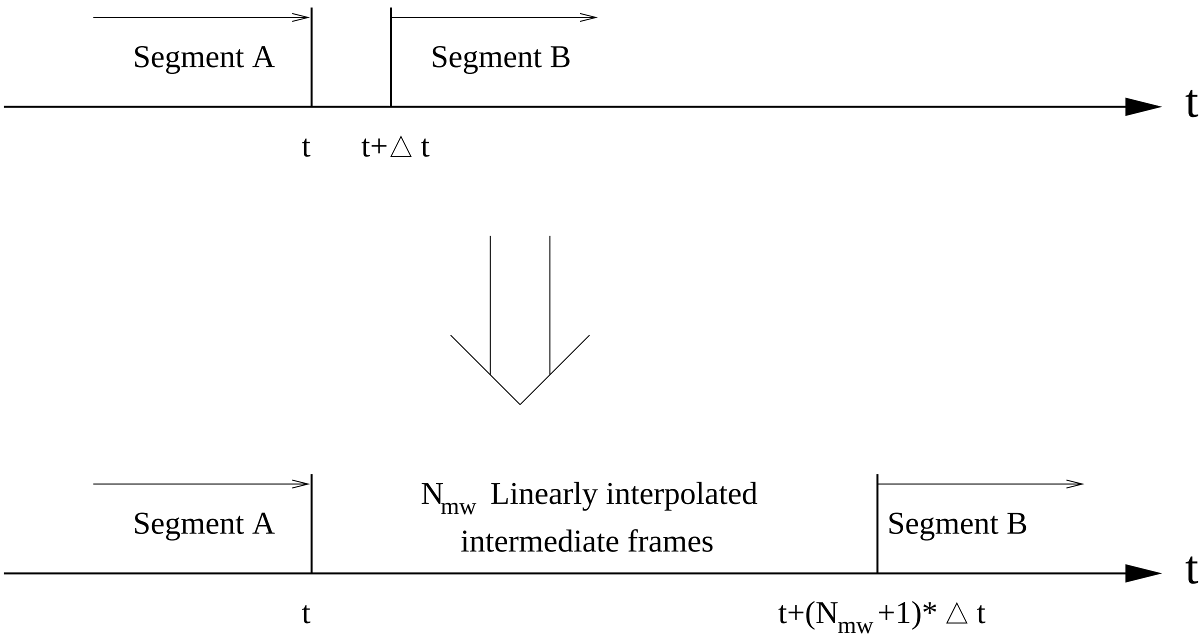



Figure 5.1: Gliding

size of the recorded representative fish. For the schooling fish, the control points can be restored similarly as:

$$\mathbf{c}_i^t = \mathbf{o}^t + \mathbf{R}^t(\alpha \mathbf{s}_i^{t-\Delta t}), \quad i = 0, 1, \ldots, 425. \tag{5.4}$$

## 5.2 Gliding

We replace the gliding action with motion warping [38] where the nodal points serve as correspondence points. It turns out that a linear blend over time is sufficient to provide a smooth transition between swimming patterns. Since the fish's tail usually undergoes the largest deformation, the distance that node 22 (refer to Figure 4.1) travels between the last frame of a data segment to the first frame of the subsequent segment is used to determine how many linearly interpolated intermediate frames are necessary to produce a seamless transition, by setting a maximum distance that a correspondence point can move in a simulation time step. The algorithm (Figure 5.1) is presented as follows:

Step 1. Determine the position for node 22 for the next time frame (i.e., the first frame of the next swim segment), $\mathbf{n}_{22}^{t+\Delta t}$ as shown in equation (5.3).

Step 2. Let $M_{max} = 2.0$ units in the world coordinate system be the maximum movement that can be taken by a correspondence point during the linear interpolation, the number of frames required to complete the transition is given by:

$$N_{mw} = \frac{\| \mathbf{n}_{22}^{t+\Delta t} - \mathbf{n}_{22}^{t} \|}{M_{max}}.$$

If $N_{mw} > 1$, then do

Step 3. Determine the nodal position changes $\mathbf{w}$ for each frame during motion warping as:

$$\mathbf{w}_i = \frac{\| \mathbf{n}_i^{t+\Delta t} - \mathbf{n}_i^{t} \|}{N_{mw}}, \quad i = 0, 1, \ldots, 22.$$

Since the orientation change occurring in any two consecutive frames is small, a linear interpolation of the nodal positions is sufficient to complete the transition. In a more general case when a large change in orientation is possible, an interpolation of the orientation is also required.

Step 4. Stretch the animation by $N_{mw}$ frames, and for each of the $N_{mw}$ intermediate frames, the nodal positions are determined by:

$$\mathbf{n}_i^{t+\Delta t} = \mathbf{n}_i^{t} + \mathbf{w}_i,$$

and the body coordinate system is established using equations (4.1)–(4.4).

Steps 1 to 3 are executed once when joining two segments belonging to different swim patterns. Step 4 is performed for each motion warping frame. We are essentially inserting $N_{mw}$ gliding frames between the last and the first frame of the two adjoining segments.

## 5.3 Balancing

The continuous application of rotation to the body coordinate system employed by motion synthesis may introduce artifacts that occasionally cause the fish to roll on its side. A compensatory rolling motion is needed to maintain the fish's balance, except when the fish is heading vertically up or down, in which case its back can point in any direction. We determine the necessary roll angle, defined as the angle of rotation about the fish's body-centered x-axis such that the projection of the y-axis in the vertically up direction is maximal, and slowly roll the fish to an upright posture.

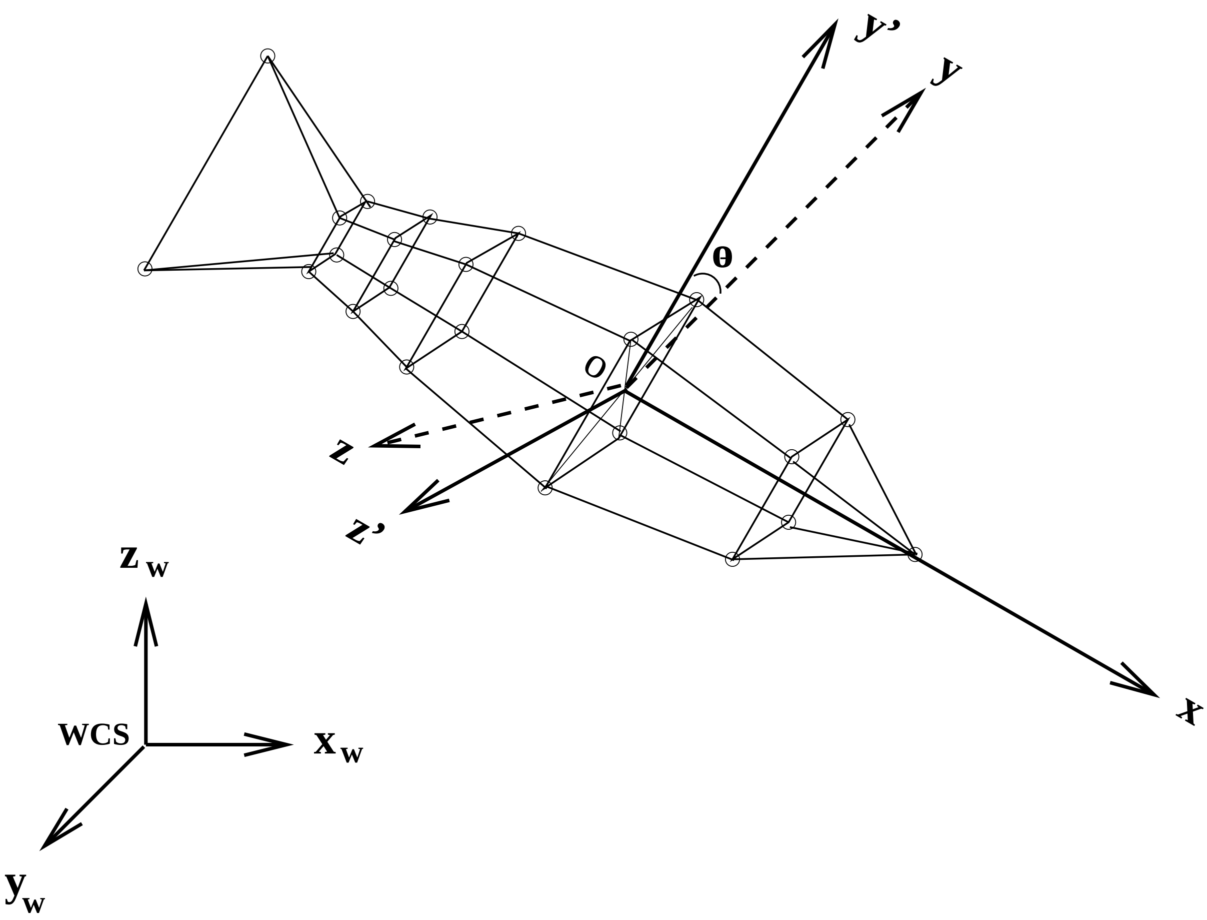



Figure 5.2: The roll angle

Let $\mathbf{y}'$ and $\mathbf{z}'$ denote the y and z axes for the body coordinate system corresponding to the upright posture, the roll angle $\theta$ is given by:

$$\theta = cos^{-1}(\mathbf{y}' \cdot \mathbf{y}),$$

and $\mathbf{y}'$ is determined as:

$$\mathbf{z}' = \mathbf{x} \times \mathbf{z}_w,$$

$$\mathbf{y}' = \mathbf{z}' \times \mathbf{x},$$

where $\mathbf{z}_w$ is the basis vector for the z-axis in the world coordinate system, which always point vertically up.

This adjustment is done as part of the orientation update; therefore, the nodal position reconstruction will be done with respect to this modified body coordinate system.

## 5.4 Level-of-Detail Animation

The use of level-of-detail techniques has proven effective in reducing processing time in various simulation applications. To speed up our simulation further, we apply level-of-detail to the motor system of the artificial fishes.

### 5.4.1 Level-of-Detail Model

The animation may be simplified significantly when the fish is not in view. In this case, it is unnecessary to render the graphical display model, hence we suppress the reconstruction of the nodal positions (5.3) and the B-spline surface control points (5.4). Motion warping is also disabled. We only update the body-centered coordinate system of an invisible fish using equations (5.1) and (5.2). Here, we still perform the balance adjustment to $\mathbf{y}$ because disabling this process may cause the fish to roll away from its upright posture. As the fish re-enters the view volume, suddenly re-enabling the balance mechanism, causes an awkward motion which, albeit transient, is plainly evident, as the fish takes several frames to regain its balance. Table 5.1 summarizes the two levels of detail that are employed in our simulations.

| Computation Process | Visible | Invisible |
|---|---|---|
| Orientation and Position Changes Recovery (update the Body Coordinate System) | Yes | Yes |
| Deformation Data Recovery (update nodal positions) | Yes | No |
| Control Points Computation/Restoration | Yes | No |
| Gliding | Yes | No |
| Balancing | Yes | Yes |

Table 5.1: Simulation Levels of Detail

### 5.4.2 Visibility Check

An object is considered visible if part of it is within the view volume and it is not completely occluded by another object. Fishes are vigorous animals and they move constantly, so in our system, occlusion tests tend to be unbeneficial. Even if a fish is blocked by another fish passing by, since both fishes are in motion, the occlusion will most likely soon be cleared. Hence, we reduce our visibility check to a clipping test, i.e., to examine if a fish is inside the view volume.

