

Physically-based Animation of Humanoid Swimming

Shahzad Malik

Nigel Morris

Paul Yang

Department of Computer Science

University of Toronto

{ smalik | nmorris | paulyang } @ dgp.toronto.edu

Abstract

In this paper, we describe a dynamic control algorithm that allows a humanoid character to swim through a fluid. The swimming is physically-based, whereby the character applies simulated muscle forces in order to drive the body through the motions of a breaststroke. The interaction between the moving body parts and a fluid dynamics system results in drag forces that cause the swimmer's body to be thrust forward. Directional control of the swimmer is achieved by applying various perturbations to the original stroke based on a desired trajectory. We analyze the results qualitatively by comparing our resulting animations against video footage of real swimmers.

1. Introduction

It has been said that the Greek philosopher Plato declared men who didn't know how to swim as uneducated. Today, virtual human characters can perform a slew of dynamic maneuvers ranging from walking and running to dancing and acrobatics, but little effort has been focused on virtual swimming. Thus, it is time to educate our virtual friends.

We describe a dynamic control algorithm that allows a rigid-body model of a human to swim through a fluid. As a start, we have chosen to animate the breaststroke, but our system is general enough to handle any type of swimming motion. The rigid body model is animated by simulating muscle forces via torques applied at the joints. State machines are used to define the behaviour of the rigid body model, and control laws define the amount of muscle strength (torque) to apply. The interaction between the rigid bodies and a simple fluid dynamics model provide the forces that are used to thrust the character forward through the liquid. To use our model, a user or animator is given access to two sets of parameters:

- *Control parameters:* these include such things as target location and orientation in order to guide the swimmer.
- *Body and environment parameters:* these include such things as masses, muscle strength, and dimensions for the various rigid body parts, and fluid parameters (currently only viscosity is defined).

The rest of this paper is organized as follows. The next section discusses existing work that is relevant to physics-based swimming animation. Section 3 then outlines our swimming animation system. Section 4 then describes the

muscle-model and directional control used to put the swimmer in motion through the fluid, followed by Section 5 which describes the fluid dynamics system. Section 6 then describes our results, and Sections 7 and 8 summarize the paper and present avenues for future research.

2. Related Work

In the past few years, many dynamic control algorithms for animating physics-based characters have been proposed in the literature. However, none has directly addressed the issue of humanoid swimming in fluids. Interesting algorithms for physics-based running, cycling, and vaulting were described in [6], but in these cases the physical environment in which the actions were taking place was considered static (unlike fluid). Similarly, independent animation algorithms for visualizing fluid flows have been proposed in [14, 17], but none of these take into account the continuous feedback between the fluid and moving bodies, as would be required for a swimmer.

Some motivating work that takes the swimmer-fluid feedback into account was described in [16]. Here, a virtual marine world with fish that hunt, flee, mate, and wander was described. The fish were modeled as spring-mass systems with sinusoidal patterns actuating the springs and propelling the fish through the water. Their layered architecture consists of an intention generator that creates goal-directed behavior, a motor system that implements higher level motion primitives such as "swim forward" or "turn right," and motion controllers that translate low level control parameters such as speed and direction into muscle actions.

In a similar fashion, a physically-based system for animating birds is described in [10]. This system models the motion of wings as a function of time, and aerodynamic principles are used to vary geometric parameters and aerofoil sections of the wing to capture realistic bird flight. Targeting is achieved by controlling the pitch of the bird's body around its centre of gravity.

Another interesting work regarding swimming bodies is described in [12]. Here, virtual creatures are "evolved" to swim, walk, jump, and follow in a virtual 3D world by using genetic algorithms to modify morphologies and muscle forces. Fitness evaluations (measures of success) act as an optimization process in order to drive the evolution towards the desired behaviour. In a similar fashion, [9] describes

how to use sensor-actuator networks to automatically generate controllers for simple jointed objects. This is done using an initial random search, followed by a hill-climbing refinement phase. While the results are nice, the problem with these stochastic approaches is the lack of user control over the resulting behaviours.

