Interacting with Boids in an Incompressible Fluid Environment

Aytek Aman
Department of Computer Engineering
Bilkent University
aytek.aman@cs.bilkent.edu.tr

Ateş Akaydın
Department of Computer Engineering
Bilkent University
akaydin@cs.bilkent.edu.tr

Uğur Güdükbay
Department of Computer Engineering
Bilkent University
gudukbay@cs.bilkent.edu.tr

***Abstract*—With increasing power of the computing hardware, computer simulations are being realized in real-time. It is now possible to create real-life phenomena using computers. In this study, we present a crowd and fluid interaction environment, where fish in the environment depict flocking behaviour, affected by the flow of an incompressible fluid that can be manipulated by a Kinect controller. This system can be used in virtual reality applications and video games to provide immersive underwater scenarios. We developed our application on a commercially available game engine that provides real-time experience for the users.**

I. Introduction

Simulation of flocks is not a trivial task. Both psychological and physical factors affect the nature of the flocking behavior which makes creating realistic flock animation task difficult. In this paper, We incorporate velocity fields into the *Boids Model* first proposed by Craig W. Reynolds [1]. This allows us to experiment the behaviour of crowds under the influence of external physical forces like wind and water current. The approach we've taken also runs at real-time for thousands of boids. We also provide an interactive method to manipulate the fluid's velocity field using a Microsoft Kinect device [2].

II. Related Work

Craig W. Reynolds developed a computer model describing the coordinated motion of animals such as flocks of birds, or schools of fish. He named this model as the *Boids Model* [1]. Every individual animal in this model is called a boid. In Boids Model, there are three main force components which affect the motion of a boid. These are the separation, alignment and cohesion forces. Separation describes the force that keeps a boid away from its crowding neighbors so that they don't collide. The alignment force orients the boid towards the average direction of the neighboring boids. Finally, the cohesion force steers the boid towards the average position of the neighboring boids. For each boid in the system, these three driving forces are calculated separately and then a final velocity is calculated. That way completely deterministic yet naturally looking crowd simulation is created. Reynolds's approach has a number of extensions which incorporate other dynamic effects. For instance, Delgado-Mata et al. [3] studied the effects of fear on flock behaviour using an extended Reynolds model. Hemerlijk and Hildenbrandt [4] studied flocks of birds. They proposed a physically accurate model by incorporating the affects of fixed wing aerodynamics into the basic Reynolds model.

Other than crowd simulation, this work also focusses on computational fluid simulation. Almost all of the fluid simulation methods are based on a nonlinear differential equation called the Navier-Stokes equation. Analytical solution of this equation is available only for several, greatly simplified and constrainted (i.e. incompressible fluids) cases. Numerical approximations are hence more common in the field of Computer Graphics. Also a number of simplifications are done on the Navier-Stokes model to approximate the fluid behaviour in real-time. Jos Stam proposed a stable animation model for fluid-like objects [5]. Ronald Fedkiw et al. proposed a method for visual simulation of smoke [6]. Later, Jos Stam extended his previous model to cover real-time applications such as video games [7]. A comprehensive book on common fluid simulation techniques for Computer Graphics was written by Robert Bridson [8].

III. Proposed Approach

Our framework consists of three different components. The first component provides the interface between the user and the system. The second component is a fast and stable fluid solver based on the approach developed by Stam [5]. The third component is the flock simulator and it is responsible for driving the autonomous agents (fish in particular) around. In order to create flocking behaviour in a fluid environment we combine these three components together as seen in Figure 1.

To represent fluid velocity in the environment, we keep a three dimensional grid, where each grid cell has an associated fluid velocity. We call this grid as the cell-wise velocity field and denote it with the function $\vec{v}(\vec{i}, t)$. $\vec{i}$ is a vector which specifies the grid index in three dimensions and t is the simulation time. For each cell, the associated velocity represents the overall movement of the fluid at the center. The cell-wise velocity field is common to all three components and can be influenced by adding (or injecting) additional velocities. To compute $\vec{v}(\vec{i}, t)$, we used a three dimensional version of the Stam's approach described in his paper [5]. The extension of the Stam's approach to three dimensions is trivial.

The flow chart given in Figure 1 demonstrates main dependencies among the three components. For each frame, user input is gathered from Kinect controller [2], and then appropriate movement information is feeded to the fluid simulator. With the user input, fluid simulator updates its state. Then, agents in the simulation environment are simulated with respect to flocking rules and fluid motion. The following sections explain these three components in detail.