A commonly used test for clipping is to employ a bounding box. Each fish can be approximated by a rectangular box so that the clipping test will be simplified to a test of 8 points corresponding to the 8 corners of the bounding box. We reduce the cost of the visibility check even further by representing each fish with its center point, and hence reduce the visibility check to a simple point clipping test. However, this simplification introduces a problem. It becomes possible that a fish would be considered invisible when its center point $\mathbf{o}$ is outside the boundary of the view volume, but part of its body is still inside (Figure 5.3). To resolve this, we cull the center point of a fish against an enlarged view volume that surrounds the actual one (Figures 5.3 and 5.4). This extended view volume includes enough space around the boundary of the actual view volume to allow the partially visible fishes with their center points outside the actual view volume to be considered as visible, yet the extra space is small enough so that not too many extra fishes will be considered visible when they are not. We enlarge the view volume by

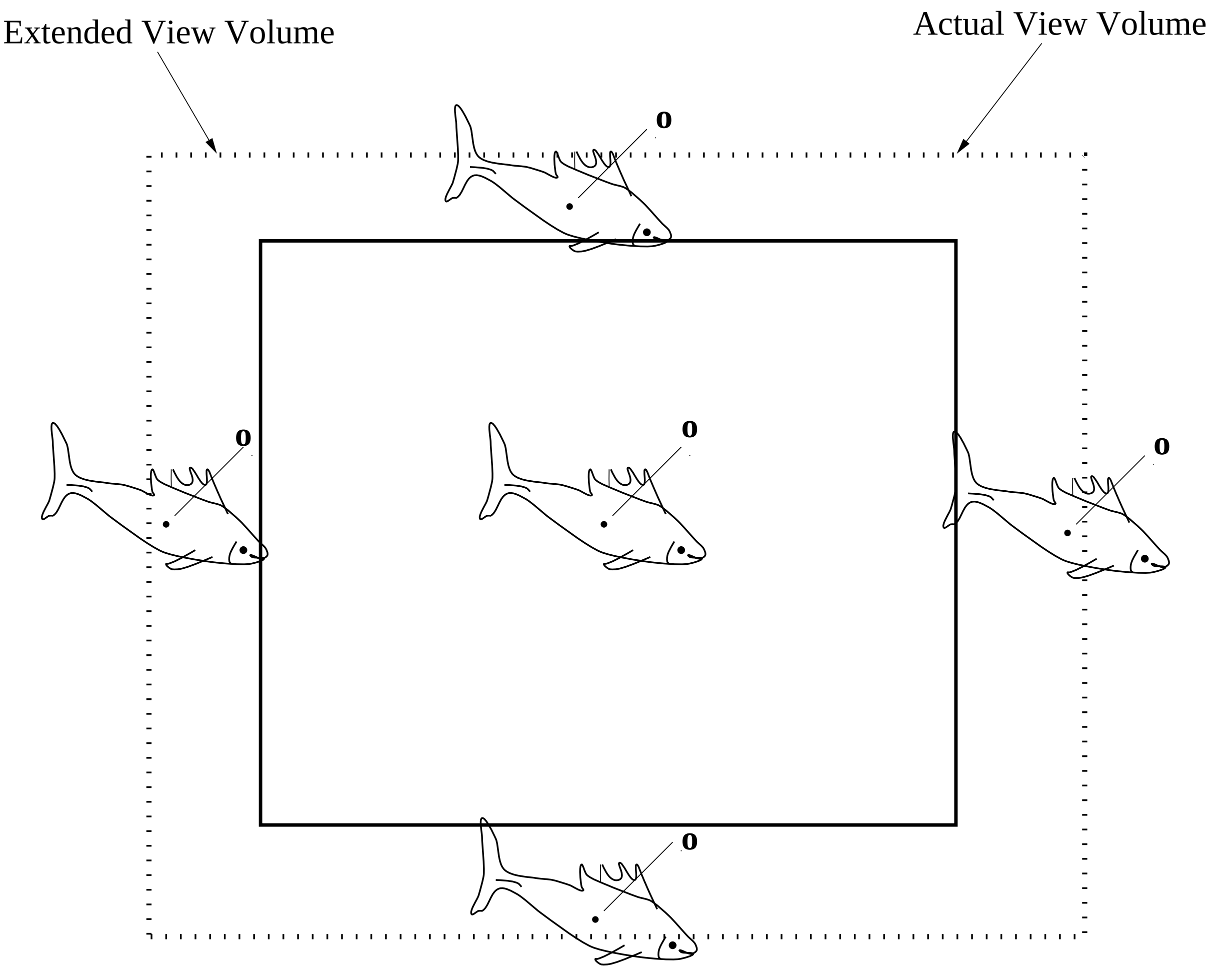



Figure 5.3: A screen cross-section of the actual and extended view volumes

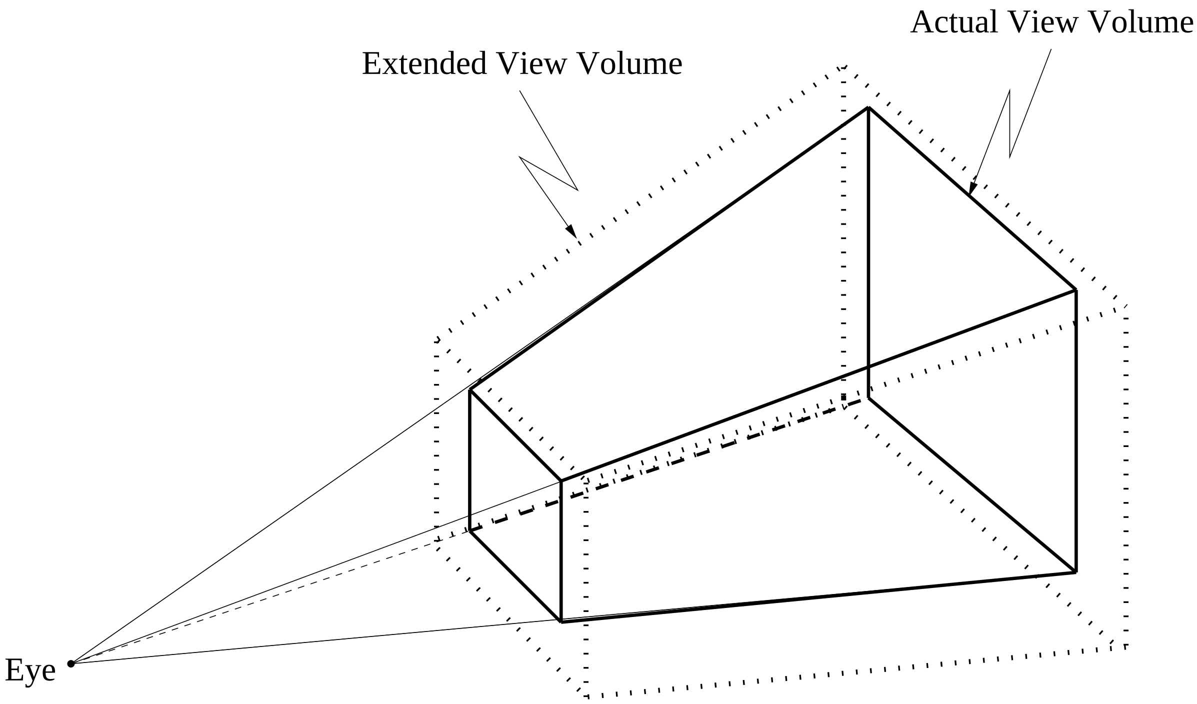



Figure 5.4: The actual view volume and the extended one

pulling the eyepoint back so that the resulting view volume matches the extended one. This strategy only affects the left, right, top and bottom boundaries of the view volume. The near and far clipping planes are not modified. The near plane is set to be fairly close to the viewpoint so that a single point clipping against it will not visibly affect the animation. The far plane is set, depending on the translucency of the water (i.e., the selected fog values), at a distance such that an object in the scene at that distance is hidden by the fog. Since this far value already leaves a small margin at the far end, there is no need to extend it further. As a result, the reduction in computation time for visibility checking more than offsets the processing time for invisible fishes that are occasionally labeled as visible due to the enlarged view volume.

# Chapter 6

# Behavior System

The behavior system (Figure 6.1) of the artificial fish is responsible for higher level behavior, such as dynamic goal setting, obstacle avoidance, foraging, schooling, mating, etc. (see [35]). At each time step, an intention generator examines sensory information acquired by the perception system and selects appropriate action. Because we replace the original biomechanical locomotion controllers of the artificial fish with our action repertoire, we must introduce a secondary controller to mediate between the intention generator and our new motor system.

## 6.1 Secondary Controller

The intention generator makes a decision about the swim pattern and sets the appropriate motor controller parameters for the dynamic model. We rely on these parameter values to select among the action segments in the action repertoire to produce the desired swim pattern. For a forward swim, the parameters determine the swimming speed—slow, medium, or fast—and a suitable swim segment is selected. Similarly for a turn, the parameters determine the turn angles—gradual or sharp—and a suitable turn segment is selected.