3. System Outline

Our system is implemented completely using the Maya API and MEL scripting language, providing us with a rich set of modeling, rendering, and animation features that we can build upon. Our swimmer is defined using rectangular rigid bodies, with hinge constraints between each to join the bodies together into a humanoid character. Note that while each hinge constraint only provides one degree of rotational freedom, the hinge axis can still be oriented in any way such that our swim stroke does not have to be confined to a single plane. Figure 1 shows a skinned version of our hinged rigid body swimmer and the associated hinge axes.

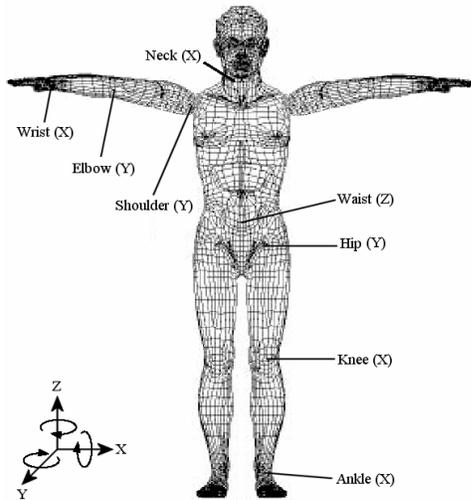


Figure 1 – Hinged swimmer and rotational degrees of freedom

The motion of our swimmer is described by applying appropriate hinge torques to the constraints (described further in Section 4). A Maya plugin called `dynSwim` was developed to encapsulate this muscle control system. At runtime, a single `dynSwim` instance drives the motion of the entire rigid body skeleton. To compute lift and drag forces to thrust the swimmer forward (Section 5), another plugin called `dynFluid` was developed. Since the fluid forces are computed separately for each body part, a separate instance of `dynFluid` is declared for each rigid body. Due to Maya’s connection-based architecture, the entire swimming system is updated each time Maya updates its current frame time.

Since the rigid body skeleton itself isn’t very impressive visually, a jointed skeleton is defined based on the positions of the various constraints. An arbitrarily complex 3D mesh can then be skinned onto this jointed skeleton for improved visuals.

As mentioned earlier, the user is provided with limited high-level control of the animation via a target position and target orientation. The only environment parameters that the user can currently set are fluid viscosity and hand dimensions. Based on these inputs, the swimmer continues to perform the desired swim stroke, and the targeting system makes the appropriate adjustments in order to bring the swimmer closer to the target pose (Section 4.3). The overall system architecture is depicted in Figure 2.

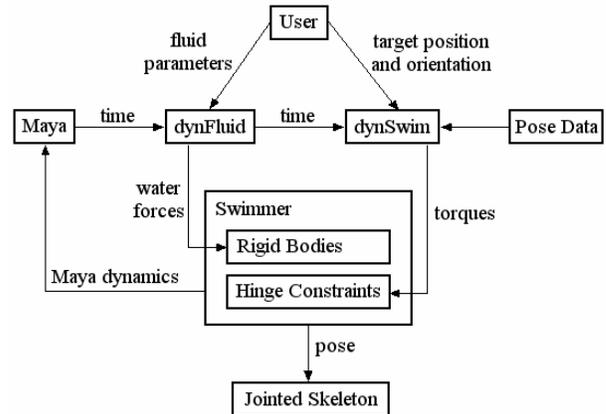


Figure 2 – System architecture

4. Muscle Model and Directional Control

Using the high-level input parameters describing the target position and orientation, the swimmer can be made to pass through a set of target points in the 3D world. Using a layered approach, we define three abstraction levels in order to control the swimmer:

- *Muscle Control Layer*: this is the lowest level, and it consists of the control law and joint torques that place the swimmer through the motions of a swimming stroke in order to thrust the body forward.
- *Primitive Rotation Layer*: this middle layer consists of basic primitive movements (rotate in the X-axis, rotate in the Y-axis, rotate in the Z-axis) that are performed by slightly modifying the stroke being performed in the Muscle Control Layer.
- *Trajectory Layer*: this high-level layer takes the user-defined target position and orientation, and drives the swimmer towards this goal using the functionality of the previous two layers.

4.1 Muscle Control Layer

While our system is general enough to handle any type of stroke, we currently focus on the breaststroke. Figure 3 shows a professional swimmer going through the motions of a breaststroke in the forward direction [8]. The various phases of the breaststroke are shown in Figure 4, with corresponding stick figures and delimiting instants [15].

