A Kite Simulation System using Position-based Method

Liu Yanhao

Graduate School of Library, Information and Media Studies

University of Tsukuba

March 2017

Contents

List of Figures

List of Tables

Chapter1

Introduction

Kites have a long history of over 2500 years. They were first invented for military purpose, like measuring distances, testing the wind, lifting men, signaling, and communication, but nowadays, kite flying has become a popular form of entertainment around the world. People fly kites at ceremonies and festivals, or enjoy as sports and games. There are various types of kites, from the simple flat (not bowed) and rectangular one to the more complicated ones, such as kite train(see Figure 1.1(a)) and 3D kites. [1]

Kite-flying are interesting and widely enjoyed by people, but in the field of computer graphics, there are few studies focus on physical-based simulations of kite. This is because wind flow affects kites in many ways; the angle of attack can changes sensitively than the wing of an airplane, the shape of a kite can be deformed easily because of its low rigidity(kite's solid frame normally made of bamboo or plastic). Furthermore, the interactions between the kite surface and the airflow are difficult to calculate even in the computational fluid dynamics field, due to its thinness feature. A kite simulation can be used in games and films, moreover, for kite makers, it will enhance the design of kite with a kite visualization system, which can visualize the kite fly scene in real time.

Generally, a kite consists of wings, string, and rigid anchors(see Figure 1.2). In this paper, we will ignore the effect of anchors for simplicity. To simulate the behavior of the string which connects the user and the kite, we use the mass-spring system at first, but it has an overshoot problem which would make the string unstable, so we use the Position-based method [Muller2007] instead. Because the transformed positions are computed directly, this method is stable and controllable for the kite string simulation.

For the purpose of making a kite simulation system, we also need to realize the effects between the kite wing and the airflow to model the kite flying behavior. In order to overcome this problem, one way is to use an elaborate grid structure to model the influence of dynamic wind flow with the thin kite, and use a discrete computational method, such as FEM, to compute the low pressure above and high

[1] photolibrary : https://www.photolibrary.jp/

(a) (b)

Figure 1.1: Different types of kites: (a) Traditional Japanese kites (b) Kite train

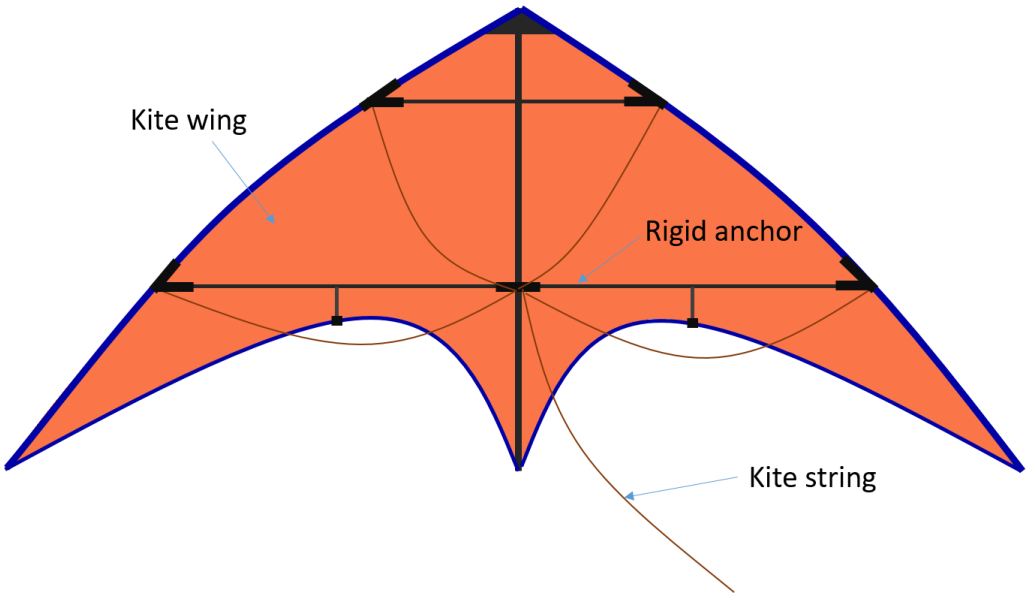


Figure 1.2: Consist of a kite: wing, string, rigid anchors

pressure below the kite, but performing in real-time is still a challenge. Alternative way is to use a particle-based method, in which the underlying equations of fluid motion(Navier-Stokes equations, see Chapter 5) are discretized on several moving particles and their interactions. Since the particle-based method fits better into current interaction with other objects like our kite, and is intuitive to set the simulation and control without any difficult. We use this method for the wind simulation, therefore, it is easy to model the airflow around the kite's surface in real-time even the kite has thinness feature, and can also account for the interactive force between the wind and the kite.

Similar to an aircraft, the kite has a greater density than air and flies according to aerodynamic forces. They generate the lift force necessary to overcome their weight. However, it is difficult to calculate the lift and drag force even we know their formula, because the lift and drag coefficients are depend on not only its shape and wind speed, but also attitude of the kite and the flow around the kite. In our system, we calculate the effect of the wind directly by a set of sensor particles which is based on the kite shape. By discretizing the kite with the sensor particles, our framework can easily handle the kite type effects, such as kite train and the 3D kite. Our method can be implemented on games, movies and industrial purpose without any difficult.

This paper makes two main contributions as follow:

- We simulate the string behavior in a stable and fast way by using the Chain Shape Matching method, which is a type of Position-based method developed for a hair simulation.

- We propose an interaction model between the airflow and the thinness objects by using SPH, which is a type of the particle-based method, and one-layer sensor particles. By using sensor particles, we can present a kite simulation system for complex kites.

Chapter2

Related Work

Kite simulation Like other flying objects including airplanes and birds, kite flying is because of the pressure on the upper and lower surface of the wing, which is produced by the air flows around the kite's surface [Wang2007]. In the field of computer graphics, the aerodynamic behavior of wings has been well studied to model the bird flight [Ramakrishnananda1999; Wu2003]. Umetani et al. [2014] provided a system that captures the parameters of a wing model to simulate paper airplanes. Although there are many approaches on wings of airplanes or birds, physical-based kite simulation are rarely seen.

The early research on kite flying simulation using the experimental data examined through elaborate wind tunnel experiments [Okamoto2009] for wings and kites. The system can precisely calculate the lift and drag force, according to the diagrams of the lift and drag coefficient acting on several aspect ratios, and simulate the kite string on a mass-spring model. Our approach is mainly based on this framework. However, the linear spring integrated with the explicit scheme would be instable [Muller2005] when the time step is large. Furthermore, the aerodynamics data are only available for flat shape kites, it is difficult to extend to more complicated shapes such as 3D types.

Recently, data-driven pipeline has commonly been used in computer graphics. Martin et al. [2016] use this method to make an omni-directional aerodynamic model(called OmniAD) for the motion of thin, rigid objects in air. In addition, they propose an intuitive and interactive interface for the design of three-dimensional kites, without the influence of the tensile force from the string in consideration. Their approach captures the aerodynamic properties of an falling object by using a single camera, so it is necessary to keep the object staying in the air for several seconds to get the falling trajectory; otherwise, it is also difficult to acquire the aerodynamic properties faithfully, especially the mutable kite train. In contrast to their approach, we focus on the efficient processing of kite-wind interaction by simulating the behavior of the wind particles and the kite sensor particles[Pirk2014] from macroscopic view.

Fluid-solid interaction In computer graphics, interaction between fluids and solid objects has been attempted with several methods. Stam [1999] proposed a semi-Lagrangian method and simplified the computation by using Fourier synthesis. Losasso et al. [2004] used an adaptive grid with a fine size around fluid and solid surfaces for fast computation, but the performance is not good enough for real time application. The improvement studies try to use tetrahedral grid [Feldman2005; Klingner2006] around solid objects, however the generation of tetrahedral meshes requires more computing cost. In recent, Langrangian fluid simulation is becoming a popular topic in computer animation. Its particle-based feature allows simulations of phenomena on small-scale, turning it into a competitive method to model dynamic effects. A particle-based method that is commonly used to model fluid behavior in computer graphics field is Smoothed Particle Hydrodynamics (SPH) [Muller2003], which we use to simulate the wind motion. To handle the fluid-solid boundaries, boundary represented by triangles are commonly used in distance detection, known as distance based penalty method , however these approaches restrict the time step [Muller2004]. Moreover, particles stick to the boundary easily because of the lack of fluid neighbors. In order to solve this problem, Akinci et al. [2009] use a virtual volume of the boundary particles to alleviate density discontinuities at boundaries and particle stickiness, which is used in this paper.

String simulation The early research on deformable objects, such as a string, cloth and rubber object, are based on mass-spring systems [Baraff1998], the Boundary Element Method (BEM) [James1999], the Finite Element Method (FEM) [Debunne2001, Muller2002, Muller2004], the Finite Volume Method (FVM) [Teran2003]. However, because of the unstable or high cost problems, in recent years, position-based approach has been popularly used. With the different structure of particle groups which is called cluster, Fedor et al. [2005] use this approach to simulate characters in games. Muller et al. [2005] simulate deformable objects by moving particles towards its certain goal positions. Nealen et al. [2006] use the position-based dynamics method for the cloth simulation. Rungjiratananon et al. [2010] expand this method to simulate the hair. In our approach, we make use of the position based dynamics to stably and efficiently simulate the kite string.

Chapter3

Overview

The goal of our system is to provide a stable real-time simulator, including the physics-based simulation of the string, wing and tail with complicated shapes. Our approach is based on the Flying Japanese Kite framework [Okamoto2009], the foremost challenge is to simulate the string and airflow around kite stable and fast. Our simulator composed of three components: string simulator, wind simulator and kite integrator including kite-wind interaction as shown in Figure 3.