As described in Section 4.3, we marked two entry points for each forward swim segment. As a fish decides to break a turn and switch to a forward swim, the next forward swim segment should start with the entry point corresponding to whichever side of the

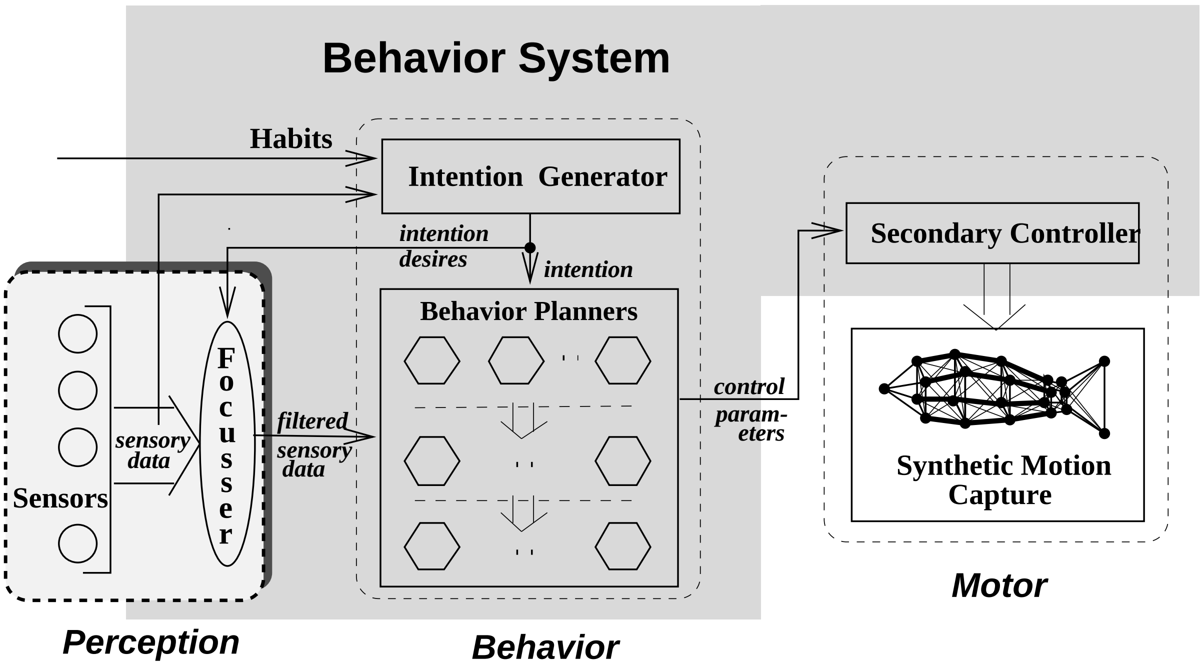



Figure 6.1: The behavior system of the artificial fish

body the tail ends up in after the turning. For example, after a right turn, the tail is on the right side of its body, so the next forward swim segment will start with the right entry point. The selection of the appropriate entry point is done by the secondary controller.

The limited number of action segments in the repertoire reduces the locomotion abilities of the fishes. As a result, the probability of multiple fishes swimming in synchrony increases. To address this problem, the secondary controller monitors how long a fish has been pursuing any particular segment. When the duration exceeds a threshold, the controller will post a recommendation to the intention generator to switch randomly to a different swim pattern. The intention generator decides the feasibility of the recommendation. This approach succeeds in synthesizing the diversity of motion essential to a natural looking marine world.

## 6.2 Behavior Planner Adaptation

Replacing the dynamic model by a kinematic action repertoire reduces the precision of the maneuverability of the fishes, and they may experience problems attaining certain

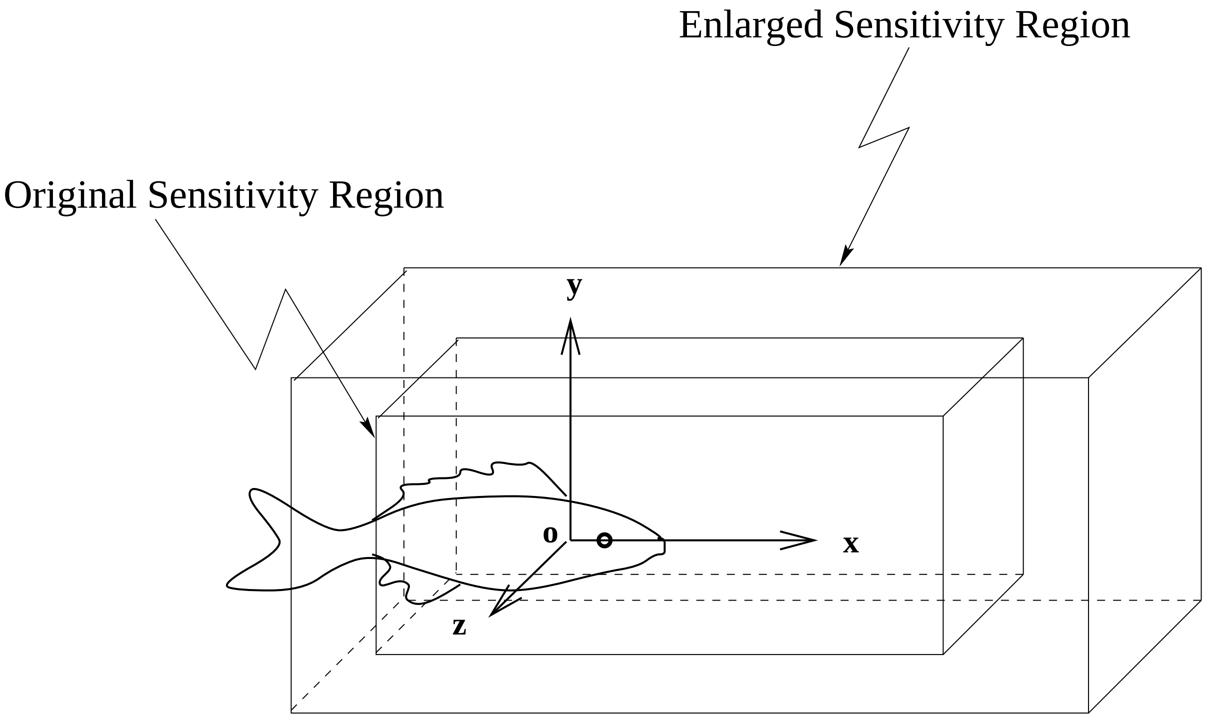



Figure 6.2: Enlarged collision sensitivity region for a fish

goals. For example, the obstacle avoidance mechanism may fail more frequently. We adjust the relevant behavior planners to compensate.

### 6.2.1 Obstacle Avoidance

For obstacle avoidance, we enlarge the sensitivity region for detecting collision threats (Figure 6.2). This gives the fishes enough time and space to maneuver around each other despite the reduced precision.

### 6.2.2 Target Pursuit

Another affected behavior is the pursuit of targets. The original approach had the fish attending increasingly carefully to the location of the target as it approaches. For instance, the fish swims merely in the general direction of a distant target. As it approaches the target, the fish tries harder to steer to its exact location. To offset the weakened motor system, we increase the fish's alertness. The motion planner begins at a further distance its careful steering towards the exact location of the target.

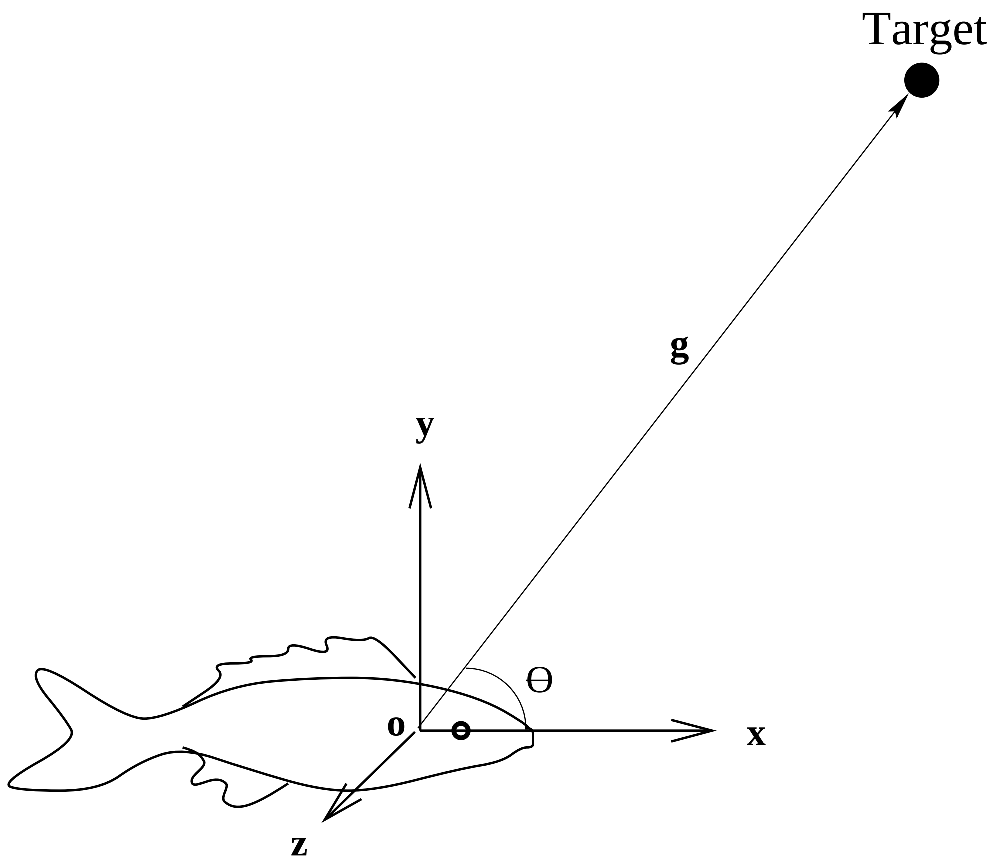



Figure 6.3: Target pursuit

Referring to Figure 6.3, let $\mathbf{g} = \frac{\mathbf{p}-\mathbf{o}}{||\mathbf{p}-\mathbf{o}||}$ denote the direction of a fish's target, where $\mathbf{p}$ is the position of the target in the world coordinate system. As the fish gets closer to its goal point, it tries to decrease the angle $\theta$ between its orientation and its target as much as possible, provided that this steering would not cause "abnormal" behavior such as taking too drastic a change in direction. Hence, we only perform a fine tuning to align $\mathbf{x}$ with $\mathbf{g}$ when $\theta < 45°$. This fine tuning starts when the fish is at a "safe" distance away from the target to allow sufficient time for a small adjustment at each time step, which would not make the change look unnatural, to succeed in bringing the fish to its goal. This strategy has proven successful, as evidenced by the fish's ability to navigate towards and ingest food.

## 6.3 World Constraints

The artificial fishes live in a virtual marine world. They need to have a sense of where the world ends so that they will not swim outside it. The world is basically limited by the seabed. A fish cannot swim underneath the sea floor or anywhere outside it, and a sea surface level must be set above which no fish can survive. A behavior planner is designed to prevent the fishes from swimming outside the world.

The task of keeping the fish inside this enclosed area is similar to collision avoidance except that in this case, the fishes are limited in which direction(s) they can turn to in order to avoid the collision. For example, if a fish is close to penetrating the sea floor, it can no longer swim in a downward direction. We solve this problem by having the controller perform a world boundary test at each time step. The world can be thought of as a room, and the fishes need to be tested against the floor, the ceiling and the surrounding walls. A fish will initiate avoidance action as it approaches the world boundary. In order for a fish to have enough space and time to maneuver in the desired way, we leave enough margin around the boundaries and test the fish against this compressed volume instead.

The sea floor consists of a polygonal mesh representing a height field. For the purpose of the floor collision test, height is the information that is important to us. We use the $x$ and $y$ positions of the center point of a fish as the index into the terrain data to find out the corresponding height for that point. If the fish's height is within a close range of the floor and the fish is still swimming in a downward direction, then the controller will steer it away by slowly turning its orientation upward to force an ascending motion. Because the orientation is specified with respect to the world coordinate system, we can easily turn it upward by increasing its $z$ component, which in world coordinate system always points upward.