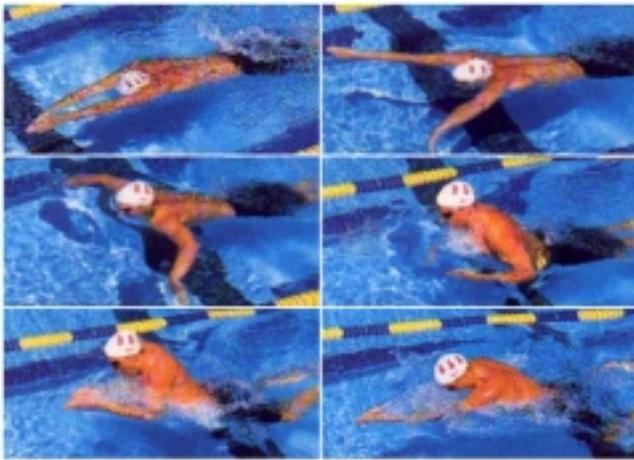


Figure 3 – Swimmer performing a breaststroke (left to right, top to bottom)

PHASE	DELIMITING INSTANT	CONTOUR	STICK
LEGS SPREADING	1. begin legs spreading I.		
	2. maximal legs spreading II.		
1 ST LEGS SQUEEZING	3. 0.08s later		
2 ND LEGS SQUEEZING	4. legs parallel to each other III. most dome-shaped body position		
	6. begin wrists deepest		
1 ST ARMS SPREADING	6. arms parallel to each other		
2 ND ARMS SPREADING	7. fingers highest or 1/2 in time spreading IV. most S-shaped body pos.		
	8. maximal arms spreading		
1 ST ARMS SQUEEZING	9. 1/2 arms squeezing		
2 ND ARMS SQUEEZING	10. wrists = shoulders width V. most cambered body position		
1 ST RECOVERY	11. knee 90° VI. most tilted trunk position		
	12. elbow 90° VII.		
2 ND RECOVERY			

Figure 4 – Phases of the breaststroke

Based on these actual breaststroke states from the biomechanics literature, we define a pose control graph (PCG) as described in [7]. The PCG, which is basically a finite state machine, specifies a set of desired joint angles for each of the swimmer’s hinges, as well as timing and transition information. This provides a convenient way to specify the torques that must be applied to our rigid body skeleton through the phases of our breaststroke.

In other words, we have a set of poses defining our swimming motion, but they define *desired* joint angles rather than *actual* joint angles. Proportional-derivative (PD) servos then make use of these desired joint angles at

each time step to compute the output torque τ required for each joint, based on the following control equation:

$$\tau = -k_p(\theta - \theta_d) - k_d(\dot{\theta})$$

where θ_d defines the desired joint angle, θ defines the actual joint angle, $\dot{\theta}$ defines the current angular velocity of the joint, and k_p and k_d represent, respectively, the proportional and derivative spring constants for the joint. Using this per-joint PD control, we can then perform full-body pose control using the pose sequence defined in the PCG, with the output torques being applied to our rigid body swimmer.

The breaststroke PCG is shown in Figure 5. Note that while we represent our breaststroke PCG as a continuous cycle, a PCG in general does not have any transition limitations. Currently, our PCG is defined by setting poses using the jointed skeleton, and then saving the desired joint angles to a pose file using a custom MEL script. Since our rigid body skeleton is hinge-based, we must make some simple approximations to the breaststroke due to our limited degrees of rotational freedom, most notably during the insweep/arm squeezing phase. Figure 6 shows the current breaststroke poses for our hinged rigid body.

