ANIMATION AERODYNAMICS

JAKUB WEJCHERT and DAVID HAUMANN

European Visualization Centre
IBM Scientific Centre, Winchester
Hampshire SO23 9DR, England.

IBM Research
T. J. Watson Research Center
Yorktown Heights, New York 10598, USA.

Abstract

Methods based on aerodynamics are developed to simulate and control the motion of objects in fluid flows. To simplify the physics for animation, the problem is broken down into two parts: a fluid flow regime and an object boundary regime. With this simplification one can approximate the realistic behaviour of objects moving in liquids or air. It also enables a simple way of designing and controlling animation sequences: from a set of flow primitives, an animator can design the spatial arrangement of flows, create flows around obstacles and direct flow timing. The approach is fast, simple, and is easily fitted into simulators that model objects governed by classical mechanics. The methods are applied to an animation that involves hundreds of flexible leaves being blown by wind currents.

Keywords: Animation, Simulation, Aerodynamics, Fluid Mechanics, Flow Primitives, Control, Motion Design, Wind, Leaves. **CR Categories** I.3.5, I.3.7, I.6.3, J.5.

INTRODUCTION

Every year leaves fall from trees and gather on the autumn ground; winds blow and scatter them in currents, whirlpools and eddies. This charming motion is a consequence of aerodynamics: the description of fluid flow and its relation to the motion of solid objects.

An Aerodynamic Model with Control: We describe a fast aerodynamic way of *modelling* and *controlling* the motion of many flexible objects in fluid currents in 3D. Physics or engineering applications would require numerical solutions of fluid flow with immersed solid objects. Because animation has less stringent accuracy requirements, we can avoid computational expense by dividing the system into two parts: a linear flow regime and an object boundary regime. The first ranges over all space and the second is used in the close vicinity of objects. In the linear flow regime we use the analytic solutions of the equations *instead* of solving for the flow numerically. These solutions define a set of flow primitives which are given as fluid velocity fields. Solutions such as vortices, sinks and uniform flows, can be linearly mixed so as to create a complex flow scenario. The primitives enable the design and control of animation sequences. The second part of the model is the interaction between the flow and the objects. This is based upon simplified boundary effects, that describe the forces exerted on object surfaces. Once the forces acting on objects are known, object motion is governed by Newtonian mechanics.

Relevant Models: Particle based systems have mimicked the visual appearance of fire [12], waterfalls, falling snow [13], and viscous jets [7]. Although these models have produced stunning effects they can only account for particle-like objects. Simulations of elasticity [11, 5, 8], have displayed the flexibility of individual objects; related models could exhibit the flapping motion of flags in uniform wind fields [14, 5]. Animation models of liquids, such as ocean foam [4] and shallow water [6] have displayed the visual appearance of liquid surfaces, but are not useful for representing internal fluid currents. Usually, models of natural phenomena are too complex to be applied by an animator using traditional techniques; a number of researchers have addressed this. Pintado [10] describes an approach that allows the control of object motion in non-physical 2D fields. Other methods allow animations to be controlled by geometrical constraints or optimization [11, 2]. However, these techniques can be numerically intensive and become unwieldly for controlling collections of objects with many degrees of freedom. In summary, the above models do not explicitly address the simulation of many flexible objects in dynamic fluid flows, combined with a fast control method.

LINEARIZED FLUID FLOW

The mechanics of a fluid can be described by the Navier Stokes equation [3, 9]. This can be simplified in the case of a fluid that is A) inviscid, B) irrotational ($\nabla \times \mathbf{v} = 0$) and C) incompresssible ($\nabla.\mathbf{v} = 0$). This is a reasonable model for air at normal speeds when it does not exhibit turbulence [1]. Here v, is the velocity field of the fluid, describing the magnitude and direction of the flow at every point. The simplified fluid satisfies the Laplace equation

$$\nabla.\mathbf{v} = \nabla.\nabla\phi = \nabla^2\phi = 0. \qquad (1)$$

The velocity field is given by the gradient of the scalar potential $\mathbf{v} = \nabla\phi$. Since (1) is a linear differential equation, if we find any two analytical solutions then their linear combination is also a solution; the application of boundary conditions then results in a *physical* solution. Typically, it is required that the flow should be A) uniform at infinity and B) have no normal component at obstacle boundaries [9].