The ceiling is dealt with in a similar fashion as for the seabed. When a fish swims sufficiently close to the ceiling and is still heading upward, the controller will direct its orientation downwards by decreasing the $z$ component of its orientation vector slightly over several frames, until the fish initiates a descending motion.

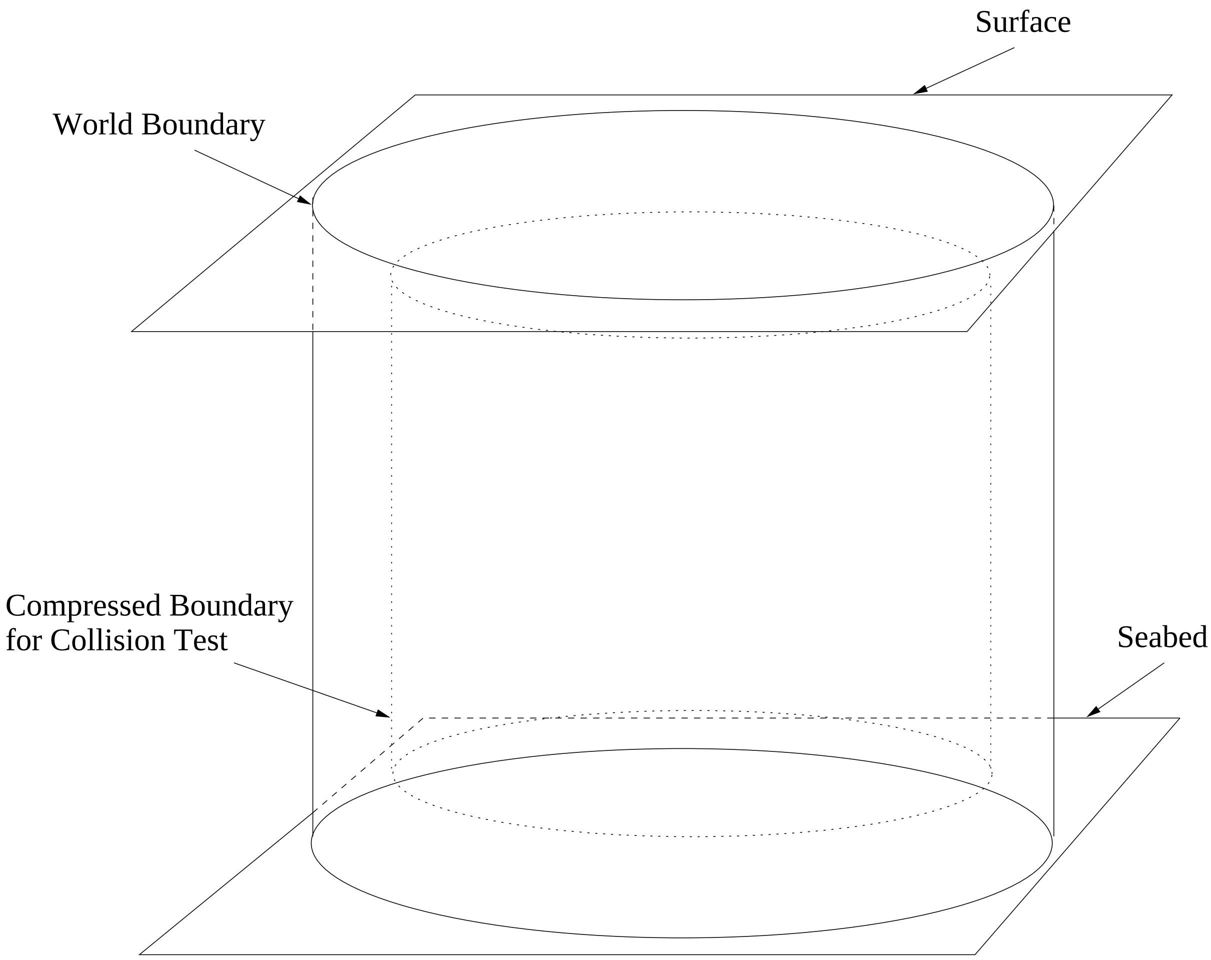



Figure 6.4: Cylindrical world boundary

The walls should coincide with the edges of the seabed. However, this approach will present difficulty for handling the corners of the walls. To avoid this complexity, we instead use a cylinder to approximate the walls (Figure 6.4). With this approach, the checking for potential collision is much easier. We simply check to see if the fish's distance from the origin is sufficiently close to the radius of the cylindrical wall. When it is so, we will turn the fish slowly towards the origin until it is safely away from the wall.

# Chapter 7

# Graphical Display Model

To provide lifelike appearances for the fishes, we use texture mapped B-spline surfaces. Each fish body consists of two juxtaposed B-spline surfaces each of which has $u \times v = 9 \times 21$ control points and is of order 3 along both the $u$ and $v$ axes ($u$ and $v$ represent the two parametric coordinate axes). For fishes that have dorsal and ventral fins, the fins are also B-spline surfaces each has $u = 2$ and $v = 12$ control points and is of order 1 along the $u$ axis and order 3 along the $v$ axis. The left and right pectoral fins are B-spline surfaces each has $u = 2$ and $v = 4$ control points and is of order 1 along the $u$ axis and order 2 along the $v$ axis. Although synthetic motion capture saves significant computation time, the complexity of the highly textured B-spline based graphical display hampers the synthesis of real-time animation. We applied view frustum culling and level-of-detail (LOD) techniques [9, 12] to reduce the rendering time.

## 7.1 View Frustum Culling

Because only a limited number of objects are typically visible at any time in a virtual world such as ours, culling the display of fishes outside the view frustum helps to maintain a relatively fast frame rate, not only by avoiding the graphical display, but also by avoiding nodal position reconstruction and the processing of the B-spline surface control points, as was described in Section 5.4. To reduce the cost, a single point visibility check is used for the fishes. Each fish is approximated by its center point and a bounding sphere. This requires a slight enlargement of the view frustum so that a fish with its center point just off-screen will still be correctly considered as visible. This may cause a few fishes that are marginally outside the view volume occasionally to be regarded as visible, but the reduction in computation time for visibility checking more than offsets. Culling is also applied to the seaweeds using bounding boxes for visibility check.

## 7.2 Fast Rendering Technique

Instead of relying on OpenGL's [21] drawing facility for B-spline surfaces, we implemented a faster drawing routine taking advantage of the properties of B-spline surfaces [22], [28]. A B-spline surface is represented as:

$$\mathbf{s}(u,v) = \sum_{i=0}^{l} \sum_{j=0}^{m} B_i^u(u) B_j^v(v) \mathbf{c}_{ij},$$

where $\mathbf{c}_{ij}$ is the control points that influence the shape of the surface. $B_i^u(u)$ and $B_j^v(v)$ are the B-spline basis functions. The basis functions for each parameter need not all have the same order, and each basis function family is indexed depending on the size of its associated knot vector. They determine the influence of the control points $\mathbf{c}$ on the shape of the surface.

Since we only use one level of tessellation for the B-spline surfaces, i.e., we know the order and knot vectors prior to drawing time, we pre-evaluated the B-spline basis functions at tessellation points and stored these values in a table. At drawing time, we can quickly update a pre-existing tessellation whenever a control point is changed, and therefore save the time to re-evaluate the basis function values at each time step.

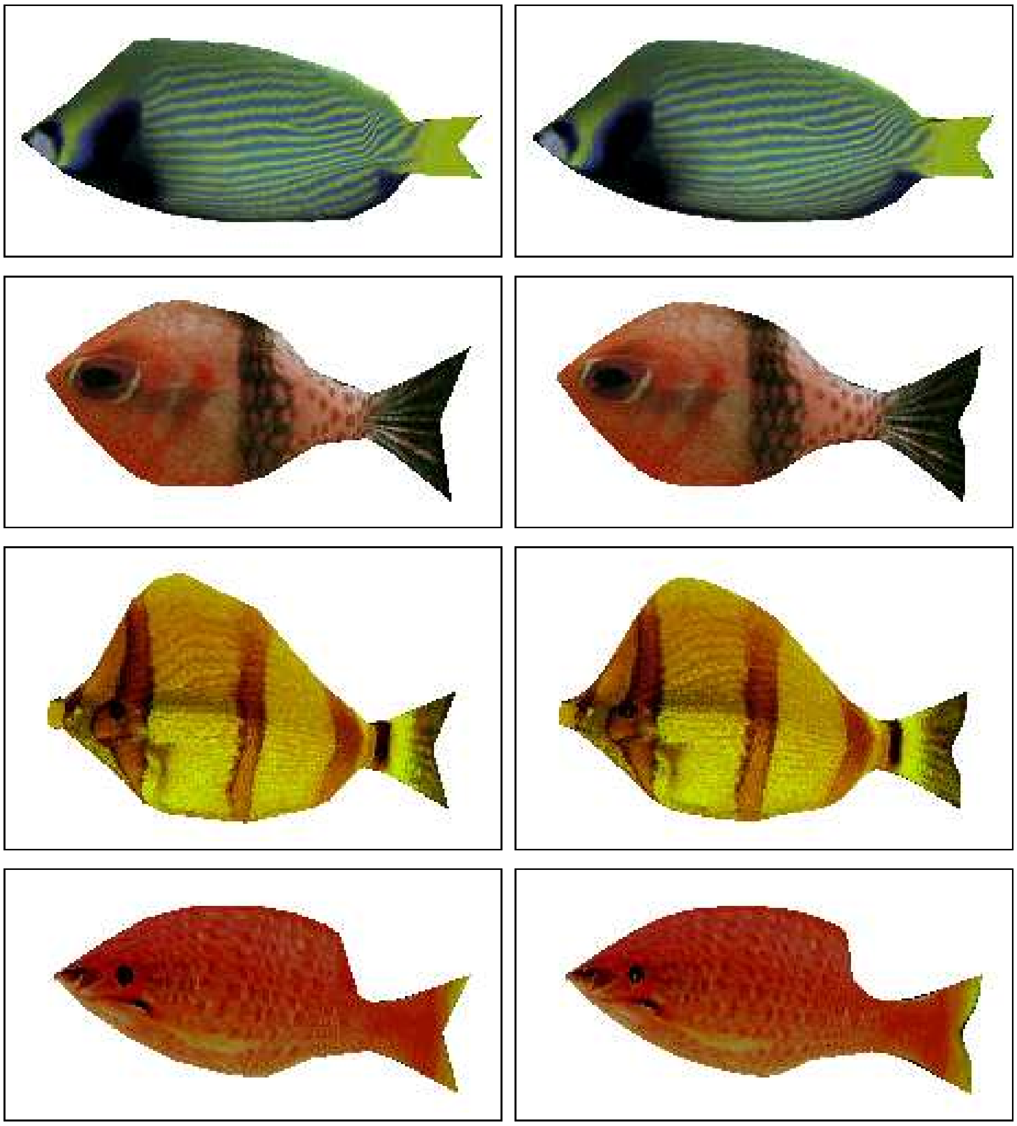
If multi-resolution is to be applied to the B-spline surfaces, we can still pre-evaluate these basis function values, provided that the less detailed levels are subsets of the most detailed one. In this case, only the basis functions for the highest level need to be computed and the lower levels can be accessed through a subrange of the pre-stored values. For example, assume two levels of detail are used, one level $A$ has $4 \times 4$ basis functions, and the other $B$ has $2 \times 2$ basis functions. We can pre-evaluate the basis functions for level A. When level B is applied to the surfaces, we simply reference every other element stored in the table instead.