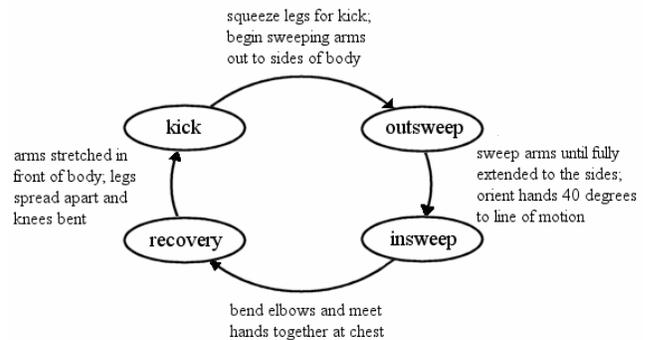


Figure 5 – Breaststroke PCG

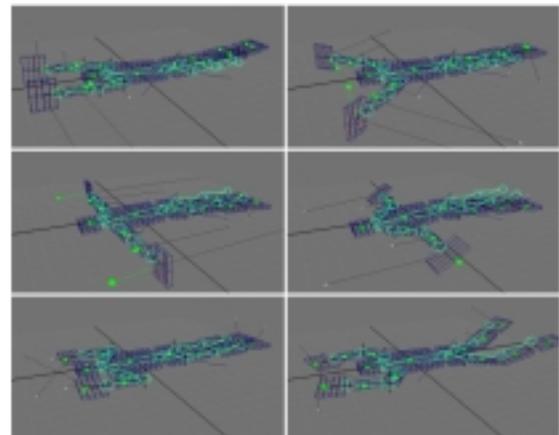


Figure 6 – Rigid body breaststroke

4.2 Primitive Rotation Layer

Using the symmetric rigid-body breaststroke as a starting point, we place the joint angles for each pose/state in a matrix M , defined as follows:

$$M = \begin{bmatrix} \theta_x^1 & \theta_y^1 & \theta_z^1 & \cdots & \theta_x^n & \theta_y^n & \theta_z^n \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \theta_x^1 & \theta_y^1 & \theta_z^1 & \cdots & \theta_x^n & \theta_y^n & \theta_z^n \end{bmatrix} \begin{matrix} \text{pose 1} \\ \vdots \\ \text{pose k} \end{matrix}$$

where $\theta_{x,y,z}^i$ represents the angle for joint $i = [1..n]$, and each k -th row of the matrix represents the k -th pose/state of our PCG [7].

We then define six minor modifications to the poses that make the swimmer rotate in each of the three rotation axes (in both the positive and negative directions). These poses are currently defined by hand using the jointed skeleton within Maya, similar to how the original symmetric breaststroke poses were defined. Given these modified poses, a new matrix M_j is defined, $j = [1..6]$, and a perturbation matrix M_j^p is computed for each:

$$M_j^p = M_j - M$$

The elements of the perturbation matrices represent the desired joint angle deltas between the original stroke and the turning strokes.

Swimmer rotation is accomplished by applying a sum of the various perturbations at each time step to the desired joint angles used for pose control, based on how much we would like to rotate. Since each perturbation matrix applies a fixed delta value to each desired joint, we can adjust the rotation magnitude simply by scaling each perturbation matrix by some factor k . Note that scaling the turning perturbation only works if the swimmer's motion changes reasonably with k . With a few test runs, we found that $k = [0..2]$ is a valid range for our current rigid-body breaststroke in each of the rotation axes.

4.3 Trajectory Layer

Now that we have some primitive rotations defined, any type of high-level path planning algorithm could be implemented, along with collision avoidance and route efficiency parameters. In our current system we implemented a simple targeting system whereby the swimmer continuously attempts to move closer to a user-defined target object. In other words, given the swimmer's current position and orientation, the targeting system attempts to determine which direction to rotate in order to get closer to the target position. Since the poses are defined such that forward progress is continuously made, the swimmer moves closer and closer to the target.

We have also implemented some simple interactive control for the swimmer. Within the Maya environment, we define a set of hotkeys for the various directions we'd like to navigate the swimmer. The target object is then automatically positioned in front of the swimmer in the current

forward direction. Each rotation hotkey then modifies the position of the target object in the desired direction. The target object then returns to its natural position (in front of the swimmer's current forward direction) over a few seconds. In other words, if the user presses a rotation key just once and then releases, the swimmer will rotate slightly in the appropriate direction, then level off and continue to move forward. If a rotation key is held down for a few seconds, the target object will remain offset from its natural position for a longer period of time, thus causing the swimmer to continue rotating in the desired direction until the hotkey is released.