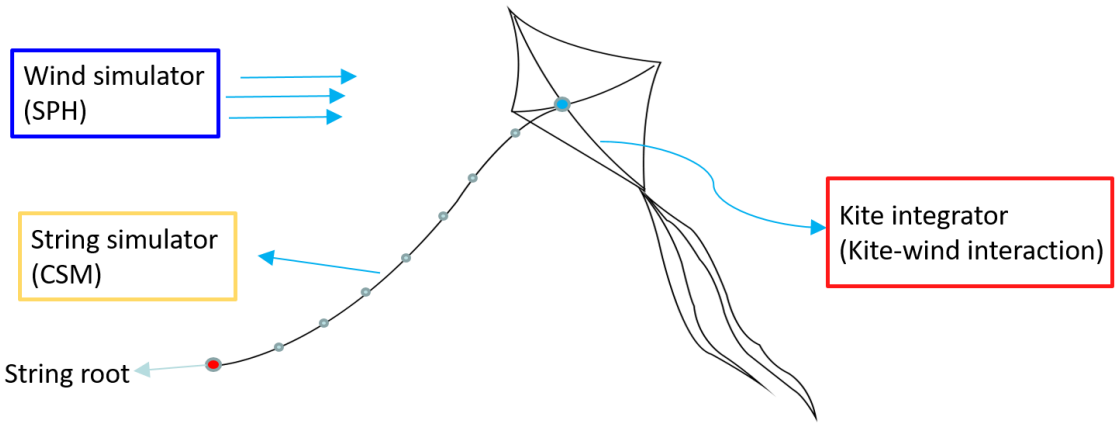


Figure 3.1: Three components of a kite: string simulator, wind simulator and kite integrator.

string simulator We use the Chain Shape Matching method [Rungjiratananon2010], an expansion of the basic Shape Matching method [Muller2005] for the simulation

of string. This method mainly accounts for the string like objects such as individual hair strands, and is simpler, faster and numerically more stable than the traditional shape matching method and the mass-spring method. The Chain Shape Matching method directly calculate the position of the string without the force, while we need the force acting on the kite, therefore, we use the final two particles of the string to calculate the tensile force $\boldsymbol{f} = -k(\boldsymbol{x}(t) - \boldsymbol{x}(0))$, just the same as the mass-spring method. We describe details of the string simulation in Chapter 4.

wind simulator We simulate the wind flow with a Lagrangian fluid model that uses moving particles to discrete the fluid motion. This method is adaptive, and allows for real-time interaction with complicated kite shapes. Chapter 5 gives a detailed description of SPH and the wind collision and response.

kite integrator The kite-wind interaction is modeled by adding particle integrators to the geometry of kite, which can consider an effect of wind particles in real time. By doing this, we can measure the effect of the aerodynamic forces easily. In this paper, we set a one-layer sensor particles(see Figure 5.2 (b)) to represent the thin structure of the kite, but this will cause an incorrect density computation because densities of fluid particles near the rigid bodies are underestimated. To modify this underestimation, we use the same method proposed in [Akinci2012]. In following sections, we use suffix s like m_s for sensor particles and suffix w like m_w for wind particles. We describe this part thoroughly in Chapter 6.

Chapter4

String Simulation

A string is an important factor of kite. It plays an important role of direct interface controlled by the player, and can effect the kite though the tensile force. In this chapter, at first, we explain a simulation of the string as an elastic body which can be described by the Shape Matching(SM) with clustering(section 4.1). Then we explain how to model a string using the Chain Shape Matching(CSM) with the strain limiting into concern(section 4.2). Finally, we describe the calculation of the tensile force between the string and the kite(section 4.3).

4.1 Shape Matching

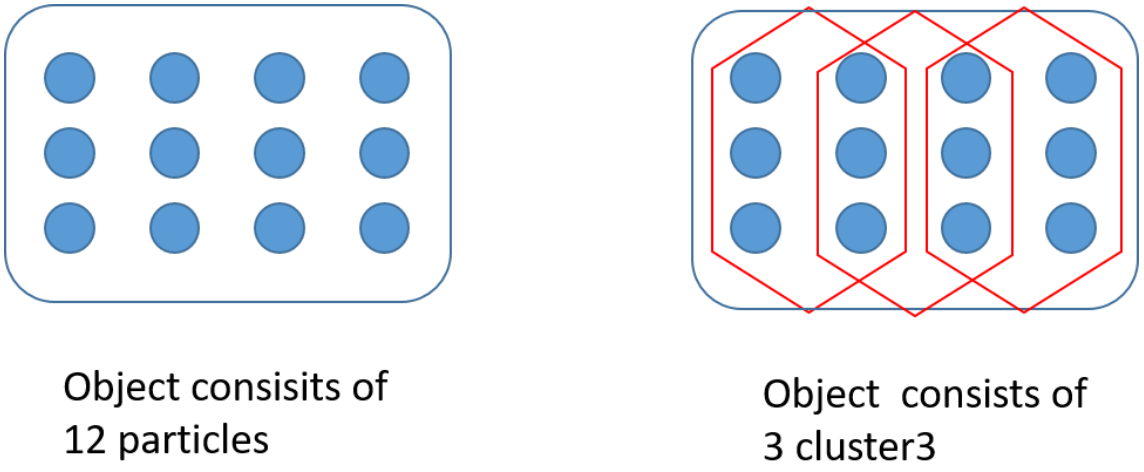


Figure 4.1: Object which consists of overlapping clusters.

Shape Matching is a kind of Position-based method. With the advantages of its unconditional stability and high controllability, it is commonly used by deformable objects' simulation. In Position-based method, transformed positions are computed

first, while other method such as mass-spring method calculate the force acting on, then, particles are updated towards its goal positions. While all the particles are moved towards the suitable positions before the next time step, the overshooting problem of explicit integration schemes can be eliminated. However, the basic SM method described can only simulate small deviations from the original shape of rigid objects. Though Muller et al. [2005] defined a linear transformation to represent shear and stretch, and a quadratic transformation to represent quadratic transformation, it is difficult to treat more complex motions. To extend the range of deformation, particles divided into overlapping groups are popular used, we call it cluster(see Figure 4.1). Then, at each time step, the original shape of each cluster is transformed to match with its current shape.

In SM, each particle i has a mass m_i and an initial position $\boldsymbol{x}_i^0$, and belongs to a cluster C_i, we make the cluster for all particles with near particles search, so that C_i means the cluster with center particle. Now, we assume that the object is moved and deformed by external forces, and its behavior is represented by three components: translation, rotation and skew. Therefore, the updated transformed position of a particle $\boldsymbol{x}_i^{'}$ (see Figure 4.2) can be expressed as:

$$\boldsymbol{x_i'} = \boldsymbol{RS}\left(\boldsymbol{x}_i^0 - \boldsymbol{x}_{cm,r}^0\right) + \boldsymbol{x}_{cm,r}^{'} \tag{4.1}$$

where $\boldsymbol{R}$ is the rotation matrix, $\boldsymbol{S}$ is the symmetric matrix representing the skew, $\boldsymbol{x}_{cm,r}^0$ is the center of the original shape of the cluster, $\boldsymbol{x}_{cm,r}^{'}$ is its translation.

We know that the elastic material, such as rubber and string, can return to its original state after deformation, so the changing terms are translation and rotation, there is no need to account for the skew part $\boldsymbol{S}$. Therefore, we can obtain the goal position $\boldsymbol{g}_i$ from:

$$\boldsymbol{g_i} = \boldsymbol{R}\left(\boldsymbol{x}_i^0 - \boldsymbol{x}_{cm,r}^0\right) + \boldsymbol{x}_{cm,r}^{'} \tag{4.2}$$

The rotation part $\boldsymbol{R}$ can be achieved by minimizing the quantity E as follow:

$$E = \sum_{i \in C_i} m_i \left|\boldsymbol{g_i} - \boldsymbol{x}_i^{'}\right|^2 = \sum_{i \in C_i} m_i \left|\left(\boldsymbol{x}_i^0 - \boldsymbol{x}_{cm,r}^0\right) + \boldsymbol{x}_{cm,r}^{'} - \boldsymbol{x}_i^{'}\right|^2 \tag{4.3}$$

In relative coordinate according to the center of mass, let us define the initial and transformed position of a point i as $\boldsymbol{q}_i = \boldsymbol{x}_i^0 - \boldsymbol{x}_{cm}^0$ and $\boldsymbol{p}_i = \boldsymbol{x}_i - \boldsymbol{x}_{cm}$. If we use an linear transformation $\boldsymbol{A} = \boldsymbol{RS}$ to represent the rotation and the deformation, then the term to be minimized turns to:

$$E = \sum_{i \in C_i} m_i (\boldsymbol{A}\boldsymbol{q}_i - \boldsymbol{p}_i)^2 \tag{4.4}$$

set the derivatives with respect to the coefficient a_{ij} of the transformation $\boldsymbol{A}$ to zero:

$$\frac{\partial E}{\partial a_{ij}} = \sum_{i \in C_i} 2m_i (\boldsymbol{A}\boldsymbol{q}_i - \boldsymbol{p}_i)\boldsymbol{q}_i^T = 0 \tag{4.5}$$