## 7.3 Level-of-Detail

Similar to the simulation process, we use level-of-detail techniques here to cut back the rendering time. There is always a trade-off between the quality of the appearance and the simplicity of the geometric model used to represent it. The highest level of detail is rendering with B-spline surfaces. The two surfaces corresponding to the fish's body have considerably more control points than the rest, and thus they consume most of the rendering time. Fortunately, due to the dense distribution of the control points, the effect of using their control point meshes, which are surface meshes formed by connecting the $u \times v$ control points of each B-spline surface, closely resemble what can be achieved using rendering with detailed tessellation, as shown in Figure 7.1. Therefore, we decide to use control point meshes to represent any fish's body at the finest level of detail. The fins edges however, appear too piecewise linear with the control point mesh approach for a close view (Figure 7.2). They have to be drawn with the tessellation at the highest level of detail.

Because we are aiming for visual realism, we cannot afford to compromise the underlying geometric model too much for a visible fish. Hence, we represent only distant fishes with control point meshes, i.e., the fins are drawn with control point meshes as well. This effectively reduces the number of polygons sent through the graphics pipeline and eliminates the tessellation time.

We classify the fishes into a few viewing categories according to how far away they are from the viewing point. Experiments yield a depth range beyond which a fish can still be seen before reaching the far threshold but will be distant enough that a switch to using the control point mesh representation will be unnoticeable. This "safe" distance guarantees a smooth transition between the two levels of detail. Figure 7.2 compares these two levels of representations against the original B-spline surface representation where tessellation is applied to both the bodies and the fins. Table 7.1 summarizes the levels of geometric representations used.

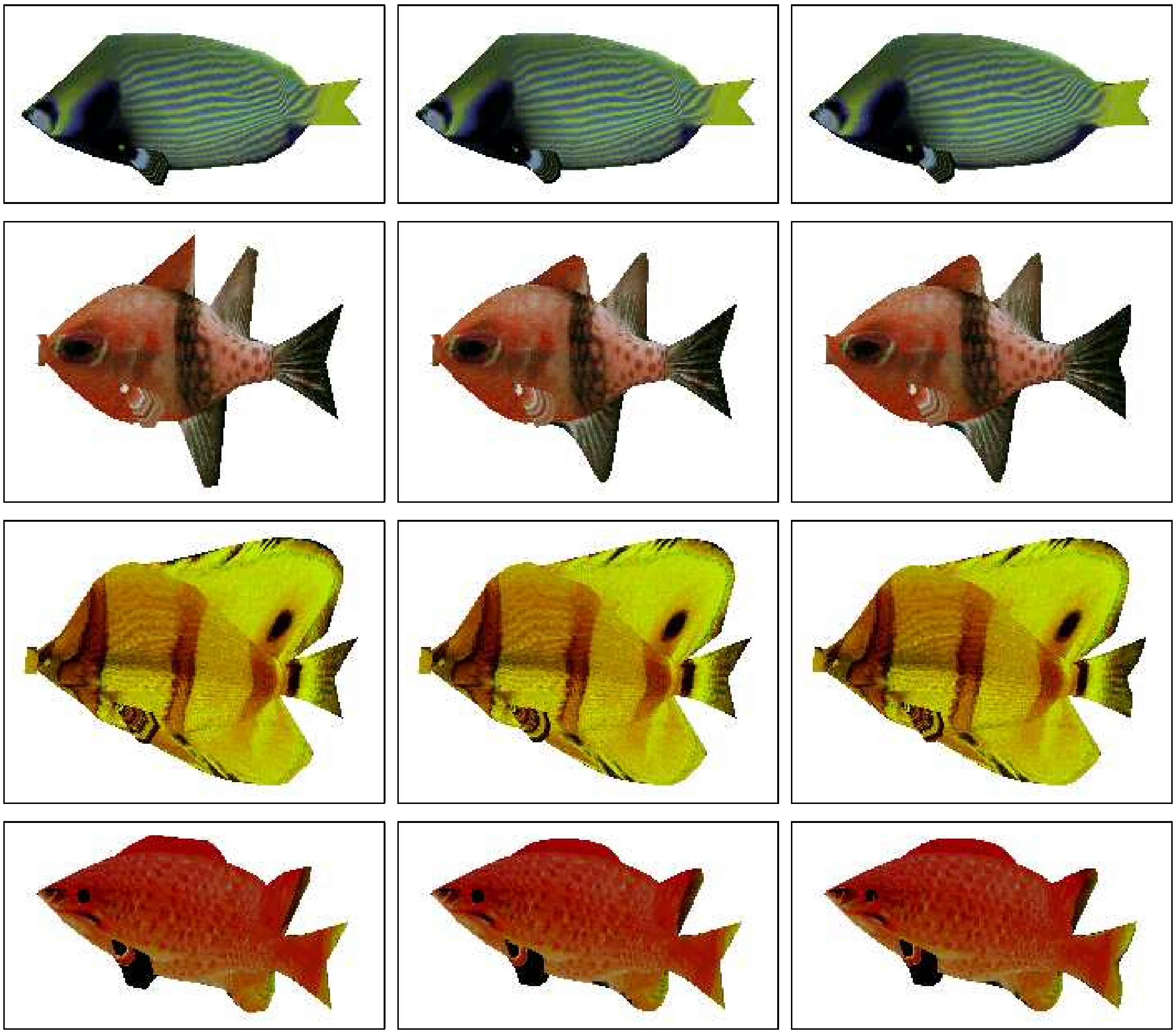


(a) (b)

Figure 7.1: Control Point Mesh Representation vs. Original B-spline Surface Representation

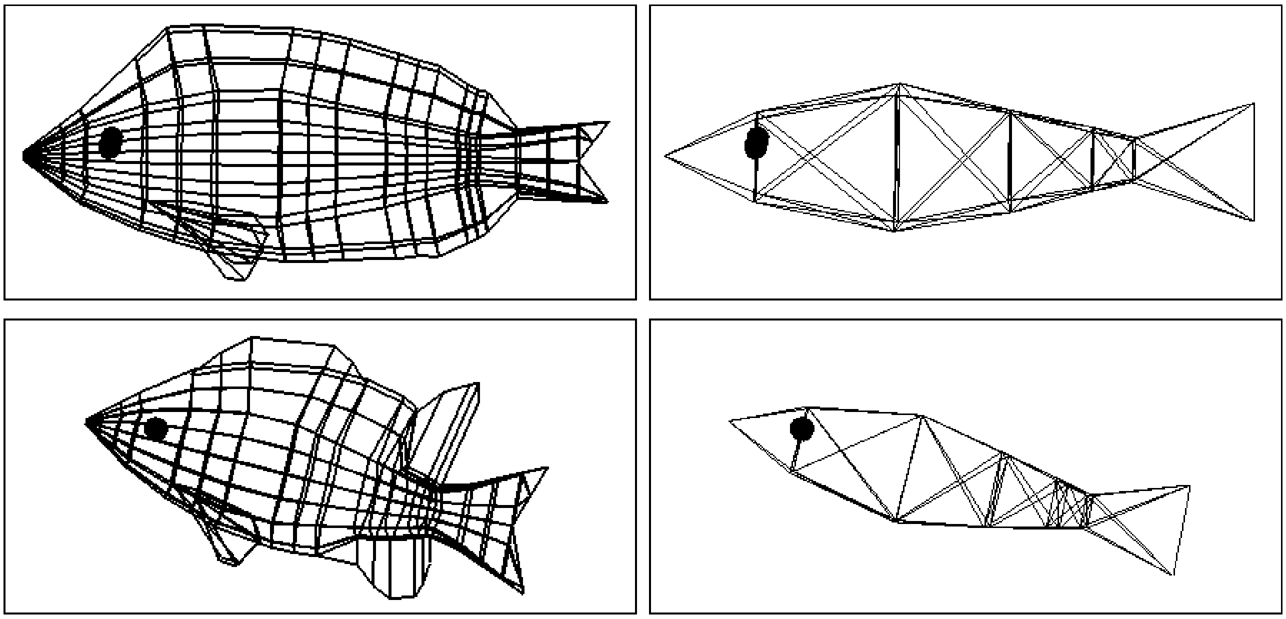
| View | Geometric Representation |
|---|---|
| Close | B-spline Surface for the fins, Control Point Mesh for the body |
| Distant | Control Point Mesh |
| Invisible | None |

Table 7.1: Different geometric representations of fishes



(a) (b) (c)

Figure 7.2: Distant View vs. Close View vs. Original B-spline Surface Representations



(a) (b)

Figure 7.3: Control Point Mesh Representation vs. Spring Wire Frame Representation

A yet simpler representation is the spring wire frame model, which is formed by connecting the nodal positions with springs. This presents too dramatic a change from even the control point mesh representation as can be seen in Figure 7.3, which makes it unusable in our case. In Figs. 1.1 and 1.2, most fishes in the school are far away and they are displayed using control point meshes, while the rest are displayed using the close view representation.

## 7.4 Stereoscopic View

We implemented an option for the users to see the world in a stereoscopic view using a CrystalEyes stereoscopic viewing system. Stereo viewing allows the users to gain compelling depth perception and have a quasi-immersive 3D experience. Figure 7.4 shows the author enjoying the ride with a pair of CrystalEyes glasses.