5. Fluid Dynamics

The forward motion of the character is based on simple laws of physics. When the character performs a stroke, the arm motion displaces a volume of water. The inertia of the displaced water creates a reaction force and propels the character to move. Assuming the movement of the stroke is relatively slow (the ratio of inertia forces to viscous forces, the Reynolds number, is ≤ 1), we can express the reaction force using Stokes law [3, 4]:

$$F = |V| * A * \eta$$

where V is the relative velocity of the motion to the fluid and $|V|$ is its magnitude, A is the cross-section area, and η is the viscosity. The resulting force is in the opposite direction of the relative velocity V .

The force is calculated on a per polygon basis on the rectangular rigid bodies. The force is then decomposed into translational force and torque about the center of mass of the object. The total translational force and torque is then summed up and applied to the Maya rigid bodies in the form of impulse and spin impulse.

The fluid does not have to be at rest. The user can define some function to describe the flow of the fluid. Currently, the fluid parameters available to users include spatial position and time. This allows the user to define the flow using some simple differential equation. However, for simplicity, the fluid is currently static. That is, its interaction with the character does not change its flow.

In our current implementation we have developed a visualization technique for the fluid forces acting on the swimmer. We draw vectors representing the total fluid drag force acting on a particular body segment as a green line attached to a circle. The length of the vector shows the magnitude of the force and its direction corresponds to the force's direction (Figure 6).

6. Results

In order to validate that the various parameters affect the performance of the swimmer as expected, we perform a series of tests by modifying the body/environment parameters and then compute the displacement of the swimmer for each. Note that Maya currently does not document the units

in which rigid body masses are defined in. As a result, we make the assumption they are in pounds, and our rigid body parts are thus assigned masses such that the total mass of the swimmer is approximately two-hundred pounds. We currently use the proportions listed in Table 1 for our rigid body parts, based on data from the biomechanics literature.

Table 1 – Mass proportions for rigid bodies

Rigid Body	Human Mass %	Simulation Mass %
Head	8% (Head and Neck)	8%
Torso	68% (Total)	49%
Pelvis		19%
Right Upper Arm	3% (Total Arm)	1%
Right Lower Arm		0.6%
Right Hand		1.4%
Left Upper Arm	3% (Total Arm)	1%
Left Lower Arm		0.6%
Left Hand		1.4%
Right Thigh	9% (Total Leg)	5%
Right Shin		3%
Right Foot		1%
Left Thigh	9% (Total Leg)	5%
Left Shin		3%
Left Foot		1%

The effect of not knowing the correct mass units is that the useful ranges of our other parameters are skewed from realistic ranges (most notably our viscosity).

In each of the following cases, the parameter being tested was sampled uniformly, while the remaining parameters remained at their default values (including the k_p and k_d parameters for each joint, which correspond to muscle strengths, as well as body mass). Using Maya’s *playblast* option, the swimmer was then put through the motions of the symmetric forward breaststroke for 200 frames (approximately 7 seconds at NTSC quality). The default viscosity parameter is 0.2, and the default hand dimensions are defined by the artist within the Maya environment.

6.1 Effect of Viscosity

Figure 7 plots the swimmer displacement against fluid viscosity ranging from 0.029 to 0.2. As expected, at low viscosities the swimmer has a difficult time making forward progress since there is less drag force (similar to trying to swim in air). When viscosity is increased, the swimmer’s strokes become more effective, propelling him through the fluid. As viscosity continues to increase this effect is reduced due to the increased drag during the insweep and recovery phases of the stroke. Additionally the swimmer may struggle to keep up to the desired poses since higher

viscosity slows his movements while muscle strength remains constant.

Our current system cannot handle significantly larger viscosity values due to numerical “blow-ups” as a result of the larger errors in the PD control. However, it is worth making some theoretical predictions regarding displacement at viscosity values beyond the range shown in the graph. Ultimately, we expect to see displacement gradually drop down to zero and then remain at that level, since there would be a critical point after which the constant muscle strength can no longer thrust the rigid bodies through the highly viscous fluid.