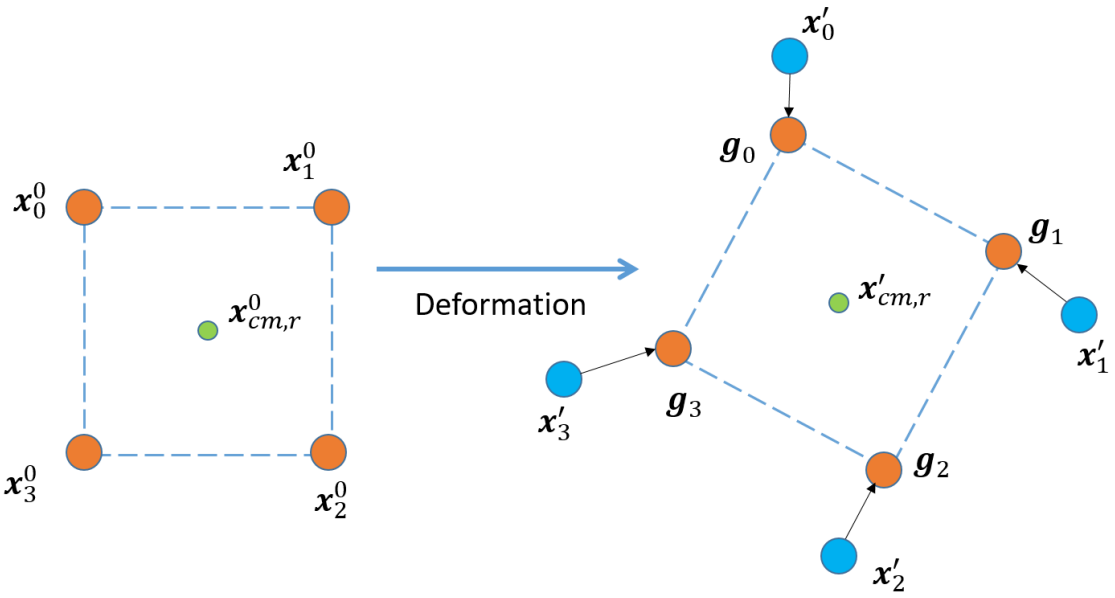


Figure 4.2: $\boldsymbol{x}_i^0$ is the original position, $\boldsymbol{x}_i$ is the updated position though a deformation containing translation and rotation and $\boldsymbol{g}_i$ is the goal position of particle, green point $\boldsymbol{x}_{cm,r}^0$ and $\boldsymbol{x}_{cm,r}^{'}$ are the center of the shape.

Now, $\boldsymbol{A}$ can be computed as follow:

$$\boldsymbol{A} = (\sum_{i \in C_i} m_i \boldsymbol{p}_i \boldsymbol{q}_i^T)(\sum_{i \in C_i} m_i \boldsymbol{q}_i \boldsymbol{q}_i^T)^{-1} = \boldsymbol{A}_{pq} \boldsymbol{A}_{qq}. \tag{4.6}$$

The term $\boldsymbol{A}_{qq} = \sum_{i \in C_i} m_i \boldsymbol{q}_i \boldsymbol{q}_i^T$ is a symmetric matrix which contains only scaling. Therefore, the matrix $\boldsymbol{A}_{pq}$ contains the deformation $\boldsymbol{S}$ and the rotation part $\boldsymbol{R}$. we can use a polar decomposition $\boldsymbol{A}_{pq} = \boldsymbol{R}\boldsymbol{S}$ to get the $\boldsymbol{R}$ and $\boldsymbol{S}$ as:

$$\boldsymbol{S} = (\boldsymbol{A}_{pq}^T \boldsymbol{A}_{pq})^{\frac{1}{2}} \tag{4.7}$$

$$\boldsymbol{R} = \boldsymbol{A}_{pq} \boldsymbol{S}^{-1} \tag{4.8}$$

we just use the rotation part $\boldsymbol{R}$ to represent the behavior of elastics material.

Finally, the velocity $\boldsymbol{v}$ and the position $\boldsymbol{x}$ of next step can be written as:

$$\boldsymbol{v}_i\,(t + \Delta t) = \boldsymbol{v}_i(t) + \frac{\boldsymbol{g}_i(t) - \boldsymbol{x}_i(t)}{\Delta t} + \frac{\boldsymbol{f}_i}{m_i} \Delta t \tag{4.9}$$

$$\boldsymbol{x}_i\,(t + \Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i(t + \Delta t) \tag{4.10}$$

where Δt is the time step length, $\boldsymbol{f}_i$ is the external force. With the elaborate cluster(like the clusters in Figure 4.1), Shape Matching method is suitable for 3D deformable objects especially complicated shapes, but to our simple kite string, it seems too expensive. So we choose the Chain like cluster instead.

4.2 Chain Shape Matching

Chain Shape Matching is an extensive approach of the shape matching method which is mainly used for hair simulations. It can achieve a sophisticated animation with complex hairstyles stably.

Obviously, it is also suitable for our kite string simulation. The mainly contribution of this method is its chain like cluster(see Figure 4.3), which is different from the shape matching method mentioned before(see Figure 4.1), and each chain region uses the same algorithm to compute. It is fast and simple to simulate the chain lite objects, such as kite string, because it just employ one-dimensional shapes. In our method, we set the chain region half-width $w = 3$(see Figure 4.3) which consider the stiffness of the kite string. With the use of parameters α, $\beta \in [0, 1]$, similar to the original shape matching paper [Muller2005], where α is the stiffness part of the simulation and β allows goal positions to undergo a linear transformation. While

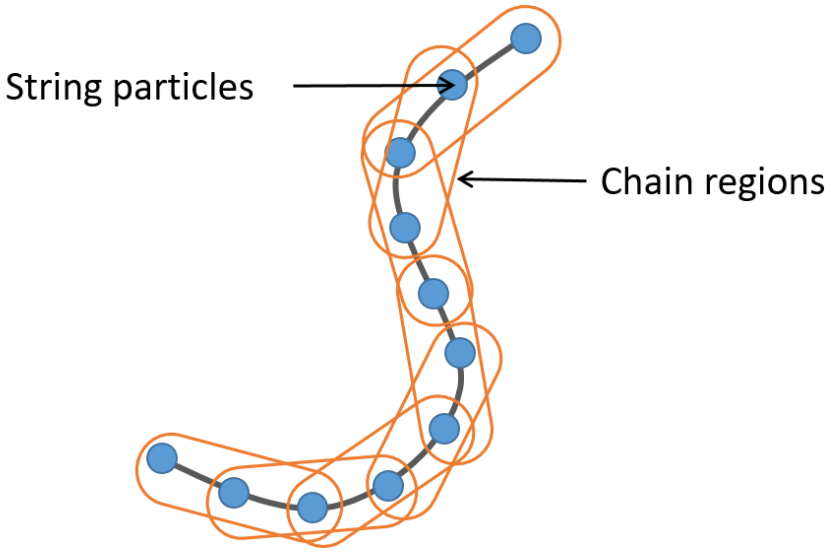


Figure 4.3: Chain-like cluster in CSM.

α and β can control the stiffness of the half-width w independently, taking into consider the parameter α and β can make the our chain region softer.

In physics-based cloth simulations, strain limiting is used to constrain stretching and handle collision for deformations. Most of these works use an elastic system to simulate the cloth, but without shorten the length of each segment (such a spring in the mass-spring cloth simulation systems), the cloth becomes very elastic and looks unrealistic. In our approach, to constrain the stretching of the kite string, we modify the position of stretched segments in the string (Figure 4.4 (a)) after executing the shape matching calculation, then we adjust the near-end particle of a stretched segment along the propagation direction of the previous segment , to the non-stretched position as shown in Figure 4.4 (b). Finally, the kite string becomes a smooth curve from its root to end.

4.3 String Force

In this section, we describe how to calculate the tensile force between the string and kite. While each particles in CSM contains the velocity and position, there is no direct way to compute the force. To solve the problem, we consider a mass-spring model to compute the internal force of the final two particles (see Figure 4.5). We just place one spring between two particles, so that the stability of the string simulator is preserved, while the general mass-spring system has many springs which is the reason of instability.

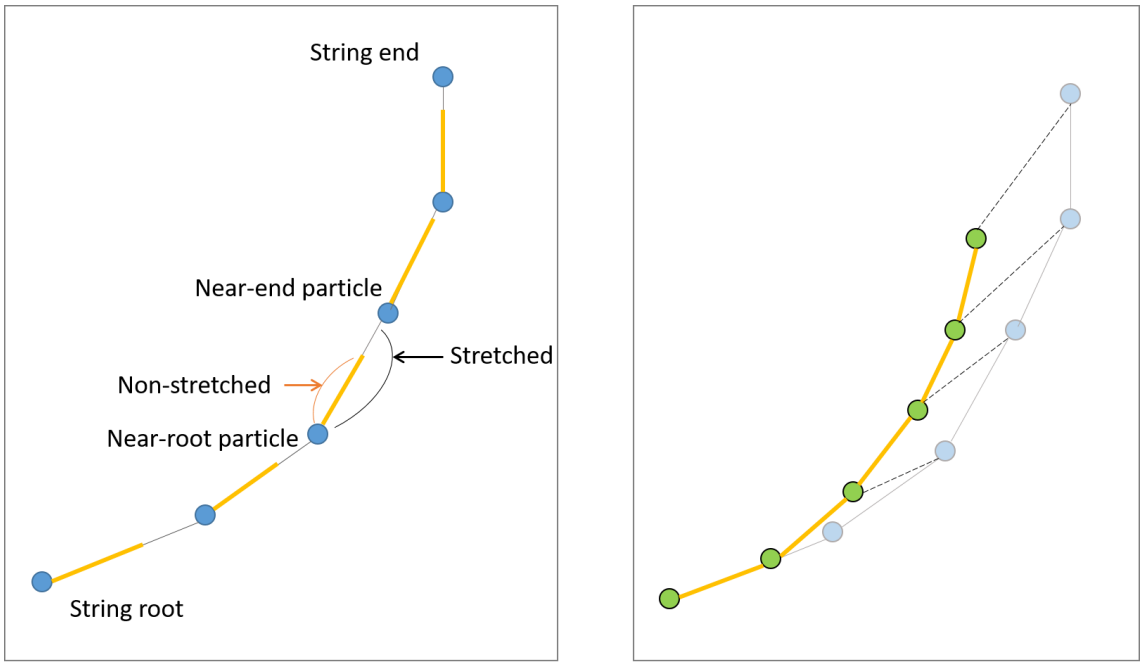


Figure 4.4: Strain limiting of a string. (a) a stretch of string after shape matching. (b) Adjust near-end particle along the propagation direction.