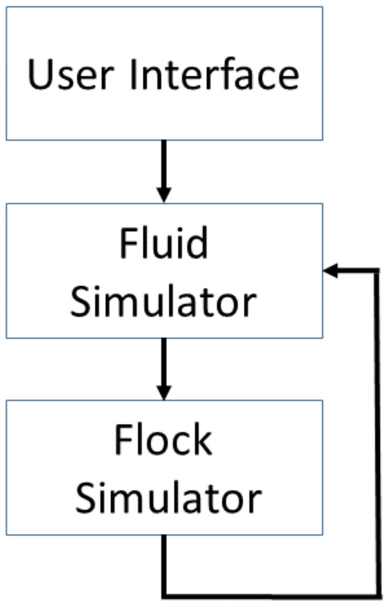


Fig. 1. The flow chart demonstrating the system components.

A. User Interface

User interaction in our framework involves localizing the user's hand position within the simulation environment. For simplicity we represent the hand as a semi-transparent spherical object centered at the hand's detected position. This object is also called the *Velocity Generator*. The position of the *Velocity Generator* is determined using the functions provided by the Microsoft Kinect API [2]. Velocity Generator tracks its own velocity across successive frames. At the beginning of each frame, random points are sampled inside the boundary of the Velocity Generator. Then, these points are used to inject velocity into the cell-wise velocity field. To do so, the grid cells which contain these sampled points are determined. And then velocities are added to these corresponding grid cells. After velocities are injected to the fluid, simulation time is advanced by a pre-determined time step. Then, the fluid simulator (Section III-B) updates the velocity field for the new time step.

B. Fluid Simulator

To compute the fluid velocity at an arbitrary position, we keep a continuous velocity field that is separate from the cell-wise velocity field. We denote the continuous velocity field with the vector function $\vec{v}_f(\vec{x}, t)$. The t is again the simulation time and $\vec{x}$ is an arbitrary position vector within the field. To compute $\vec{v}_f(\vec{x}, t)$ at some $\vec{x}$ we first determine the cell which contains $\vec{x}$. Then, the index vectors ($\vec{i}$) of the eight local neighbors of this cell are computed separately. For each of the eight neighbors, we retrieve the velocities from the cell-wise velocity field. Finally, we use trilinear interpolation to compute fluid velocity at $\vec{x}$. Having computed the cell-wise velocity field, the control is passed to the Flock Simulator to simulate flock behavior.

C. Flock Simulator

Flock simulator uses flocking rules proposed by Reynolds [1]. In addition to basic alignment, cohesion and seperation forces, we also have boundary forces that make the fish stay in the simulation environment. In order to achieve real time performance, we keep a secondary grid structure (different from the cell-wise velocity field) to cache the positions of the boids in the simulation space. This new grid structure reduces neighborhood search calculations considerably. Additionally, threading is used for the movement of boids. Each boid is assigned to a particular thread for processing.

At the beginning, the flock simulator computes the accelerations of the boids with respect to the Reynold's rules. We've used a simplified model to compute boid acceleration which is given in Equation 1. For simplicity we assumed that the boids' mass can be omitted. Hence, we dropped the mass out of the equation for motion. In Equation 1, vectors $\vec{f_a}(m)$, $\vec{f_c}(m)$ and $\vec{f_s}(m)$ correspond to the alignment, cohesion and separation forces, respectively. These forces are computed separately for each boid (with index m) using the common formulation proposed by Reynolds [1]. The coefficients ω_a, ω_c and ω_s control the magnitudes of the three components, respectively. We tune these coefficients empirically to obtain a natural looking simulation.

$$\vec{a}(m,t) = \omega_a \vec{f_a}(m) + \omega_c \vec{f_c}(m) + \omega_s \vec{f_s}(m) + \vec{f_o}(m) \tag{1}$$

The only additional force we've introduced is the obstacle force ($\vec{f_o}(m)$) which keeps the boids away from the aquarium boundaries and the terrain. The obstacle response force is computed as in Equation 2.

$$\vec{f_o}(m) = \frac{\mathit{sgn}(\delta_x(m))\vec{e}_x}{|\delta_x(m)| + h} + \frac{\mathit{sgn}(\delta_y(m))\vec{e}_y}{|\delta_y(m)| + h} + \frac{\mathit{sgn}(\delta_z(m))\vec{e}_z}{|\delta_z(m)| + h} \tag{2}$$

In Equation 2, $\vec{e}_x, \vec{e}_y, \vec{e}_z$ represent the basis vectors pointing at right(x), up(y) and forward(z) directions, respectively. *sgn* represents the sign function. A response force is activated only when a boid moves close to any of the six boundaries along the three directions. $\delta_x(m), \delta_y(m)$ and $\delta_z(m)$ are scalars which store the difference between the boid's position and the closest boundary for the respective direction. For any of the six boundaries along the three directions, if the boid's distance to the boundary is greater than a threshold, then the respective direction component is dropped from the equation. Finally h parameter is used to set an upper bound for the obstacle avoidance force. The magnitude of the obstacle avoidance force along any direction is never bigger than $1/h$.