ACM-0-89791-436-8/91/007/0019 $00.75

Since solutions are analytic, we bypass the task of solving the fluid equations numerically and provide a fast and simple technique for creating flows for animation.

Flow Primitives: We call a velocity field that satisfies eq (1) and the boundary conditions, a *flow primitive*. Given a set of flow primitives, an animator can construct more complicated flows from these building blocks, in a manner similar to that used by Sims [13]. In fact, the primitives provide a physical basis for the "velocity operators" that he used to direct particle systems. Our simplest primitive is *uniform* flow: the velocity lines follow straight lines. Other solutions include source, sink and vortex flows. A *source* is a point from which fluid moves out in all directions; a *sink* is a point to which fluid flows uniformly in all directions and disappears; and fluid moves around a *vortex* in concentric circles (fig 1). Using cylindrical coordinates, the potential and the velocity field for a line of source at the origin, with strength a, is:

$$\phi = \frac{a}{2\pi} \ln r; \quad v_r = \frac{a}{2\pi r}; \quad v_\theta = 0; \quad v_z = 0. \qquad (2)$$

For a sink the constant a is set negative. A vortex at the origin with strength b is given by:

$$\phi = \frac{b}{2\pi} \theta; \quad v_r = 0; \quad v_\theta = \frac{b}{2\pi r}; \quad v_z = 0. \qquad (3)$$

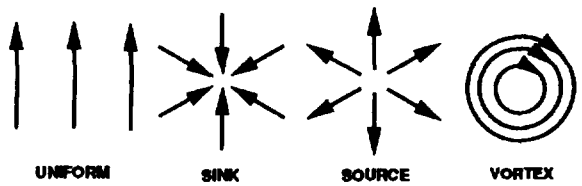

Figure 1. A schematic diagram of the flow primitives.

Addition of Flows: Because the system is linear (and because of a uniqueness theorem) if a flow satisfies eq (1) and has the required properties on object boundaries and at infinity, then it is the correct solution [1, 9]. Thus (as in aerodynamics) we can add the primitives to create more complicated flows:

$$V = v_{vort}(x,y,z) + v_{sink}(x,y,z) + v_{source}(x,y,z) + ... \qquad (4)$$

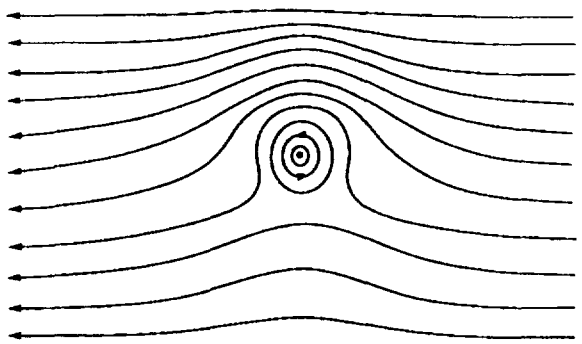

Figure 2. The addition of uniform and vortex flow.

Figure 2 shows the flowlines that result from the addition of a uniform flow with a vortex. The flow defines the whole temporal path of the fluid at the beginning, middle and end of the motion. Since the positions and strengths of the primitives can be chosen, the approach allows for a simple, physically-motivated way of designing the paths of objects. Once objects are placed in the fluid, their trajectories have already been determined by the user to a first approximation.

Flow Obstacles: We can also use flow primitives to design flows around large solid obstacles, and to bound the spatial extent of flows. Obstacles can be built out of primitives that are strong enough to cause a main flow to be directed from certain regions. Similar methods are used to study the flow around obstacles such as airfoils [1]. Figure 3 shows the effect of adding together a uniform flow with a point source. This can be taken to represent flow around a solid object. No fluid flows across the "stagnation" flow line shown in bold, so if a solid object with the geometry of the stagnation curve were placed in the fluid, there would be no flow across its surface. This approach is faster than normal collision detection algorithms and allows the smooth and natural motion of the objects as they interact with obstacles. The method was used to create the motion of leaves around obstacles such as slides or walls.