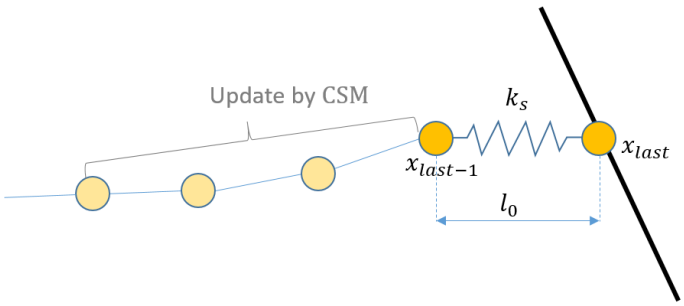


Figure 4.5: We update the string by the CSM method and calculate the force by a mass-spring model between the last two particles.

In our paper, the segment string between the final two particles has a rest length l_0 and spring constant k_{spring}. Then, the tensile force of the kite string can be computed as follow:

$$\boldsymbol{T}_{string} = -k_{spring}(|\boldsymbol{x}_{last} - \boldsymbol{x}_{last-1}| - l_0)\frac{\boldsymbol{x}_{last} - \boldsymbol{x}_{last-1}}{|\boldsymbol{x}_{last} - \boldsymbol{x}_{last-1}|} \tag{4.11}$$

Where $\boldsymbol{x}_{last}$, $\boldsymbol{x}_{last-1}$ denotes the final two particles' position of the string.

With the tensile force achieved, we can computed the total force of the kite in Chapter 6.

Chapter5

Wind Simulation

The behavior of the fluid such as water and wind can be described by the Navier-Strokes equations:

$$\rho\frac{D\boldsymbol{v}}{Dt} = \nabla P + \mu\nabla^2\boldsymbol{v} + \rho g \tag{5.1}$$

$$\nabla \cdot \boldsymbol{v} = 0 \tag{5.2}$$

where the first equation is called the momentum equation which describes change in momentum of infinitely small fluid element. ρ is the density, $\boldsymbol{v}$ and P denote the velocity and pressure of the fluid. The term $\frac{D\boldsymbol{v}}{Dt}$ is known as the material derivative, and can be computed as $\frac{D\boldsymbol{v}}{Dt} = \frac{\partial\boldsymbol{v}}{\partial t} + \boldsymbol{v}\cdot\nabla\boldsymbol{v}$. It describes the rate of movement at a fixed point in space. The three terms on the right side reflect the influence of the forces due to pressure and viscosity as well as external forces such as gravity. The second equation is known as the incompressibility condition. This constraint preserves the mass of the fluid and implies constant density throughout the fluid body.

A variety of solutions for the Navier-Stroke equations has been described in [Bridson2008]. In our paper, we use Smoothed Particle Hydrodynamics(SPH), which is considered a competitive tool to model hydrodynamic effects recently, we give a detailed description at section 5.1. Different from other fluid simulation methods which require spatial subdivision, particles in SPH are trackable objects, and are easy to simulate. We explain the collision and response between the wind particle and the kite at section 5.2. By using the sensor particles mentioned in Section 5.3, we can simply model the interaction between the kite and the wind.

5.1 Smoothed Particle Hydrodynamics

In SPH, each particle has a position $\boldsymbol{x}_i$ and can represent physical quantities such as density or pressure, which can be approximately computed as:

$$A(\boldsymbol{x}) = \sum_{j=1}^{N} \frac{m_j}{\rho} A_j W(\boldsymbol{x} - \boldsymbol{x_j}, h), \tag{5.3}$$

where $\boldsymbol{x}$ is the particle position, m_j and $\boldsymbol{x}_j$ are the mass and position of its neighbor particle j, ρ is the fluid density, N is the total number of its neighbor, A_j is the quantity stored in the particles, and W donates the smoothing kernel within a kernel radius h. As shown in Figure 5.1, for each particle i, the distance from one of its neighbor particles j is r, and can be expressed as $r = |\boldsymbol{x_i} - \boldsymbol{x_j}|$. The contributions of each neighbor particle which is governed by the kernel function W , are weighted according to the distance r within the radius $r \leq h$.

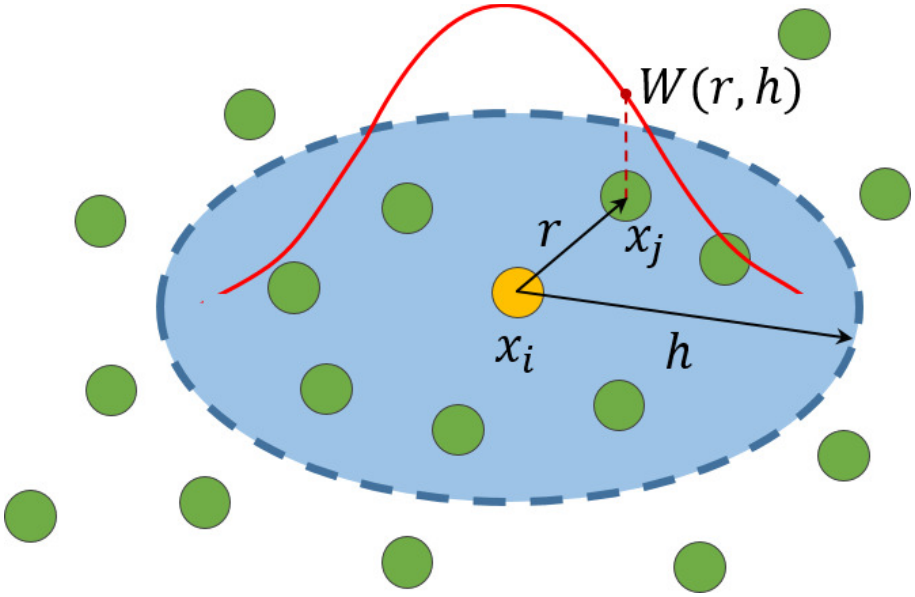


Figure 5.1: Construction of SPH: For each particle $\boldsymbol{x}_i$ in the simulation field, its corresponding physical quantity is calculated from a weighted sum of its neighbor particle $\boldsymbol{x}_j$ within the radius h.

In SPH, density is an essential field variable to compute pressure and viscosity forces, can be directly estimated by substituting A with ρ in Equation 5.3 in the form:

$$\rho_i = \sum_j m_j W(\boldsymbol{x_i} - \boldsymbol{x_j}, h) \tag{5.4}$$

and the gradient of pressure included in 5.1 could be computed by using:

$$\boldsymbol{F}_{pressure} = \nabla P = \sum_{j=1}^{N} m_j \frac{p_i^2 + p_j^2}{\rho_i \rho_j} \nabla W(\boldsymbol{x_i} - \boldsymbol{x_j}, h), \tag{5.5}$$

here, we calculate the pressure p_i and p_j in the form: $p = k_{press}(\rho - \rho_0)$ proposed by [Desbrun1996]. The viscosity describes the resistance of the fluid to deform, water has low viscosity while honey as high viscosity. In SPH, the viscosity part is expressed as:

$$\boldsymbol{F}_{visicosity} = \mu \nabla^2 \boldsymbol{v} = \sum_{j=1}^{N} m_j \frac{\boldsymbol{v}_j - \boldsymbol{v}_i}{\rho_j} \nabla^2 W(\boldsymbol{x_i} - \boldsymbol{x_j}, h), \tag{5.6}$$

where $\boldsymbol{v}$ denotes the velocity of a particle.

The kernel function W we used in this paper is Poly6 kernel [Muller2003] which can be computed as following:

$$W_{poly6}(\boldsymbol{r}, h) = \alpha_1 \begin{cases} (h^2 - r^2)^3, & 0 \le r \le h \\ 0, & otherwise \end{cases} \tag{5.7}$$

$$\alpha_1 = \frac{4}{\pi h^8}, \frac{315}{64\pi h^9} \quad for \quad 2D, 3D \tag{5.8}$$

and the gradient and the Laplacian are in the form:

$$\nabla W_{ploy6}(\boldsymbol{r}, h) = \alpha_2 \begin{cases} (h^2 - r^2)^2 \boldsymbol{r}, & 0 \le r \le h \\ 0, & otherwise \end{cases} \tag{5.9}$$

$$\alpha_2 = -\frac{24}{\pi h^8}, -\frac{945}{32\pi h^9} \quad for \quad 2D, 3D \tag{5.10}$$

$$\nabla^2 W_{ploy6}(\boldsymbol{r}, h) = \alpha_3 \begin{cases} 3(h^2 - r^2)^2 - 4r^2(h^2 - r^2), & 0 \le r \le h \\ 0, & otherwise \end{cases} \tag{5.11}$$

$$\alpha_3 = -\frac{24}{\pi h^8}, -\frac{945}{32\pi h^9} \quad for \quad 2D, 3D \tag{5.12}$$

(5.12)

5.2 Collision and Response

Since we use the particle-based dynamic as the wind model, it is easy to calculate collisions of with rigid bodies. Normally, We can use the set of multi-layer boundary particles which is the same as wind particles to represent the object(see Figure 5.2 (a)), we call these sensor particles, and compute the collisions between them. But the wing of a kite is quite thin, we can use only one layer set of sensor particles(see Figure 5.2 (b)) to sample the surface of kite. We know that density summation approach approximates the fluid density correctly only if the fluid has the same initial density. However, the wind particles near the boundary do not have enough wind particle neighbors(see Figure 5.2 (b)), which would make the density gradient remains discontinuous. To avoid this problem, we take the neighboring sensor particles into consider. We would give a solution in section 5.3. In our system, we check