The stereo option requires each scene to be drawn twice, one for the left-eye view, and one for the right-eye view. We map the left-eye image to the top half of the viewport and the right-eye image to the bottom half. Figure 7.5 shows a sample scene with the

Figure 7.4: The author is enjoying the ride stereoscopically

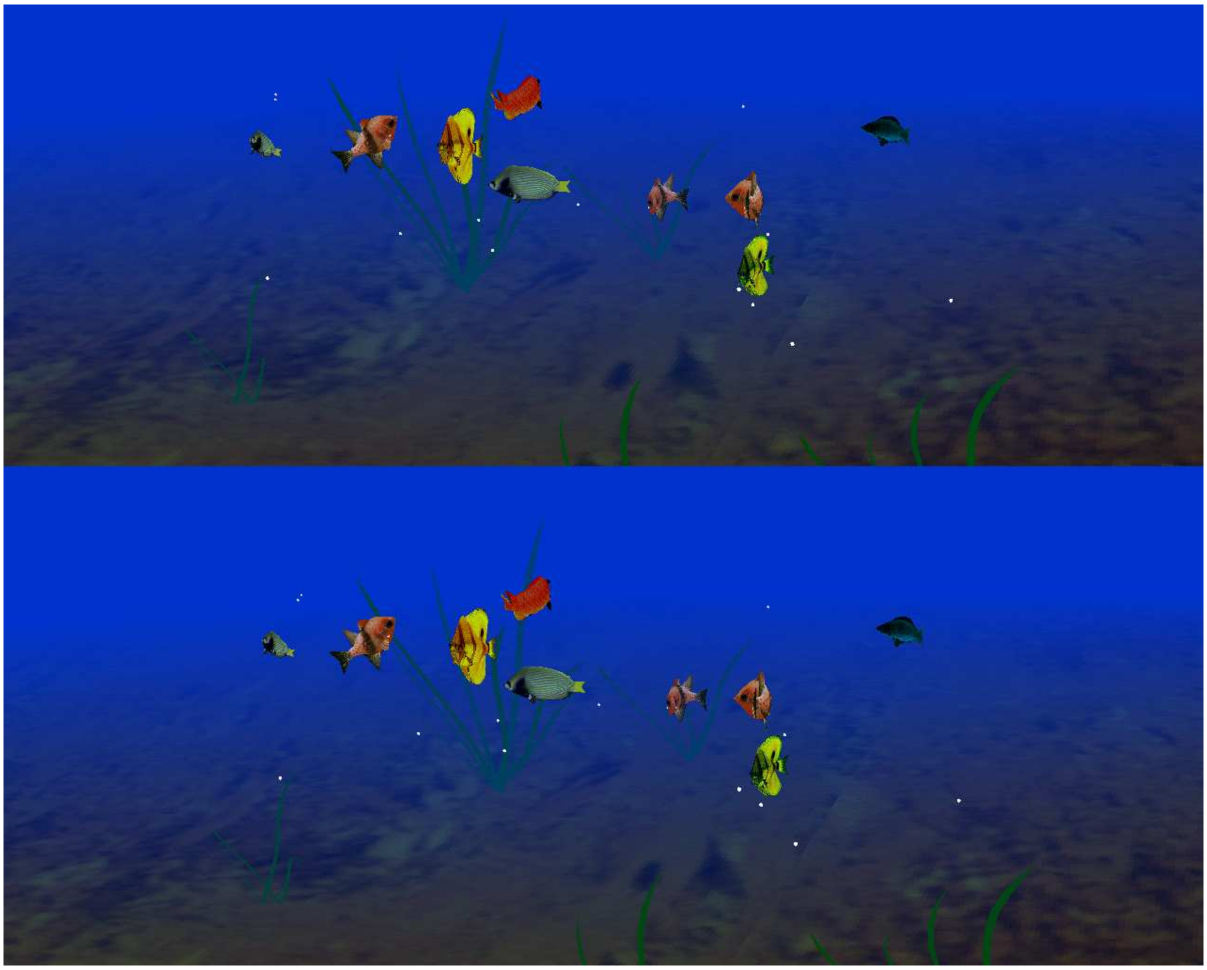


Figure 7.5: Stereoscopic View

left-eye view sitting on top of the right-eye view. The views are presented in alternating fashion to the eyes in synchrony with the triggering of the CrystalEyes shutter glasses by the infrared emitter placed on top of the CRT display.

## 7.5 The Marine Environment

The virtual water in the marine environment is translucent and it is rendered using a bluish fog effect. Besides the fishes, the marine environment also includes plankton, seabed, and seaweeds. In our peaceful marine world, the fishes feed on the white planktons or hide under the aquatic plants as they please.

We continue to model the seaweeds using physics-based modeling as in the original system [34], since this simulation process does not take long. As a fish swims through or passes by, the leaves of these marine plants swing in response to a simulated water force generated by the fish's body. A bounding box is used to cull these seaweeds so that when they cannot be seen, no simulation or display is performed.

# Chapter 8

# User Interface

The dramatically improved frame rate makes interaction possible. Users can now pilot a submarine to navigate around the virtual underwater world. They communicate with the system through a user interface, which consists of a submarine control mechanism and a menu for selecting among several options. We implemented the interface using the GLUT library ([15]).

## 8.1 Submarine Control

The possible maneuvers of the submarine include accelerating, decelerating, retreating, turning left, turning right, pitching up, pitching down, rising up, diving down, and standing still. These steering commands are controlled via the mouse. We treat the display screen as a control panel for this purpose.

Depressing on the left mouse button triggers turning and pitching motions. The angles of turning and pitching depend on the position of the mouse with respect to the center point of the screen, $\mathbf{c} = \{c_x, c_y\}$. A 2D coordinate system $\mathbf{I} = \{\mathbf{i}_x, \mathbf{i}_y\}$ (see Figure 8.1) is set up to assist in calculating the desirable angles. Any time the left mouse button is depressed, the position of the mouse pointer, $\mathbf{m} = \{m_x, m_y\}$ is recorded, and a distance vector $\mathbf{d} = \{d_x, d_y\}$, where $d_x = m_x - c_x, d_y = m_y - c_y$ is evaluated to determine how far away the mouse pointer is from the center point. The horizontal movement of the mouse controls the turning motion, i.e., the turning angle is directly proportional to $d_x$.

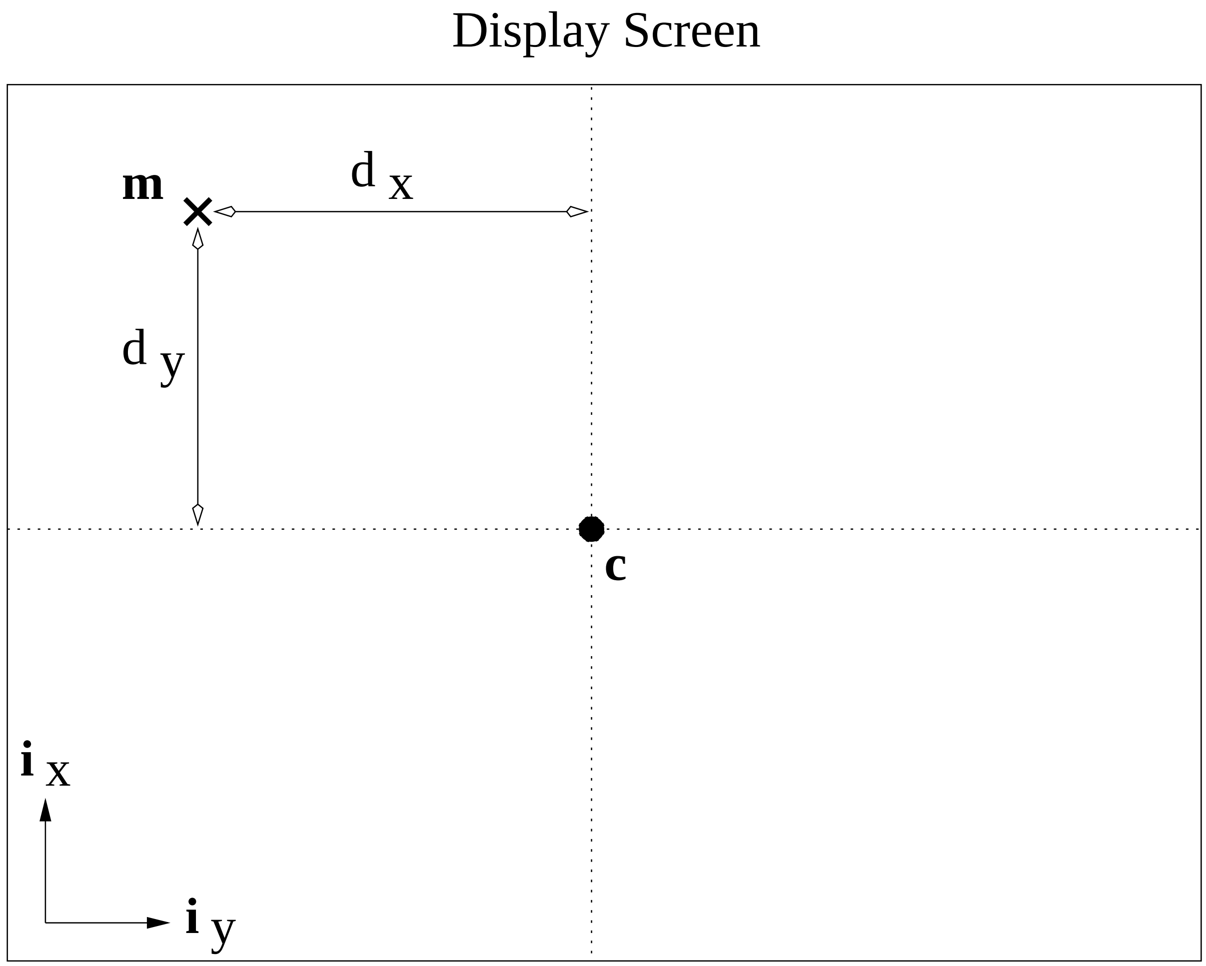



Figure 8.1: The 2D display screen coordinate system

Placing the mouse to the left of **c** results in a left turn and placing the mouse to the right of **c** results in a right turn. The bigger the value for $d_x$, the sharper the turn would be. Similarly, the vertical movement of the mouse controls the pitching motion, i.e., the pitching angle is directly proportional to $d_y$. A mouse position above the center point leads to pitching upward, and below the center point leads to pitching downward. A bigger value of $d_y$ leads to a sharper pitch.

The middle mouse button controls the rest of the movements, except for standing still. When depressed alone, it controls accelerating, decelerating, and retreating. In this case, we are only concerned with the value of $d_y$. Positioning the mouse pointer above the center point causes the submarine to move forward, and positioning it below the center point causes the submarine to retreat. The further away the mouse pointer is from **c**, the

| |
|---|
| Stereoscopic View (On/Off) |
| B-Spline Surface |
| Control Point Mesh |
| Spring Wire Frame |
| Suspend Simulation (On/Off) |
| Reset View |
| Record Mode (On/Off) |
| Quit |

Table 8.1: Selection menu

greater is the magnitude of the acceleration. With two consecutive clicks of the middle mouse button, we see a deceleration when the $d_y$ value associated with the first click is bigger than with the second, and an acceleration vice versa.

When depressed simultaneously with a press of the SHIFT key, the middle mouse button is responsible for lifting and diving motions. A positive $d_y$ value leads to a lifting, and a negative $d_y$ value results in a diving. The rate at which the submarine lifts or dives is directly proportional to the magnitude of $d_y$.