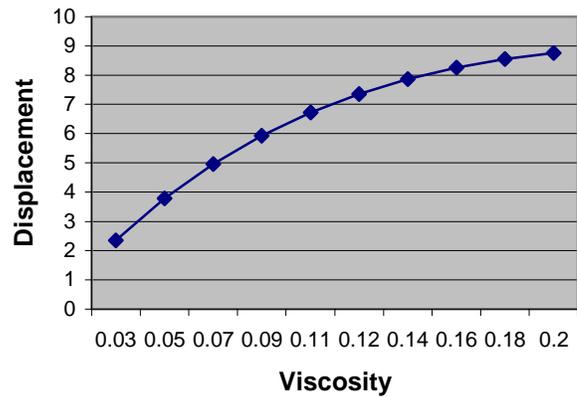


Figure 7 – Plot of viscosity vs. displacement

6.2 Effect of Hand Dimensions

Our current system implementation visually depicts the effect of water forces on the various rigid bodies within Maya, and for the breaststroke it was observed that the largest forward thrust was achieved during the outswEEP phase of the stroke. As a result, we tested the effect of changing the dimensions of the hand to see what effect, if any, they had on the swimmer’s performance. This corresponds to modifying the cross-section area of the body interacting with the fluid (Section 5).

Figure 8 plots the displacement of the swimmer against changing hand dimensions (via unit deltas in hand width and hand length). For the hand width and hand length changes, the area of the hand against the fluid throughout the outswEEP motion increases, causing more drag and thus more forward thrust. Intuitively, there would eventually be a drop off in increasing forward thrust, since muscle strength (torque) at the joints remains constant but there is increased drag against the rigid body that must be overcome. Both of the experiments increase the hand area by the same amount, but hand length lends the swimmer greater propulsion due to the increase in torque provided by the longer arms with higher velocities at the tip of the hand.

The displacement due to hand width falls off slightly due to pitching of the entire figure caused by the extended hands.

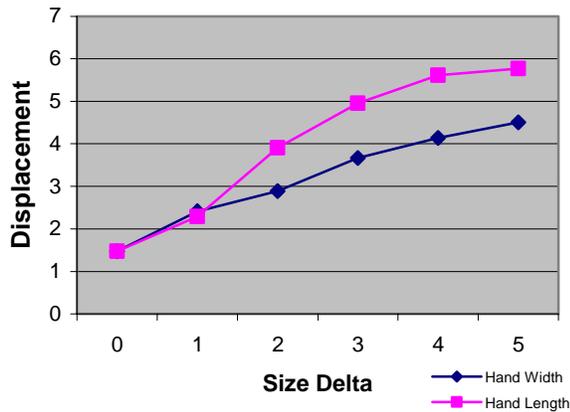


Figure 8 – Plot of viscosity vs. hand dimensions

6.3 Muscle Strength and Mass Settings

The k_p and k_d spring constants for the joints, as well as the rigid body masses, could also be modified to vary the performance of the swimmer. However, there does not seem to be any science when choosing appropriate values for these parameters. Slight modifications of our current masses and constants results in large oscillations of the rigid bodies during PD control (due to large errors between the desired and actual joint poses). Admittedly, significant manual tweaking of these parameters was required in order to generate stable simulation results.

6.4 Comparison with Real Swimmers

The main goal of this work was to determine whether natural and realistic swimming strokes were possible to simulate using rigid body dynamics. Figure 9 shows images from video footage of a real swimmer performing a breaststroke placed side-by-side with rendered images of our rigid body swimmer performing the same stroke. As can be seen, the resulting motion is fairly good visually, but does not always accurately mimic the real-life swimmer, most notably during the insweep phase of the stroke (since our hinged swimmer has only a single degree of rotational freedom at the shoulder joint). Additionally, during the kick phase of the breaststroke, a real swimmer typically rotates the hips in both the X and Y axes, whereas our swimmer can only rotate hips in the Y axis, reducing the propulsive effect of the overall kick. Finally, forward thrust is relatively low in comparison to the real-life swimmer's displacement, largely due to our simple fluid model that assumes slow moving bodies. Increasing the mass of the swimmer could potentially help increase forward momentum, but this would require additional corresponding tweaking of the joint spring constants.