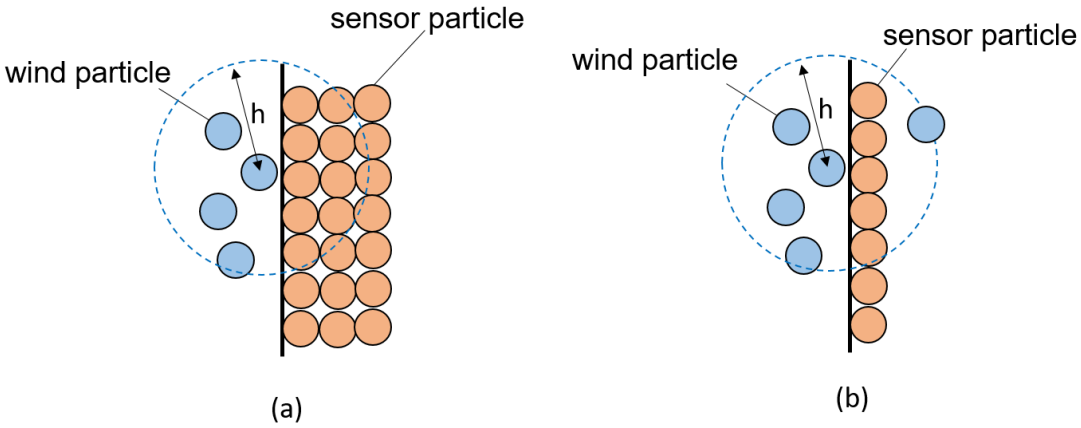


Figure 5.2: (a) Multi-layer boundary (b)One layer boundary

the collision by looking-up the distance to the sensor particles during the particle movement computation, which is similar to [Pirk2010]. By doing this, we can get two advantages: First, the sensor particles allows us to derive a model to handle different shapes, such as kite train an 3D kites; the particles successfully alleviates the aerodynamic problems near the kite surface, which means we can achieve to calculate the forces affected by wind directly.

5.3 Corrected density, pressure and friction

Since we focus on the interaction of wind flow with thinness kite, we should set a one-layer sensor particles. But the particles on one side of a sensor particle would

affect potential fluid particles on the other side because of its thinness, so we have to set the multi-layer boundary particles as shown in Figure 5.2 (a). To avoid this, We calculate the density by taking the virtual volume V_{s_i} of each sensor particle s_i into account, similar to [Akinci2012]:

$$V_{s_i} = \frac{m_{s_i}}{\rho_{s_i}} = \frac{m_{s_i}}{\sum_k m_{s_k} W_{ik}}, \tag{5.13}$$

where m_{s_i}, ρ_{s_i} are the mass and density of the particle s_i , W is the smoothing kernel, k denotes neighbors of the sensor particle. By using this volume, the corrected density of a wind particle w_i can be written as:

$$\rho_{w_i} = m_{w_i} \sum_j W_{ij} + \sum_k \rho_0 V_{s_i} W_{ik}, \tag{5.14}$$

where ρ_0 denotes the rest density of the wind flow that the kite is interacting with.

Due to the applied pressure from wind to the sensor does not have any kinematic influence on the nearby wind particles, we modify the pressure force mentioned in section 5.1 from sensor particle s_j to a wind particle w_i as:

$$\boldsymbol{F}^p_{w_i \leftarrow s_j} = -m_{w_i} \rho_0 V_{s_j} \left(\frac{P_{w_i}}{\rho^2_{w_i}} \right) \nabla W_{ij} \tag{5.15}$$

where P_{w_i} denotes pressure of the wind particle and ρ_{w_i} is the wind density calculated by Equation (5.14).

We know that, in every interaction, there is a pair of forces acting on the two interacting objects. So the symmetric pressure from a wind particle w_i to sensor particle $s_i$is:

$$\boldsymbol{F}^p_{s_j \leftarrow w_i} = -\boldsymbol{F}^p_{w_i \leftarrow s_j} \tag{5.16}$$

As described in [Akinci2012], viscosity force from sensor particle s_i to wind particle w_i can also be represented as:

$$\boldsymbol{F}^v_{w_i \leftarrow s_j} = -m_{w_i} V_{s_j} \Pi_{ij} \nabla W_{ij} \tag{5.17}$$

with the laminar artificial viscosity model:

$$\Pi_{ij} = -\nu \left(\frac{\max(\boldsymbol{v}_{\mathrm{ij}} \cdot \boldsymbol{x}_{\mathrm{ij}}, 0)}{|\boldsymbol{x}_{ij}|^2 + \epsilon h^2} \right), \tag{5.18}$$

with the viscous factor $\epsilon = \frac{2\sigma h c_s}{\rho_i + \rho_j}$, where σ is the viscosity coefficient between wind and kite, c_s denotes the speed of sound, and we set 90m/s, similar as [Monaghan2005], $\boldsymbol{v}_{ij} = \boldsymbol{v}_i - \boldsymbol{v}_j$, $\boldsymbol{x}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$, and the term ϵh^2 is used to avoid singularities for $|\boldsymbol{x}_{ij}| = 0$(we use $\epsilon = 0.01$). Then, the symmetric friction force from a wind particle w_i to a sensor particle s_i can be written as:

$$\boldsymbol{F}^v_{s_j \leftarrow w_i} = -\boldsymbol{F}^v_{w_i \leftarrow s_j} \tag{5.19}$$

Finally, we compute the total force acting on a wind particle $\boldsymbol{F}_{w_i}$ and the total force acting on a sensor particle $\boldsymbol{F}_{s_i}$ as:

$$\boldsymbol{F}^{total}_{w_j} = \sum_j \left(\boldsymbol{F}^p_{w_i \leftarrow s_j} + \boldsymbol{F}^v_{w_i \leftarrow s_j} \right), \tag{5.20}$$

$$\boldsymbol{F}^{total}_{s_j} = \sum_j \left(\boldsymbol{F}^p_{s_j \leftarrow w_i} + \boldsymbol{F}^v_{s_j \leftarrow w_i} \right). \tag{5.21}$$

Now, we know the forces from the wind particle, then, we can compute the total force of the kite, and the torque written as $\boldsymbol{T} = \boldsymbol{F}\boldsymbol{x}$, which will be described detail in the next chapter.

Chapter6

kite Simulation

In this chapter, we describe our kite models in section 6.1 and the computation of the total forces exerted to the kite in section 6.2.

6.1 Kite model

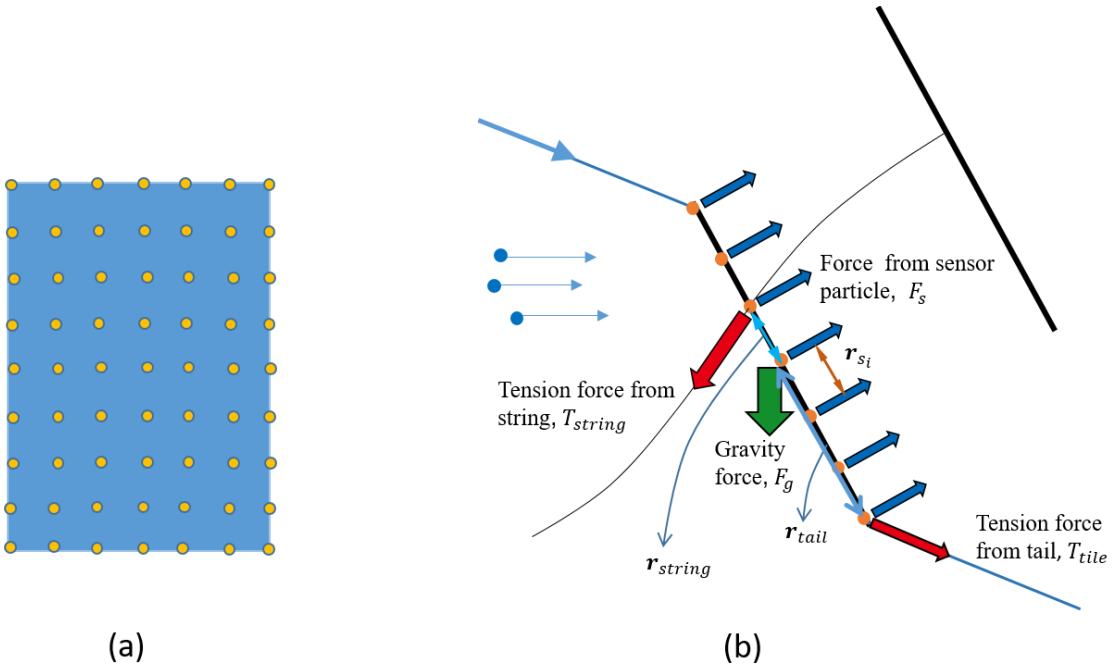


Figure 6.1: (a) Rectangular kite and its sensor particle set. (b) Forces exerted on the kite from side view.

The shapes of kite appear in various characteristics. Several kites have complicated shapes, such as Yakko kite and delta-wing kite(see Figure 1.2), the box kite

has a 3D-box shape and the kite train is constituted by a series of sub-kites, which is common in rectangular, diamond or other simple shapes. In our paper, we use a rectangle as the simplest kite model, and the sensor particles setting are shown in Figure 6.1 (a). Because sensor particles allows us to extend our method to handle different shapes, we then simulate a kite train based on several rectangle kites, each kite has the same sensor particle setting as the simplest one. We also use a simple 3D cube for the box kite model, and use a random position from polygon for the set of sensor particles.

6.2 Kite forces

As shown in Figure 6.1 (b), forces exerted on the kite consist of tension force from the kite string and kite tails, aerodynamic forces affected by wind particles, and the kite gravity force. Until now, we have computed the forces of sensor particles in chapter 5, and acquired the tension force of the kite string from chapter 4.