It is often useful to be able to bring the submarine to a standstill, when the user discovers a hotbed of activity. A press of the **S** key will achieve this effect.

## 8.2 Menu

A click on the right mouse button activates a menu from which the viewer select from several options. Table 8.1 shows the available options.

The first four options allow the user to select the display mode. The first one, **Stereoscopic View** is a toggle option (an option that has two states, on and off, a selection switches it from one state to the other). When turned on, the system displays the world in stereo. The next three options **B-Spline Surface**, **Control Point Mesh** and **Spring Wire Frame** allow the user to select among different representations for a fish. **B-spline Surface** uses the geometric models described in Chapter 7. **Control Point Mesh** uses control point meshes to represent each fish including their fins regardless of how far they are from the view point. Texture mapping is turned off with this option. **Spring Wire Frame** uses a spring wire frame model to represent each fish and texture mapping is also turned off here. These last two options produce an even faster frame rate, and they help interested users to learn about the underlying geometric structures used to build the fishes.

Because the fishes are constantly moving, it is sometimes hard for the viewer to see details about a fish. It is therefore useful to be able to suspend the simulation temporarily so that the viewer can observe a fish closely without having to maneuver the submarine trying to keep up with the fish. The toggle option **Suspend Simulation** is provided for this purpose. When turned on, the simulation will be suspended until it is turned off.

The **Reset View** option brings the submarine back to where it was at the beginning of the trip.

Users often like to take pictures for later appreciation. It would be nice to catch the colorful marine life on still images and the toggle option **Record Mode** provides this option. The system will automatically record each frame in RGB image file format when **Record Mode** is turned on and continue to do so until it is turned off. The user can retrieve these image files later. The last option **Quit** allows the user to terminate the program.

# Chapter 9

# Results and Discussion

Using the techniques presented in this paper, we achieve an interactive frame rate on a Silicon Graphics $1 \times 194$ MHz R10000 InfiniteReality workstation. With the improved speed, users can explore the virtual marine world as if they are piloting a submarine. The user can navigate the vessel using the mouse. Figure 1.1, 1.2 and 1.3 are still images captured on one of our virtual submarine dives. We also implemented the option to see the world stereoscopically, providing a compelling depth perception and a quasi-immersive 3D experience. Figure 7.4 shows the author enjoying the ride, wearing a pair of CrystalEyes stereo glasses.

## 9.1 Speed Comparison

The speed up over the original biomechanical animation is attained by accelerating the motor system and the graphical display model. Table 9.1 compares the computation times required for the dynamic model and our motion capture model, for a single fish.

| Simulation Method | CPU Time (ms) |
|---|---|
| Dynamics | 40 |
| Synthetic Motion Capture | 0.01 |

Table 9.1: The processing time for each fish with dynamic simulation and with synthetic motion capture

| Graphics | CPU Time (ms) |
|---|---|
| Without LOD | 2340 |
| With LOD | 85 |

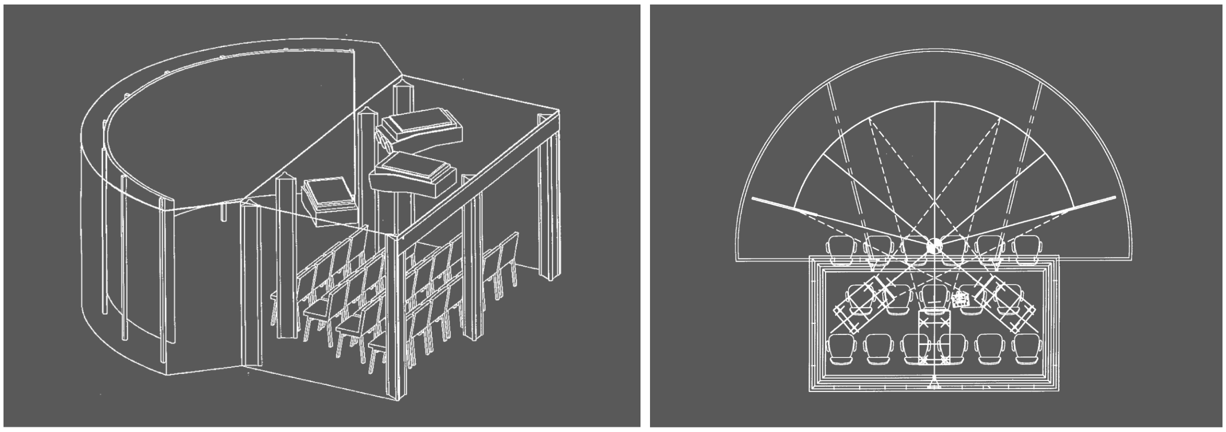
Table 9.2: Rendering time for a similarly complex scene without and with the use of culling and level-of-detail rendering

The indicated times are for the case when the fish is fully visible, which includes the reconstruction of the body coordinate system and positions of all the nodal points. The computation time is reduced by a factor of 4000. Table 9.2 shows the rendering time for a complex scene (about 40 fishes can be seen) with and without culling and multi-resolution. Our current technique cuts the rendering time by a factor of about 27.5.

The sustainable update rate of the original (biomechanics-based) artificial fishes animation system is about 0.25 frames per second, making it impossible for the user to perceive continuous motion, let alone navigate the virtual world. With the help of culling and level-of-detail in the graphical display, our current implementation has a fluctuating frame rate that depends on the number of visible fishes and the percentage of fishes that are in close view. We observe frame rates in the range of 10 to 50 frames per second. This is fast enough to provide the user a sense of action and interactiveness. We allow the frame rate to fluctuate because to keep a consistent rate, the frame rate must be set at the minimum, which is only quasi-real time. Fixing the frame rate at this relatively low speed affects the overall animation. As the hardware improves, when we can achieve fast enough speed, we can then fix the rate to achieve more realistic visual effect.

## 9.2 Reality Theater

We have furthermore developed a large scale version of our virtual undersea world in a "Relocatable Reality Theater" (Figure 9.1(a)) marketed by Trimension, Inc. [33], which combines a Silicon Graphics $8 \times R10000$ CPU Onyx2 system and multichannel PRODAS projection technology (Figure 9.1(b)) from SEOS Displays, Ltd. The system features three InfiniteReality graphics pipelines, each feeding video to an overhead projector.



(a) (b)

Figure 9.1: Relocatable Reality Room

This system animates and renders our virtual world at a sustainable rate of at least 30 frames per second. It renders through the three projectors a seamless image of approximately $4000 \times 1000$ pixel resolution across a $18 \times 8$-foot curved screen, producing a large panoramic display that fills the peripheral view. Figure 9.2 shows the theater.

To take advantage of the separate pipelines and multiple processors, we switched from using the GLUT library to using IRIS Performer [29], [7]. The drawing routines remain in OpenGL. With IRIS Performer, we divide our system into 3 processes: application, culling and drawing. The application process is responsible for initializing the channels and simulating the movements of the fishes. The culling process is responsible for database culling, and the drawing process is responsible for displaying geometry produced by the culling process. Because there are three separate pipelines, the drawing process is further forked into three subprocesses each drawing to a separate graphics pipeline corresponding to the three projectors.

Having achieved real time performance on this system, a sustainable frame rate of over 30FPS, the results are spectacular. With the hemi-cylindrical display, the audience enjoys a compelling submarine ride (Figure 9.2 and 9.3). When the stereoscopic view option is turned on, users can see the fishes moving around them and they enjoy the visual sensation of diving.

## 9.3 Discussion

Rendering speed limits how many fishes can be incorporated into the system. Currently, there are 60 fishes in total belonging to 7 species. Our system has been built in such a way that it is easy to increase the number of fish for an existing species. Due to the fast kinematic simulation time, processing time will no longer be the main concern. However, graphical display remains the bottleneck. Too many fishes seen in full view at a time may reduce the frame rate well below interactive rates. With graphics hardware evolving rapidly, we will soon be able to not only achieve true real-time performance, but also enrich our environment with more artificial animals.

Adding fish from an unsimulated species is not as easy as from a simulated species. Because the source of our action repertoire is the physical simulation, we would have to build a biomechanical model for the new species and be able to simulate its locomotion physically first, unless a physical model for it already exists. We would like to expand the potential source of our data to include information that can be gathered from marine biology literature and extracted data from existing animations.

Figure 9.2: The virtual undersea world experienced on the panoramic display

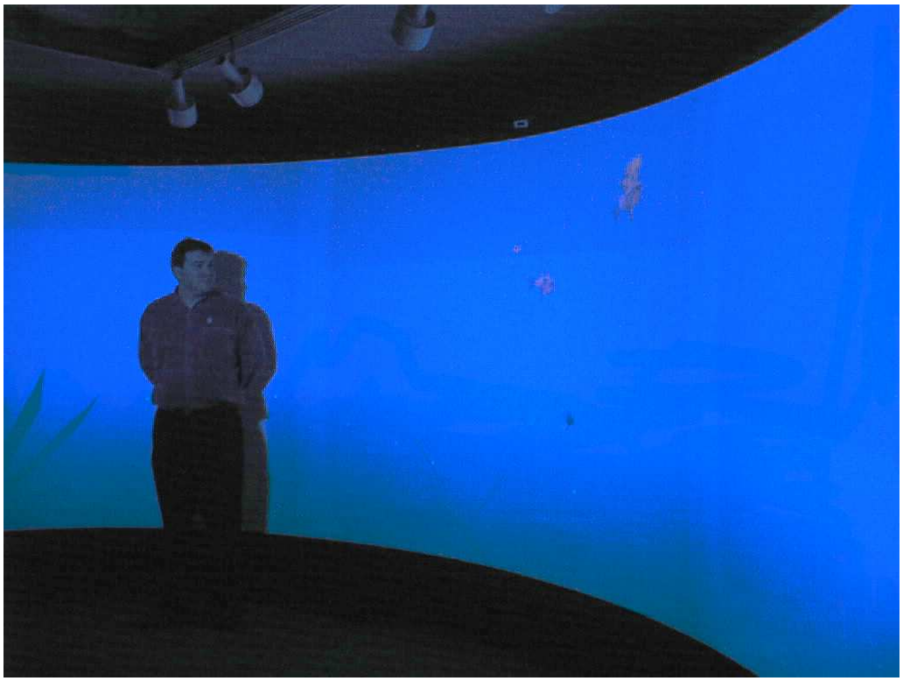


Figure 9.3: A closer look at the curved screen inside the Reality Theater

# Chapter 10

# Conclusion and Future Work

## 10.1 Conclusion

In this thesis, we have introduced the idea of replacing biomechanical models of animals for use in computer animation, with ultra-fast kinematic replicas that capture with reasonable fidelity the locomotion abilities of the original models. In applying synthetic motion capture, we collect segments of motion data generated through the systematic numerical simulation of the biomechanical model, select a minimal set of action segments that parsimoniously represents the various locomotion patterns, and compile these segments into an action repertoire for the artificial animal. The motor system retrieves action segments from the action repertoire to synthesize continuous kinematic locomotion, using motion warping to smooth transitions between different locomotion patterns. The artificial animal's behavior system combines locomotion patterns into meaningful higher-level behavior.