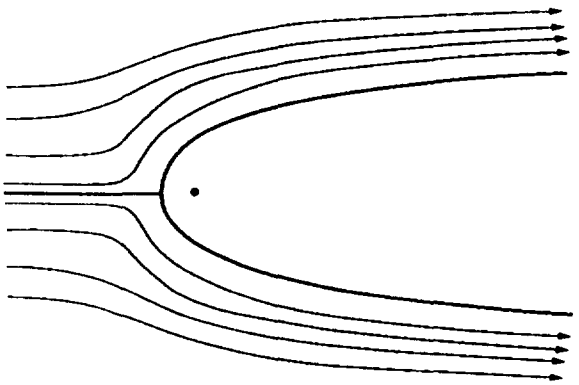

Figure 3. Creating solid obstacles to flow using the addition of primitives.

Time-dependent Flows: We can also model time-dependent flows with the condition that changes to the primitives are directed by forces that are external to the system (user specified). Although the time evolution of the forces may not be physically based, the resulting flow at each frame will be. Time-varying fields enable a user to change the flow lines, by directing the positions of the primitives with time. This gives a further degree of control, allowing obstacles to move and events to occur at specific times. Coupled with bounded fields, it enables the control of *collections* of objects to follow specified paths.

OBJECT BOUNDARY REGIME

Dividing the system into two regimes, linear flow and boundary layer, simplifies our general problem. For the major part fluids are taken to behave as a linear inviscid system; however, in the vicinity of objects we must include boundary effects such as viscous drag and pressure. In this way forces exerted on the objects may be calculated.

Particles in Flows: A model for particles in flows can be based on the Stoke drag equation. This gives the force exerted on a spherical particle with radius a, moving with relative velocity v' in a fluid with viscosity η as:

$$F = 6\pi a \eta v'. \qquad (5)$$

Given a mass particle with velocity **p**, the relative velocity with respect to a fluid velocity field **v** is: $v' = v - p$. So from eq (5) we define the force on the particle as

$$\mathbf{F} = \alpha \mathbf{v}^r. \tag{6}$$

Here α represents a coupling strength between the flow and the particles. Particles not moving at the fluid velocity will experience adjustment forces until they do so. For α large, particles will be forced to track the flow closely, as they would in a viscous fluid (η large). If $\alpha = 0$ then the fluid fields have no effect on the particles. The parameter α is similar to the 'field affinity' parameter used to direct particles along 2D spline fields by Pintado [10].

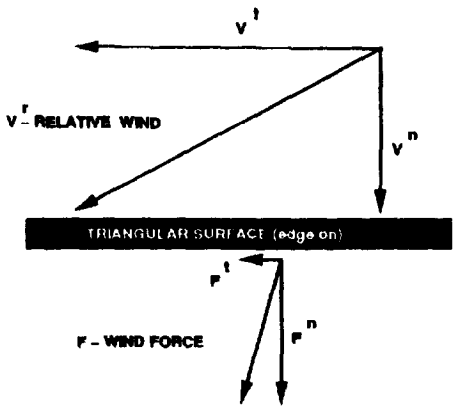

Figure 4. Side view of a triangular area in a fluid.