The tail is to balance the weight and keep the kite stable enough to against the wind. For the convenience in implement, we use a mass-spring model for the tail simulation, and compute the tensile force as follow:

$$\boldsymbol{F}_{T_{tile}} = k_{tail}\Delta l_t + k_d \boldsymbol{v} \tag{6.1}$$

where k_{tail} is the same as Equation (4.11), Δl_t is an extension of the tail, k_d is a factor of damping and $\boldsymbol{v}$ is a relative velocity at the end of the tail. Because the tail is connected to the kite, it get an influence from the kite and the kite get its reaction tension force. Here, we shorten the length of the tail to keep the simulation realistic.

Then, we can get the total force of the kite as:

$$\boldsymbol{F}_{kite} = \boldsymbol{F}_{T_{string}} + \boldsymbol{F}_{T_{tail}} + \boldsymbol{F}_g + \sum_i \boldsymbol{F}_{s_i}^{total} \tag{6.2}$$

The moment of the kite can be computed by:

$$\boldsymbol{M}_{kite} = \tilde{\boldsymbol{F}}_{T_{string}} \times \boldsymbol{r}_{string} + \sum_i \tilde{\boldsymbol{F}}_s \times \boldsymbol{r}_{s_i} + \tilde{\boldsymbol{F}}_{T_{tail}} \times \boldsymbol{r}_{tail} \tag{6.3}$$

where $\tilde{\boldsymbol{F}}$ is the normal direction component of $\boldsymbol{F}$, $\boldsymbol{r}_{string}$ is the position vector of the string tension force bearing point from the center of gravity, $\boldsymbol{r}_{s_i}$ is the position vector of each sensor particle s_i from the the center of gravity, $\boldsymbol{r}_{tile}$ is the position vector of the tail tension force bearing point from the center of gravity(see Figure 6.1 (b)). Then, we can calculate the rotation of the kite and update the kite in real time.

Chapter7

Implementation and Results

In this chapter, we will describe our implementation environment in section 7.1, and show the results in section 7.2, which contains the simplest rectangle kite, kite train and the box kite.

7.1 Implement Environment

We implemented our kite-flying simulation system with the using of C++ on a desktop computer, and the detailed specification is given in Table 7.1.

CPU	Intel Core i7 3.20 GHz
RAM	16 GB
GPU	NVIDIA GeForce GTX 780
VRAM	4GB
OS	Windows 7
Graphic API	OpenGL

Table 7.1: Specification of the running environment

In our system, the physical constants used in the string simulation is listed in Table 7.2, the settings of wind particles and sensor particles are shown in Table 7.3 and Table 7.4.

7.2 Results

By using the sensor particle setting described in Figure 6.1 (a), we simulate a simple rectangle kite. To make the kite looks like flying in the sky, we make a sky environment in addition. As shown in Figure 7.1, the kite consists of a rectangle body and two tails. We make a wind flow generator which can produce 200 wind particles each time step, and the position of the generator is according to the center of the

Type	Constant and Symbol	Value
String	Gravitational Acceleration g	9.81 m/s^2
	Stiffness Parameter α	0.8
	Deformation Parameter β	0.9
	Spring Constant of Kite String k_{spring}	250
	Spring Constant of Tail k_{tail}	100
	Damping Factor k_d	1
	Chain Region Half-width w	3
	Segment length between two points k_{spring}	0.1 m

Table 7.2: Physical constants used in our string simulation

Type	Constant and Symbol	Value
Wind	Rest Density of Fluid ρ_{fluid}	1000 kg/m^3
	Mass of Fluid Particle m_{fluid}	0.001 kg
	Max Fluid Particle Number N_{fluid}	30000
	Viscosity Value of Fluid μ_{fluid}	0.01 $kg/(s \cdot m)$
	Initial wind particles velocity v	8.0 m/s
	kernel radius h_w	0.04 m
	Gas Constant k_{press}	3.0 $J/(mol \cdot K)$

Table 7.3: Physical constants of wind particles used in our wind simulation

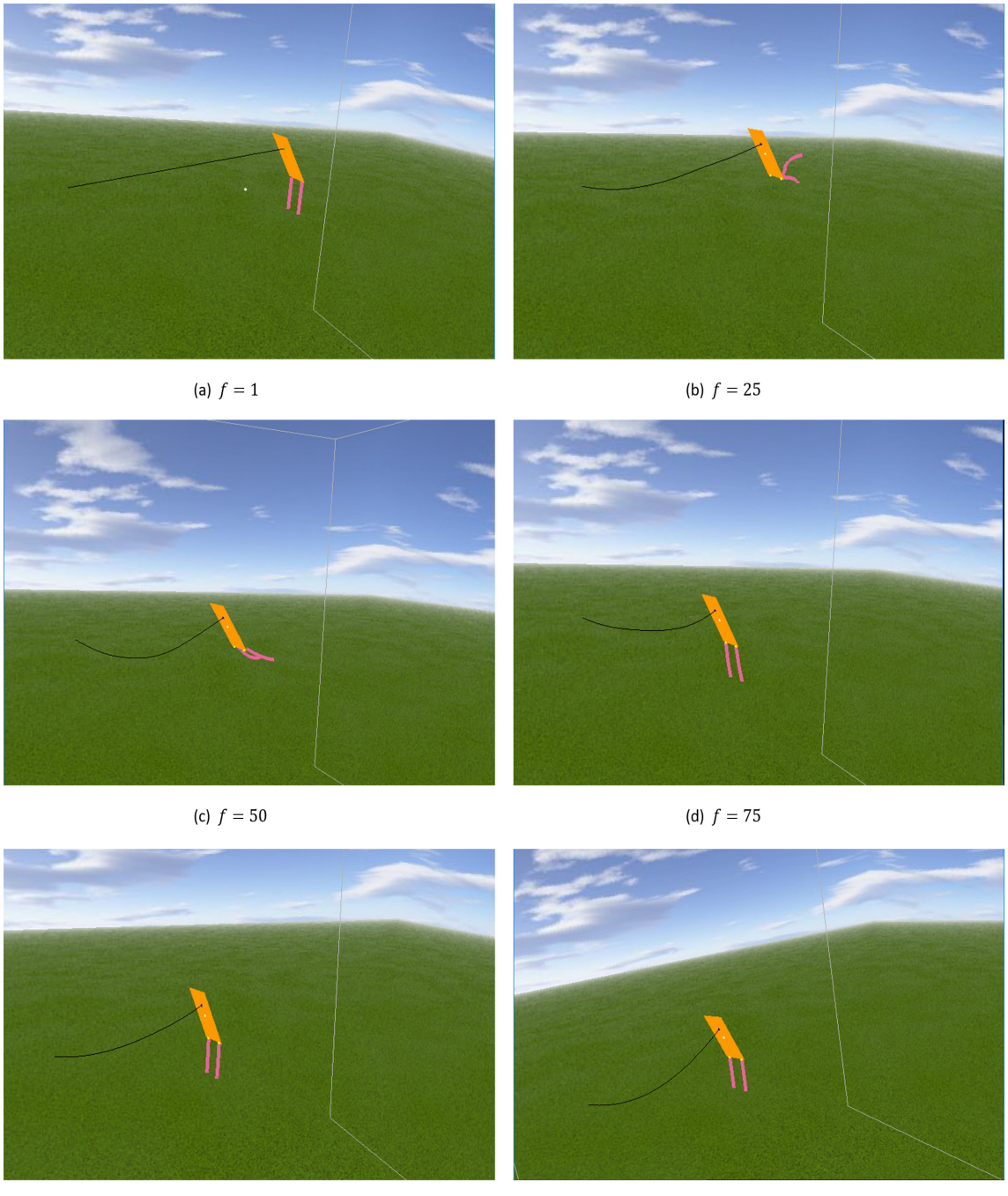
Type	Constant and Symbol	Value
Kite	Rest Density of Sensor ρ_{sensor}	1500 kg/m^3
	Mass of Sensor Particle m_{sensor}	0.002 kg
	Viscosity Value of Sensor μ_{sensor}	1.0 $kg/(s \cdot m)$
	Sensor Particle number n	169 per kite
	kernel radius h_s	0.056 m

Table 7.4: Physical constants of sensor particles used in kite simulation

kite. In Figure 7.2, we can see that the wind flow changes according to the behavior of the kite in real-time simulation, the black points are the wind particles and the green line represent the force direction of the forces acting on sensor particles. Then we change the position of connection point in Figure 7.3, finding that the final state of kite became vertical and the kite flies lower.

Figure 7.5 shows a kite train simulation. We use three rectangle kite models just the same as Figure 7.1. Each model has a sensor particle setting(see in Figure 6.1) respectively and are connected by a string, we calculate the kite string tension from two mass-spring segment on both sides of the sub-kite (see Figure 7.7). We weaken the influence of the rotation to keep the strain kite flies stable here. In Figure 7.4, we can see the wind particles around the kite train. The wind flow and the kite bodies have an effect on each other in every time step. Then, we modify the length between each two kites in Figure 7.6, we found that the second and third kite fall at frame $f = 20$, this is because the kites in the back can not interact with wind particles at the beginning, we allow a left movement of the string in Figure 7.6, and can see the acting effect on the kite.

We create a 3D cube to representing the box kite, for the purpose to extend our method to 3D complicated shapes. The example in Figure 7.8 illustrate our 3D box sensor particle setting, and we use this structure to simulate the box kite shown in Figure 7.9. Figure 7.10 shows the wind flow according to the 3D shape. Because the cube is not based on actual data, it can not fly as reality.