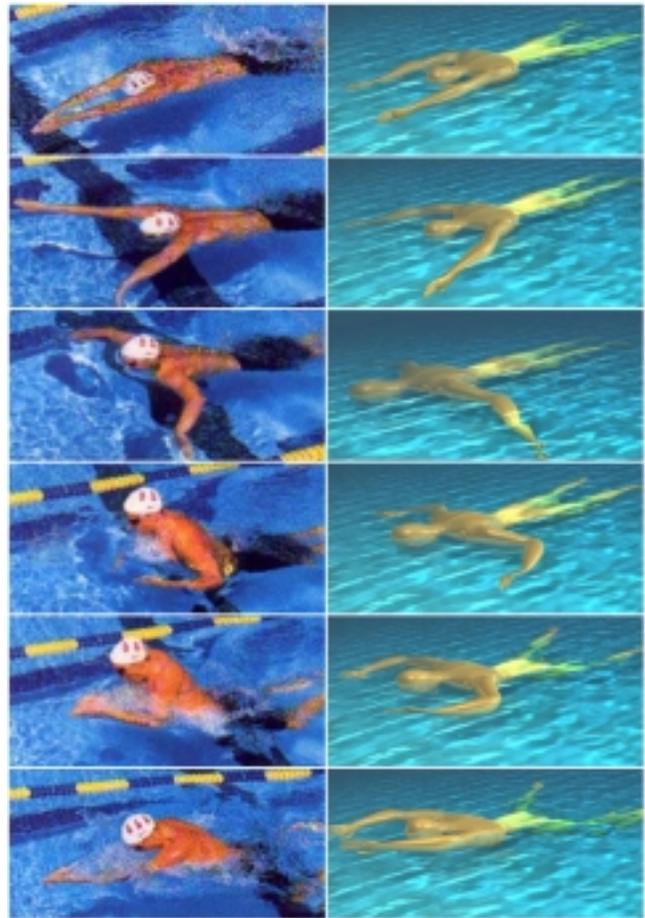


Figure 9 – Images of real and simulated breaststrokes

7. Discussion and Future Work

Clearly, one of the main limitations of our system is the use of hinge constraints instead of pin (ball and socket) constraints for the various joints of our swimmer. As a result, the range of realistic swimming strokes we can simulate is greatly reduced as described earlier. The use of hinge constraints for all joints was a result of limitations within the Maya dynamics system. Currently, Maya only allows applying torques (spin impulses) at the centre of mass of a rigid body, instead of at the actual constraints. Therefore, we were required to implement our own joint torque system within the Maya environment. As a result, implementing and thoroughly testing this using hinges instead of pin constraints simplified the process. The other option would have been to use an existing dynamics library (or implement our own) with the desired functionality instead of using Maya's dynamics system, but at the cost of losing other useful dynamics features that Maya already provides (emitters, gravity fields, etc.) Another option is to extend the current hinged system to support higher degrees of freedom on joints by using multiple axis-aligned hinges in close proximity to each other. For example, at the shoul-

der, we could define three orthogonal hinges connecting extremely small rigid bodies together. Then PD control could be performed individually on each of these hinges independently, providing us with a close approximation to pin constraints with the added ability to apply torques at the joints.

Another limitation of our current system is the assumption that the Reynolds number of our rigid bodies in fluid will be less than or equal to 1. This prevents us from generating fast, powerful strokes with complex drag forces (Reynolds number greater than one). If we could implement such a system, it would allow the swimmer to experience significant forward thrust that would more closely resemble a real swimmer in water. Currently, due to our Reynolds assumption, the swimmer's continuous forward acceleration is not as significant as it would be in reality. An alternative system would dynamically compute the drag coefficients for body parts as the fluid passes by them. Most fluid dynamics textbooks provide experimental data on drag coefficients for simple 2D and 3D shapes. We may be able to apply this data to compute the drag of a body part as it changes its orientation in the fluid.

In our current implementation, the PCG is defined by manually orienting the jointed skeleton into desired poses, and then saving the joint angles out to a data file. One interesting area of future work is using vision-based motion capture techniques to both automate the PCG acquisition process, as well as make our poses more closely resemble actual swimmers. Traditional motion capture using magnetic or marker-based sensors is difficult underwater, and thus vision techniques seem to be the most promising approach, but existing methods fail to handle other key issues such as self-occlusions, underwater visual distortions, and precise tracking of constant skin tone areas [13].