Objects in Flows: Unlike the case of particles, the forces acting on a surface depend on its area and orientation with respect to the flow. Surfaces defining an object are divided into triangular patches with a mass point at each vertex. The relative velocity of each particle is resolved into the normal and tangential components with respect to the triangular surface: $\mathbf{v}^r = \mathbf{v}^n + \mathbf{v}^t$ (see fig 4). The normal component of the force is due to pressure difference between the front and rear of the surface. It can be shown that the force of a uniform flow with speed v and density ρ, that strikes a flat surface of area A, is given by:

$$F = \rho A v^2. \tag{7}$$

The tangential force component is due a fluid with viscosity moving across a surface. This is given by the viscous shear stress times the area.

$$F = A\eta \frac{dv}{dy}, \tag{8}$$

where y is measured perpendicularly from the object surface into the fluid. For a non-slip condition we have at $y = 0$, $v = 0$ and for $y \to \infty$, $v \to v^t$. Typically the velocity profile is parabolic, but in the vicinity of the surface we may take the velocity gradient to be linear: $F \sim A\eta v^t$. Therefore we write the normal and tangential forces as:

$$\begin{aligned} \mathbf{F}^n &= \alpha^n A v \mathbf{v}^n, \\ \mathbf{F}^t &= \alpha^t A \mathbf{v}^t. \end{aligned} \tag{9}$$

$\mathbf{F}^n$ is the force experienced by a surface facing into the fluid, while $\mathbf{F}^t$ is due to the viscous drag of fluid flowing across the surface. This may be interpreted as a generalization of eq (6): a set of physically based dynamic control equations, that determine the degree to which objects follow the fluid.

Overall Approximations: In our model we have chosen a balance between physical exactness, execution speed and control. For example, linearized air flow cannot exhibit turbulence but if we used a non-linear system: a) mixing flow primitives would give non-physical solutions, b) it would be numerically intensive. Objects in wind exhibit complex motion mainly due to their geometry and the fluid-object interaction (little is due to the local turbulence of the fluid itself) so using a linear fluid is not unreasonable. It should also be understood that dividing the system into two parts results in the flows affecting the objects and not *vice versa*. This holds better for small objects, spaced relatively wide apart.

APPLICATION

Simulation: We integrated the methods into a simulator developed by Norton [8], that models the flexibility and fracture of solids. Objects are constructed of masses and springs governed by Newtonian mechanics. The evolution of a collection of objects with time is carried out by integrating $\mathbf{F} = m\mathbf{a}$. $\mathbf{F}$ is the total force acting on a mass, made up of contributions from gravity, spring stretching *and* the external fluid forces:

$$\mathbf{F} = \mathbf{F}_{grav} + \mathbf{F}_{spring} + \mathbf{F}_{fluid} + \ldots. \tag{10}$$

This determines the accelerations from which the extrapolated velocities and new positions can be calculated.

Motion Design: To design an animation of a collection of objects (leaves) being blown by wind, we: 1) Design leaf geometries and construct them out of masses and springs. 2) Design a set of wind fields that will define the motion paths of the objects during a sequence. 3) Simulate the motion and preview the results. 4) If needed, make changes to the wind velocities, fluid-object interaction, position of flow primitives, and the number of objects.

Object Geometry: Our first test leaves were point particles, whose motion in air was directed by eq (6). These were useful for seeing the overall motion of collections of leaves in air currents. To exhibit individual rotational motion, a leaf was built out of masses and springs using the geometry of six triangles. The leaf was duplicated with slight variations in geometry, mass distribution and stiffness. Even in a uniform flow objects glide and twirl in a realistic way because of the different forces experienced in the lateral and tangential directions due to eq (9).

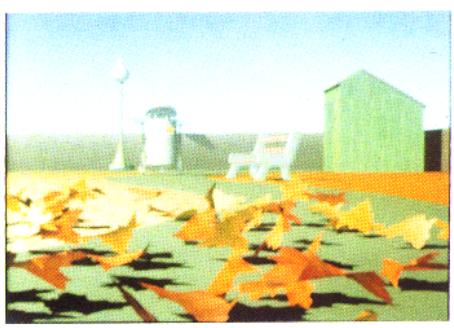

Figure 5. Leaves being chased by a garbage bin.

Adding Flow Primitives: By combining field primitives (eq (4)), whole motion paths could be designed. In an animation sequence a garbage bin chases and then inhales leaves trying

to escape. Uniform and vortex flows made leaves travel along the ground (fig 5) and then fly and twirl up into the air (fig 6). Finally, the leaves were funneled in by a cyclone consisting of vortex and sink primitives coincident with the bin mouth (fig 7).