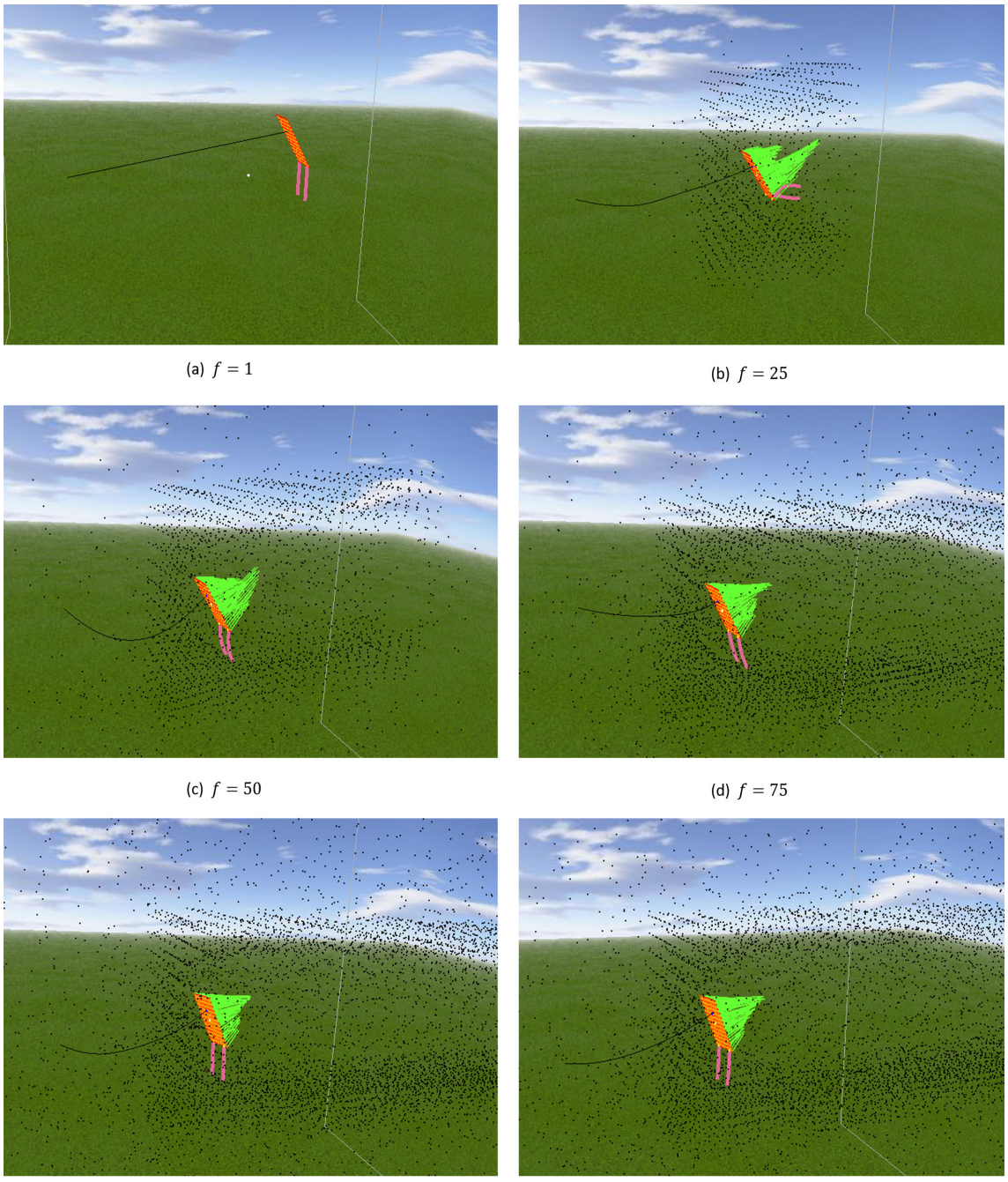


(a) $f = 1$ (b) $f = 25$

(c) $f = 50$ (d) $f = 75$

(e) $f = 100$ (f) $f = 125$

Figure 7.1: Kite consists of a rectangle body and two tails.

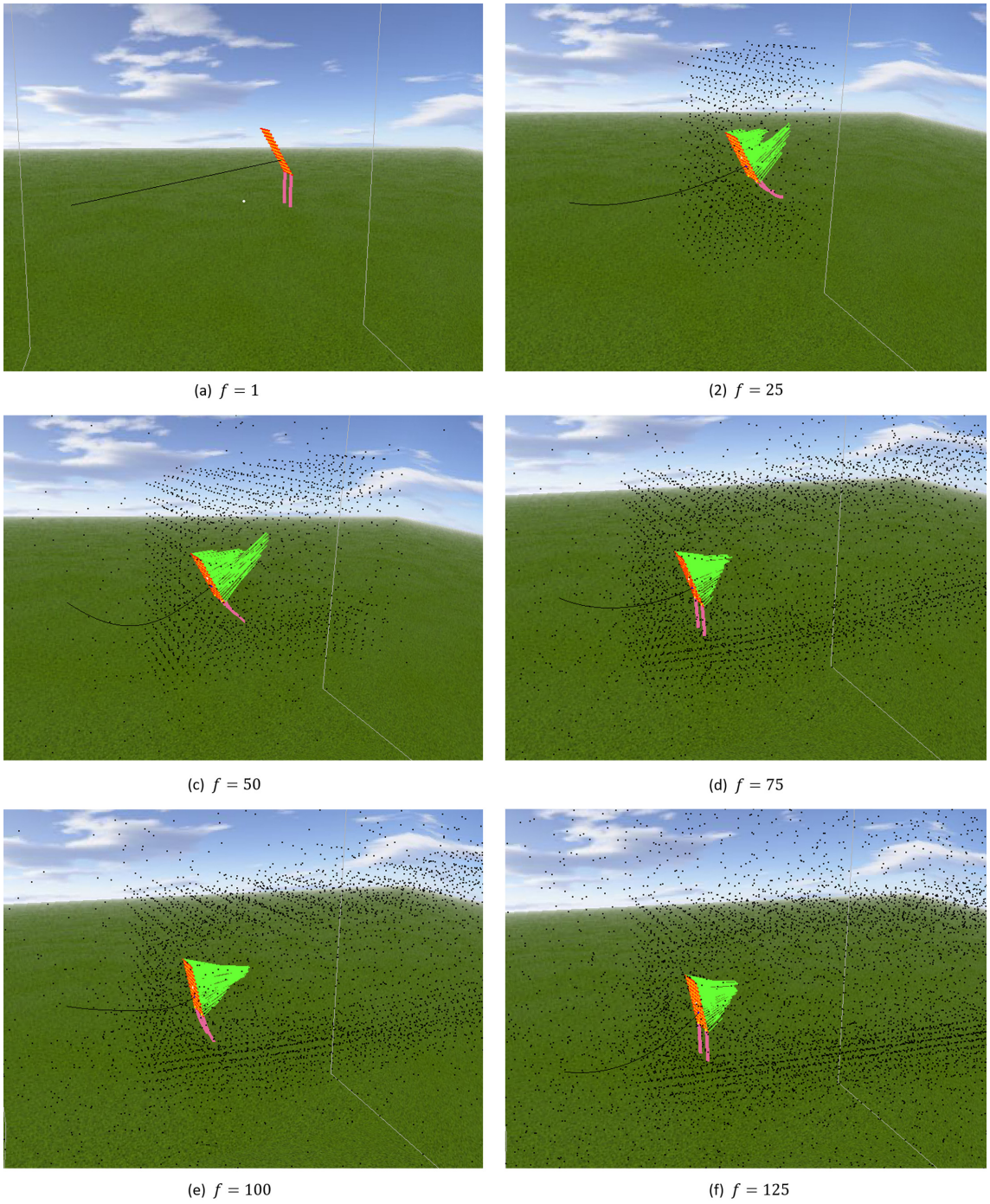


(a) $f = 1$ (b) $f = 25$

(c) $f = 50$ (d) $f = 75$

(e) $f = 100$ (f) $f = 125$

Figure 7.2: The wind flow changes according to the behavior of the kite. The green line shows the force direction of a sensor particle.