For simulation environments where we use a terrain at the bottom of the aquarium, we slightly modify the obstacle response force along the up direction. We represent the terrain as a height map. In this case the upward difference $\delta_y(m)$ represents the difference between the position of the boid and the height map(at boid's position) along up direction.

We then compute an intermediate boid velocity at the current simulation time by summing up the boid's current velocity and acceleration (Equation 3). If the boid's speed exceeds the maximum permissible speed, we clamp the velocity such that its magnitude becomes the maximum speed.

$$\vec{v'}(m,t) = \vec{v}(m,t) + \vec{a}(m,t) \tag{3}$$

Each boid is affected by the fluid around it. For simplicity, we directly displace the boid with some portion of the fluid velocity vector at its position. The fluid velocity at the boid's position is determined from the continuous velocity field ($\vec{v}_f(\vec{x},t)$). In our test runs, we observed that, just displacing the boids with respect to the fluid velocity around it creates rather synthetic movement. To overcome this problem, we add some other portion of the fluid velocity to the boid itself. Therefore, boid movement under fluid force can be formulated as in Equation 4.

$$\begin{gathered}\vec{p}(m,t+1) = \vec{p}(m,t) + \vec{v'}(m,t) + (1-\alpha)\vec{v}_f(\vec{p}(m,t),t) \\ \vec{v}(m,t+1) = \vec{v'}(m,t) + \alpha\vec{v}_f(\vec{p}(m,t),t)\end{gathered} \tag{4}$$

In Equation 4, α is an experimental value. Increasing α creates the illusion of stronger fluid behaviour where boids are dragged along the velocity field of the fluid. Displacement doesn't affect the direction of the boids, thus, dragging effect is more pronounced as it should be. We use spherical linear interpolation to slowly update the directions of the boids such that they are aligned with their velocity vectors.

At any instance the user can interact with the system by relocating the velocity generator. When such an interrupt occurs the system schedules a user interface update for the next time step. At the beginning of the next time step, the control is delivered to the user interface component and the new velocities are added to the cell-wise velocity field as described in Section III-A

IV. Results

Using the methods described above, we developed a multi-threaded sample application using the Unity 3D Framework [9]. Users can interact with the system via either mouse or Kinect Controller [2]. Both controllers are used to localize the velocity generator.

Several snapshots from the application are given in Figure 2. Each of the snapshots are taken at run-time and they depict different flocking behaviour.

The performance of the system is bound to two parameters. The number of boids in the scene and the number of grid cells of the fluid solver. We tested our application on a workstation with the following hardware configuration: 8 cores, 2.8 GHz, 16 GB of RAM and Quadro K3000 GPU. The graphs given in Figure 3 demonstrate the system performance with respect to the fluid cell resolution and the number of boids in the system.

V. Conclusion

In this study we demonstrate a framework where boids and the fluid interact with each other. Fluid velocity can be manipulated by the user with the help of a Kinect controller [2]. In this work, we've primarily focussed on visually appealing flock simulation. The physical correctness of the simulation was a secondary concern. Our application runs at interactive frame rates. The approach we've taken is simple to implement and it is suitable to be used in modern video games or other computer graphics applications to provide immersive user experience.

VI. Acknowledgements

This work is supported by The Scientific and Technological Research Council of Turkey (TÜBITAK) under Grant No. 112E110.

References

[1] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[2] Microsoft, Inc., "Xbox 360 + Kinect, XBOX," http://www.xbox.com/en-US/kinect, 2014.

[3] C. Delgado-Mata, J. I. Martinez, S. Bee, R. Ruiz-Rodarte, and R. Aylett, "On the use of virtual animals with artificial fear in virtual environments," *New Generation Computing*, vol. 25, no. 2, pp. 145–169, 2007.

[4] C. K. Hemelrijk and H. Hildenbrandt, "Some causes of the variable shape of flocks of birds," *PloS One*, vol. 6, no. 8, article no. e22479, 13 pages, 2011.

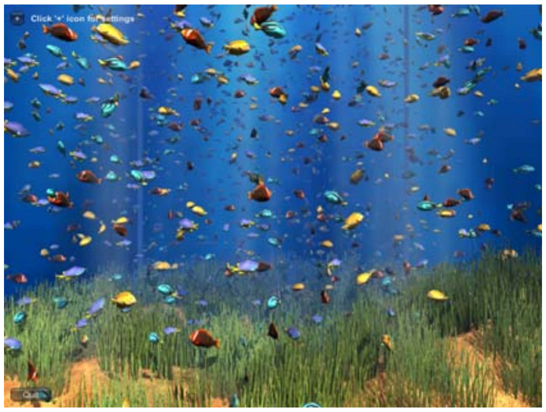
[5] J. Stam, "Stable fluids," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '99. ACM Press/Addison-Wesley Publishing Co., 1999, pp. 121–128.