Flow Obstacles: Mixing and positioning flow primitives, enabled us to build large flow obstacles around which leaves would travel. An animation sequence required leaves to be blown up a slide. To do this, a uniform field was used as the main driving flow and leaves then passed over an obstacle wedge made out of fields that prevented the flow lines from penetrating the slide geometry. This enabled the smooth motion of leaves blowing over the solid obstacle (fig 8).

Figure 6. Leaves escaping by flying in the air.

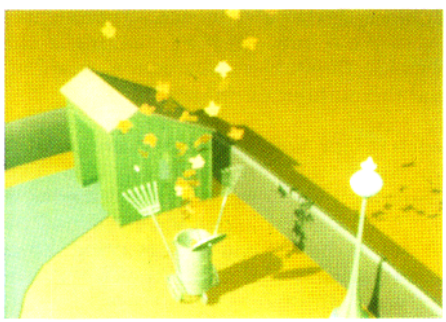

Figure 7. Leaves being sucked in by a garbage bin.

CONCLUSION

We have described a model based on aerodynamics that can be used for object motion simulation and control. In the model we chose a balance between physical exactness, execution speed and animation control. Future extensions could include aerodynamic flight of birds or airplanes. In practice, the application of the methods are quite simple. We hope that the methods can be integrated and extended by others into their physically based simulation systems.

We wish to give credit to the members of the Animation Systems Group: Alan Norton, Bob Bacon, Paula Sweeney, Kavi Arya, Al Khorasani and Jane Jung; and to Mike Henderson for discussions on fluid mechanics. Thanks to the staff at Winchester for encouragement and support, and to Roz for proof reading.

Figure 8. Leaves being blow up and over a slide.

REFERENCES

1. Anderson J (1985) "Fundamentals of Aerodynamics", McGraw-Hill Publishers.
2. Barzel R, Barr A (1988) "A Modeling System based on Dynamic Constraints", Computer Graphics (SIGGRAPH 88 Proceedings) 22 (4) 179.
3. Feynman R, Leighton R, Sands M (1965) "The Feynman Lectures on Physics", Addison Wesley.
4. Fournier A, Reeves W (1986) "A Simple Model of Ocean Waves", Computer Graphics (SIGGRAPH '86 Proceedings) 20 (4) 75.
5. Haumann D, Parent R (1988) "The Behavioral Test-Bed: Obtaining Complex Behavior from simple Rules" The Visual Computer 4 (6) 332.
6. Kass M, Miller G (1990) "Rapid, Stable Fluid Dynamics for Computer Graphics" Computer Graphics (SIGGRAPH 90 Proceedings) 24 (4) 49.
7. Miller G, Pearce A (1989) "Globular Dynamics: A Connected Particle System for Animating Viscous Fluids", Computers and Graphics, Vol 13, 305.
8. Norton A, Turk G, Bacon R (1990) "Animation and Fracture by Physical Modeling", To appear in "The Visual Computer".
9. Patterson A (1989) "A First Course in Fluid Dynamics" Cambridge University Press.
10. Pintado X, Fuime E (1989) "Grafields: Field-Directed Dynamic Splines for Interactive Motion Control" Computers and Graphics Vol 13, 77.
11. Platt J, Barr A (1988) "Constraint Methods for Flexible Models", Computer Graphics (SIGGRAPH 88 Proceedings) 22 (4) 279.
12. Reeves W (1983) "Particle Systems - A Technique for Modeling a Class of Fuzzy Objects" Computer Graphics (SIGGRAPH 83 Proceedings) 17 (3) 359.
13. Sims K (1990) "Particle Animation and Rendering Using Data Parallel Computation", Computer Graphics (SIGGRAPH 90 Proceedings) 24 (4) 405.
14. Terzopoulos D, Platt J, Barr A, Fleishcer K (1987) "Elastically Deformanble Models", Computer Graphics (SIGGRAPH 87 Proceedings) 21 (4) 205.