Currently, swimming coaches have to rely on simple video footage recorded at fixed camera positions, or extremely expensive fluid flow equipment that is attached to a swimmer's body [11], in order to analyze the performance of professional swimmers. The ultimate goal of a physics-based swimming animation system would be to have the interaction between the fluid and swimmer behave so realistically that swimming instructors and coaches could use the system as a tool to analyze the effects of various stroke techniques on swimmer performance. Our current system is quite far from being used for such purposes, but with a more advanced fluid model that takes swimmer interaction into account as well as a detailed rigid body swimmer model with more rotational freedom, such a tool should be quite achievable.

8. Conclusion

In this paper, we outlined a system to animate humanoid swimming using a rigid body muscle control system and a simple fluid dynamics model. The system is shown to pro-

duce fairly nice visual results, but realism is limited due to the use of hinge joints instead of more general ball and socket joints. Directional control is implemented using a layered approach, allowing the swimmer to navigate through the fluid algorithmically or interactively via user input.

Acknowledgments

A huge thank you goes to both Joe Laszlo and Karan Singh for the fruitful discussions and myriad of ideas they provided regarding this project.

References

- [1] F.P. Beer, E.R. Johnston, Jr. *Vector Mechanics for Engineers*. WCB McGraw Hill, Sixth Edition, Boston (1996).
- [2] M. Berger, A. Hollander, G. De Groot. "Determining propulsive force in front crawl swimming: A comparison of two methods". In *Journal of Sports Sciences*, 17:97-105 (1999).
- [3] Y. Cengel, R. Turner. "Fundamentals of Thermal-Fluid Sciences". McGraw Hill, New York (2001).
- [4] A.J. Chorin, J. E. Marsden. *A Mathematical Introduction to Fluid Mechanics*. Springer-Verlag, New York (1979).
- [5] A. Craig, D. Pendergast. "Relationships of stroke rate, distance per stroke, and velocity in competitive swimming". In *Medicine & Science in Sports*, 11(3):278-83 (Fall 1979).
- [6] J. Hodgins, W. Wooten, D. Brogan, J. O'Brien. "Animating Human Athletics". In *Proceedings of ACM SIGGRAPH*, pp. 71-78 (1995).
- [7] J. Laszlo. "Controlling Bipedal Locomotion for Computer Animation". M.A.Sc. Thesis, University of Toronto, 1996.
- [8] T. Laughlin. "Breaststroke Breakthrough". Web link: http://www.burlingameaquatics.com/age_swim/swim_corner/breaststroke.htm
- [9] M. van de Panne, E. Fiume. "Sensor-actuator networks". In *Proceedings of ACM SIGGRAPH*, pp. 335-342 (1993).
- [10] B. Ramakrishnananda, K. Wong. "Animating Bird Flight Using Aerodynamics". *The Visual Computer*, 15:494-508 (1999).
- [11] S. Riewald. "Designing the Optimum Stroke". Article in *Fluent Newsletters* (Spring 2000).
- [12] K. Sims. "Evolving Virtual Creatures". In *Proceedings of ACM SIGGRAPH*, pp. 15-22 (1994).
- [13] C. Sminchisescu. "Estimation Algorithms for Ambiguous Visual Models". Ph.D. Thesis, MOVI Group, INRIA, 2002.

- [14] J. Stam. "Stable Fluids". In Proceedings of ACM SIGGRAPH, pp. 121-128 (1999).
- [15] J. Troup. "The Physiology and Biomechanics of Competitive Swimming". Clinics in Sports Medicine, Volume 18, Number 2, April 1999.
- [16] X. Tu, D. Terzopoulos. "Artificial Fishes: Physics, Locomotion, Perception, Behavior". In Proceedings of ACM SIGGRAPH, pp. 43-50 (1994).
- [17] J. Wejchert, D. Haumann. "Animation Aerodynamics". In Proceedings of ACM SIGGRAPH, pp. 19-22 (1991).
- [18] A. Witkin, D. Baraff. "Physically Based Modeling: Principles and Practice". ACM SIGGRAPH Course Notes (1997).