[6] R. Fedkiw, J. Stam, and H. W. Jensen, "Visual simulation of smoke," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 15–22.

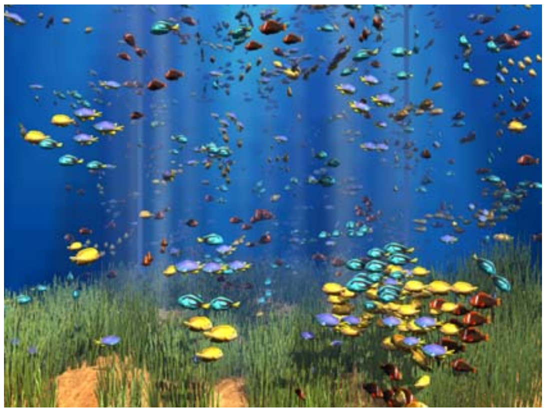
[7] J. Stam, "Real-time fluid dynamics for games," in *Proceedings of the Game Developer Conference*, vol. 18, 2003, p. 25.

[8] R. Bridson, *Fluid Simulation for Computer Graphics*. A K Peters Ltd., 2008.

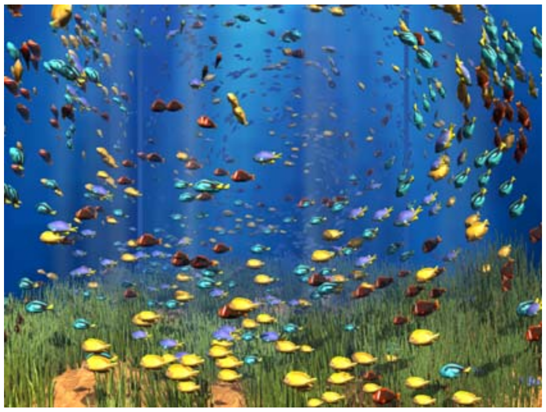
[9] Unity Technologies, "Unity 3d," http://unity3d.com, 2014.



(a)

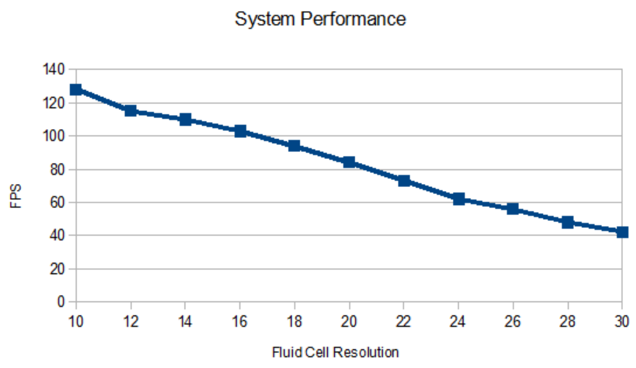


(b)

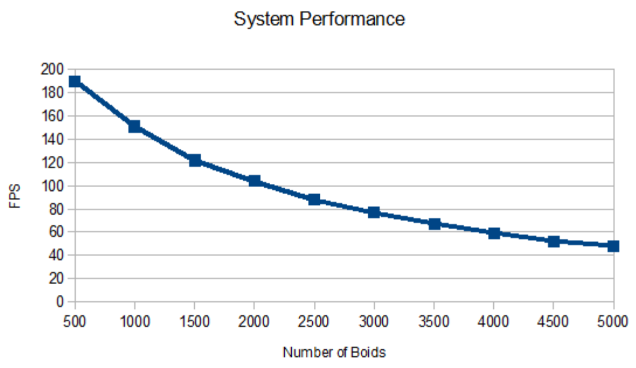


(c)

Fig. 2. Snapshots taken at run-time from the application. (a) Snapshot demonstrating the effect of weakening cohesion (ω_c) and alignment (ω_a) coefficients. The flocking behaviour is hardly visible. (b) Snapshot demonstrating flocking behaviour. Separation coefficient (ω_s) is set to a small value. Alignment and cohesion coefficients are set to greater values. (c) Scenario demonstrating the fluid's influence on the flock. The user had created a vortex by doing circular motion around the center of the aquarium. The vortex motion also directs the boids.



(a)



(b)

Fig. 3. Performance of the application with respect to different parameters. (a) System performance in terms of frames per second (FPS) with respect to the resolution of the fluid velocity field. (b) System performance with respect to the number of boids in the system