To demonstrate the power of our approach, we have developed an interactive system that provides users a virtual undersea experience. The user pilots a virtual submarine to explore a marine environment populated by lifelike fauna. Excluding rendering, our synthetic motion capture approach is three orders of magnitude faster than the original biomechanical simulation of the artificial fishes. By eliminating the significant burden of numerical simulation, the frame rate of our virtual world becomes bounded by graphics rendering performance. We accelerated rendering by culling objects relative to the view frustum and displaying visible objects with a suitable geometric level of detail based on their distance from the viewpoint.

Combining advances in artificial life and virtual reality, our system enables a user to explore an exciting virtual underwater world without having to confront any of the dangers that scuba divers face. With regard to artificial life, we have demonstrated the production of lifelike animations of marine animals at interactive frame rates, so that users can see continuous motions on the screen as they are generated. With regard to virtual reality, a system of realism equivalent to ours would normally require much more expensive, specialized equipment than what we used.

## 10.2 Future Work

**Increased Interactiveness**

In the current implementation of our virtual world, the users have full control over the submarine with which they can explore the world and observe the underwater life. However, the inhabitants of the world are not aware of the submarine. We would like to heighten the excitement along the trip by having the fishes respond to the submarine. One possible scheme is to have the fishes regarding the submarine just as another large fish. The pacifists may be intrigued by such a stranger to their world and attempt to swim closer for a better look to satisfy their curiosity. The prey fish however, would more likely treat the submarine as a huge predator. Hence, upon encounter, they will either form a school or scatter in terror trying to flee to safety.

Another possible scheme is to have the user attaching him/herself to a fish. The user will then have control over this particular fish's action, seeing the world through the fish's eyes. With different users attached to different fishes, we can effectively create a game where multiple users can cooperate together to set up their own marine kingdom.

**Improving the Action Repertoire**

The current action repertoire contains a relatively small library of locomotion data which allows for realistic swimming motions. Elaborate behaviors such as mating are affected more seriously by the limited amount of locomotion data, hence the reduced maneuverability of the fishes, than are the simpler behaviors.

One possible solution is to expand the action repertoire. Increasing the number of action segments recorded improves the motor control. There is a trade-off between the memory requirement and the quality of motion. We have chosen the current collection of data to achieve a healthy balance between the two. As the memory requirement increases, the possibility of paging also increases, which may in turn affect the frame rate.

Using physics-based modeling as the source of motion data for our action repertoire, we eliminate the need for expensive motion capture equipment, and the noise that is normally present in data collected using conventional motion capture techniques. Furthermore, attaching sensors to live animals and have them act in certain ways may prove to be difficult. However, because the dynamical simulation is only an approximation for the actual movements, it limits the quality of the captured data. In another word, the captured motion can only be as realistic as what can be generated by the dynamic simulation. It would be useful to expand the source of our motion data to include data captured from real animals.

### More Action

At present, we have a peaceful marine world where every fish feeds on plankton and there are no predators. This limits the kind of actions one can see. The addition of predator species will bring some interesting scenarios into the otherwise quiet marine environment. A greater variety of pelagic creatures also helps to increase the realism and stimulates the viewer's curiosity. As physical models of other animals like Grzeszczuk's shark and dolphin [11] already exist, they serve as the perfect candidates to introduce into our virtual marine world.

### Improving the Quality of Animation

As far as providing natural-looking motion is concerned, dynamic simulation still dominates in the world of simulation models. The reason that we have abandoned it completely in our approach is due to the lengthy computation time associated with it. It is impossible on our current workstation to achieve interactive frame rate with any dynamics left in the system. However, ever increasing computing power will one day make it possible for dynamic simulation to play a role in our multiple level-of-detail simulation models without excessive sacrifices in the frame rate. When that day comes, we can use dynamics for fishes within a close view and our synthetic motion capture model for distant fishes. This will improve the quality of the animation. It will also eliminate the need to expand the action repertoire in order to have the fishes succeed in participating in more complex actions. A smooth blending between the kinematic model and the dynamic model can be easily achieved by adding some spring values associated with the biomechanical model to our data collection for the action repertoire.

A "Turing test" will be useful in helping us to assess the quality of our animation. By comparing the locomotion produced by our technique to live actions of real fishes, we may be able to see clearer directions for future improvement.

Improving the virtual environment inhabited by the artificial animals helps to enhance the quality of the animation. By introducing a larger variety of marine plants and including more interesting terrain data, for example, instead of a relatively plain seabed, we can have small hills, the seabed can be covered not only with sand but also with rocks, and coral reefs, etc, we can create a much more realistic underwater world.

**Generalizing Our Technique**

In the current implementation, we targeted our technique towards generating swimming motions for fishes. We believe it can be generalized to other biomechanical animal models. For example, we may be able to use synthetic motion capture to generate swimming motions for humans, or to produce other types of motions such as walking, and jumping for physics-based articulated figures. However, when applied to articulated figures, we run into the problem of path planning. The feet of articulated figures should maintain contact with the ground. Therefore, terrain variations pose a problem in adapting motion capture data. Although research has been done on the subject, it remains an open problem.

# Bibliography


[1] Norman I. Badler. Articulated figure animation. *IEEE Computer Graphics and Applications*, 7(6):10–11, 1987.

[2] Christoph Bregler and Jitendra Malik. Video motion capture. Technical Report CSD-97-973, Computer Science Division, University of California, Berkeley, 1997.

[3] Armin Bruderlin and Lance Williams. Motion signal processing. In *SIGGRAPH 95 Conference Proceedings*, pages 97–104, August 1995.

[4] N. Burtnyk and M. Wein. Computer-generated key-frame animation. *Journal SMPTE*, 80:149–153, 1971.

[5] N. Burtnyk and M. Wein. Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *Communications of the ACM*, 19(10):564–569, October 1976.

[6] Deborah A. Carlson and Jessica K. Hodgins. Simulation levels of detail for real-time animation. In *Proceedings of the Graphics Interface*, pages 1–8, May21–23 1997.

[7] Patricia Creek, Jed Hartman, and Gary Kuhn. *IRIS Performer Programmer's Guide*. Silicon Graphics, Inc., 1995.

[8] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, August 1993.

[9] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 247–254, August 1993.

[10] John P. Granieri, Jonathan Crabtree, and Norman I. Badler. Production and playback of human figure motion for visual simulation. *ACM Transactions on Modeling and Computer Simulation*, 5(3):222–241, July 1995.

[11] Radek Grzeszczuk and Demetri Terzopoulos. Automated learning of Muscle-Actuated locomotion through control abstraction. In *SIGGRAPH 95 Conference Proceedings*, pages 63–70, August 1995.

[12] Paul S. Heckbert and Michael Garland. Multiresolution modeling for fast rendering. In *Proc. Graphics Interface '94*, pages 43–50, May 1994.

[13] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In *SIGGRAPH 95 Conference Proceedings*, pages 71–78, August 1995.

[14] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[15] Mark J. Kilgard. The opengl utility toolkit (glut) programming interface: Api version 3. Technical report, Silicon Graphics, Inc., February 23 1996.

[16] D. H. U. Kochanek and R. H. Bartels. Interpolating splines with local tension, continuity, and bias control. In *SIGGRAPH '84 Conference Proceedings*, pages 33–43, July 1984.

[17] Alexis Lamouret and Michiel van de Panne. Motion synthesis by example. In *Proceedings of the 7th Eurographics Workshop on Simulation and Animation*, pages 199–212, Poiters, France, 1996.

[18] R. Maiocchi. 3-D character animation using motion capture. *Interactive Computer Animation*, pages 10–39, 1996.

[19] C. Manetta and R. Blade. Glossary of virtual reality terminology. *International Journal of Virtual Reality*, 1(2), 1995.

[20] Gavin S. P. Miller. The motion dynamics of snakes and worms. In *SIGGRAPH 95 Conference Proceedings*, pages 169–178, August 1988.

[21] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL Programming Guide*. Addison-Wesley, Reading MA, 1993.

[22] Victor Ng-Thow-Hing and Eugene Fiume. Interactive display and animation of b-spline solids as muscle shape primitives. In *Computer Animation and Simulation '97*, pages 81–97. Springer-Verlag/Wien, 1997.

[23] J.T. Ngo and J Marks. Spacetime constraints revisited. *ACM Computer Graphics, SIGGRAPH'93*, 27:343–350, August 1993.

[24] R. Pausch and T. Crea. A literature survey for virtual environments: Military flight simulator visual systems and simulator sickness. Technical Report CS-92-25, Department of Computer Science, University of Virginia, August 19 1992.

[25] Randy Pausch, Jon Snoddy, Eric Hazeltine, Robert Taylor, and Scott Watson. Disney's aladdin: First steps toward storytelling in virtual reality. In *SIGGRAPH 96 Conference Proceedings*, pages 193–204, August 1996.

[26] John C. Platt and Alan H. Barr. Constraint methods for flexible models. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):279–288, August 1988.

[27] W. T. Reeves. Inbetweening for computer animation utilizing moving point constraints. *Computer Graphics*, 15(3):263–269, August 1981.

[28] David F. Rogers and Linda A. Adlum. Dynamic rational b-spline surfaces. *Computer-aided Design*, pages 609–616, August 1990.

[29] J. Rohlf and J. Helman. IRIS Performer: A high performance multiprocessing toolkit for real-time 3D graphics. *Computer Graphics*, 28(Annual Conference Series):381–394, 1994.

[30] Charles F. Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH 96 Conference Proceedings*, pages 147–154, August 1996.

[31] I. E. Sutherland. The ultimate display. In *Proceedings of IFIPS Congress*, volume 2, pages 506–508, New York City, NY, May 1965.

[32] D. Terzopoulos, editor. *Artificial Life for Graphics, Animation, Multimedia, and Virtual Reality*, volume 23 of *ACM SIGGRAPH'97 Course Notes*. 1997.

[33] Trimension, inc., http://www.trimension-inc.com.

[34] Xiaoyuan Tu. *Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception, and Behavior*. PhD thesis, University of Toronto, 1996.

[35] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, pages 43–50, July 1994.

[36] M. van de Panne and E. Fiume. Sensor-actuator networks. *ACM Computer Graphics, Annual Conference Series, Proceedings of SIGGRAPH'93*, August 1993.

[37] Michiel van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–223, 1997.

[38] Andrew Witkin and Zoran Popović. Motion warping. In *SIGGRAPH 95 Conference Proceedings*, pages 105–108, August 1995.

[39] Johnson K. Yan. Advances in computer-generated imagery for flight simulation. *IEEE Computer Graphics and Applications*, 5(8):37–51, August 1985.