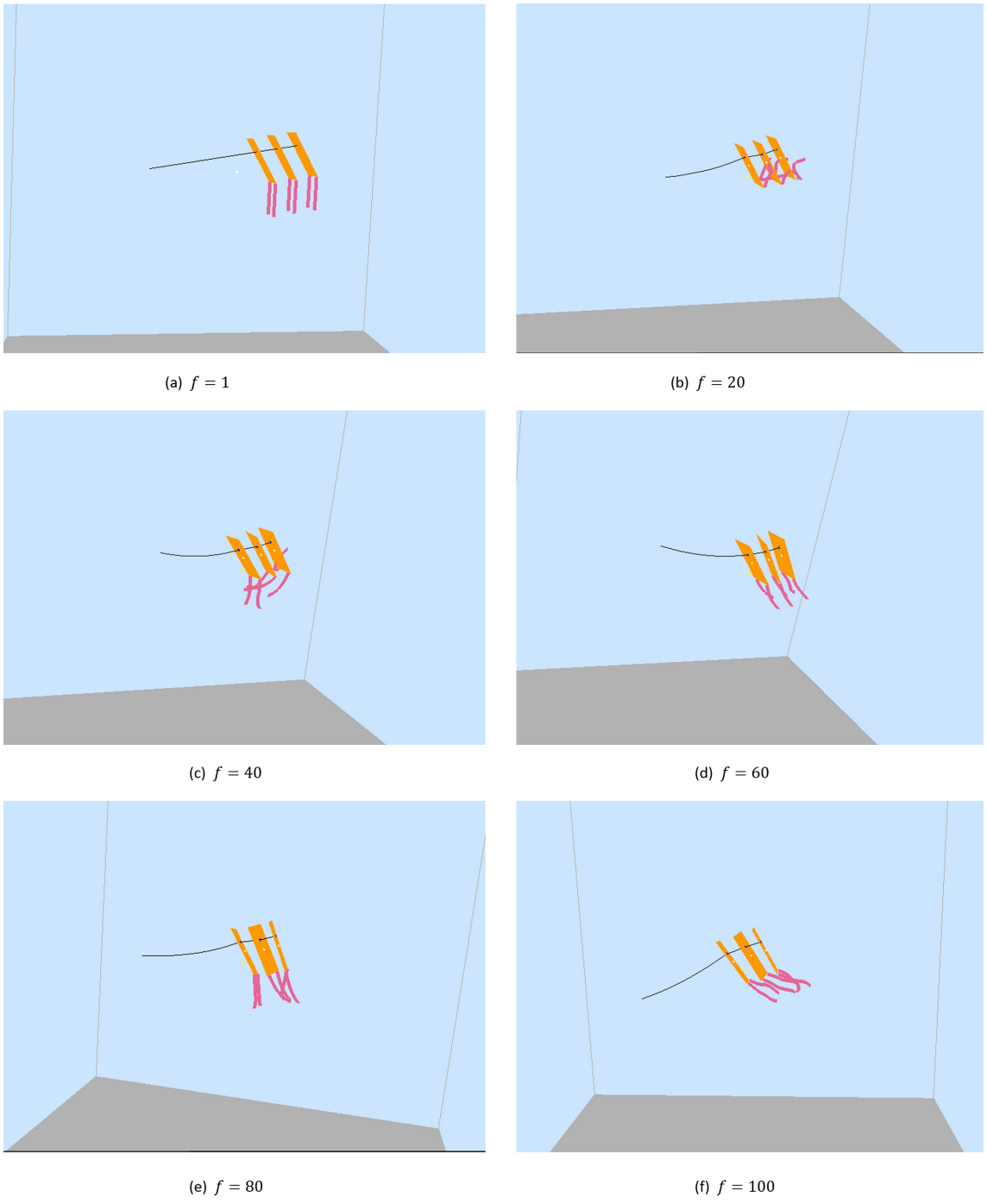


(a) $f = 1$ (2) $f = 25$

(c) $f = 50$ (d) $f = 75$

(e) $f = 100$ (f) $f = 125$

Figure 7.3: Change the position of connection point.

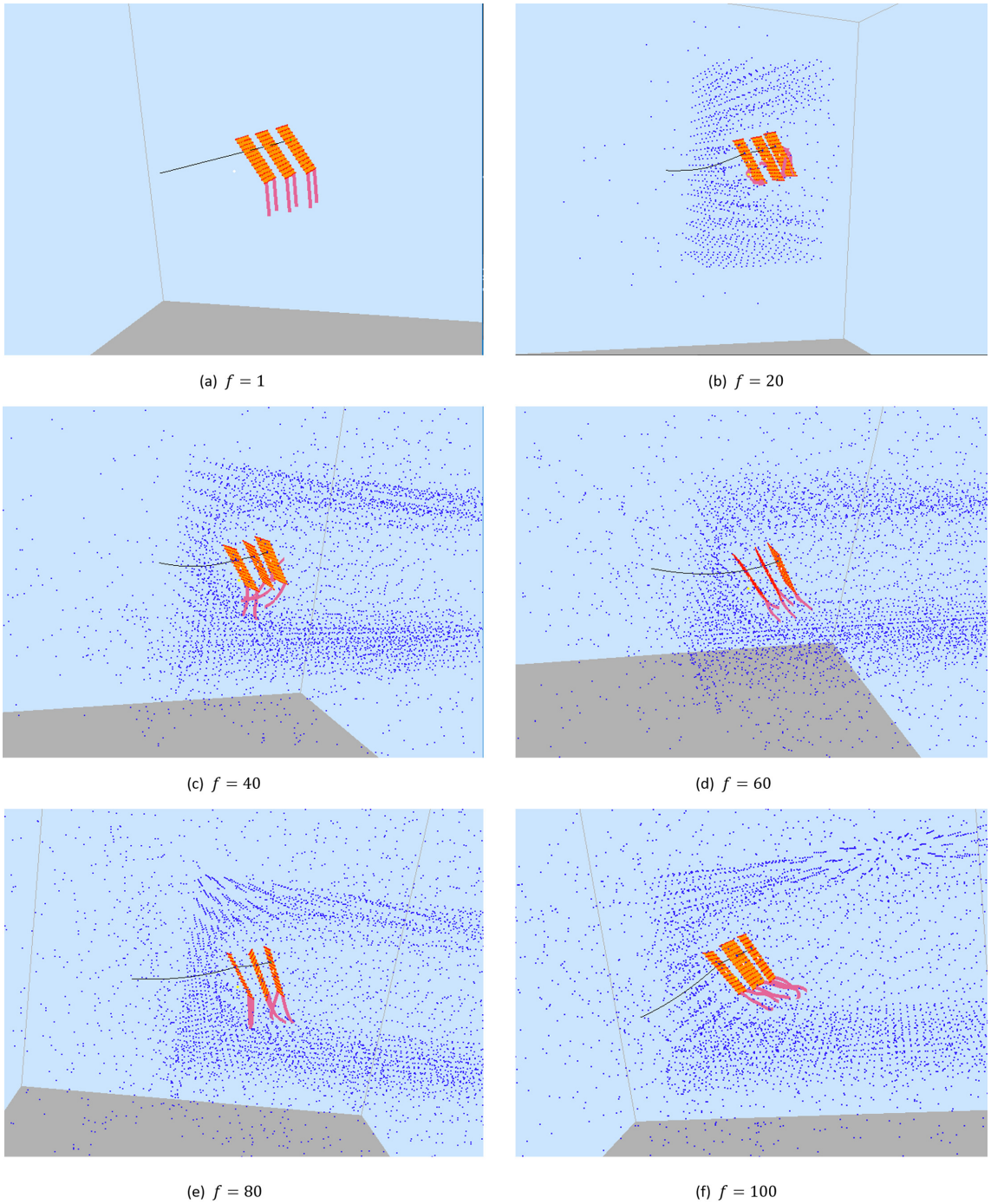


(a) $f = 1$ (b) $f = 20$

(c) $f = 40$ (d) $f = 60$

(e) $f = 80$ (f) $f = 100$

Figure 7.4: Movement of a kite train with wind particles drawing.

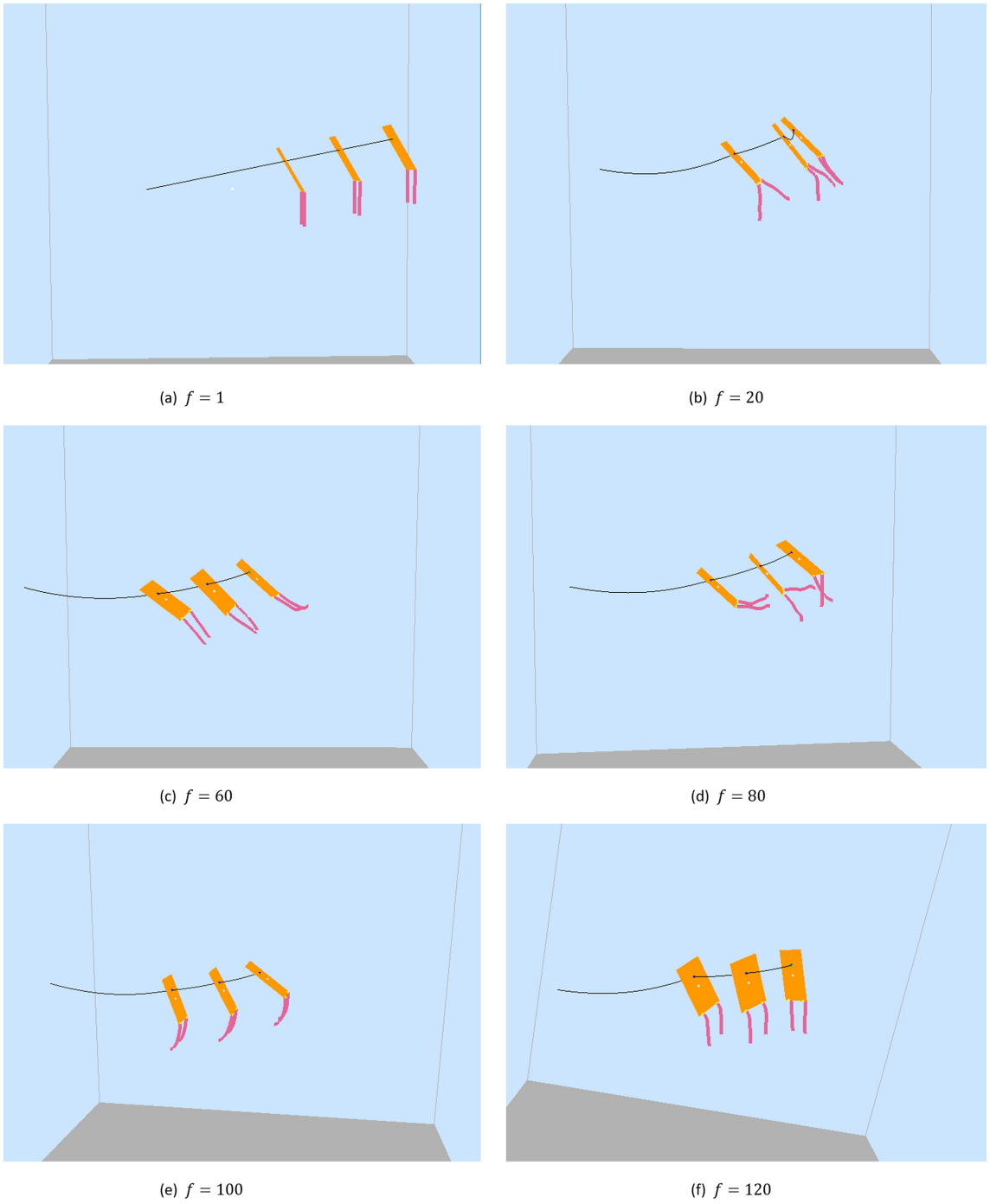


(a) $f = 1$ (b) $f = 20$

(c) $f = 40$ (d) $f = 60$

(e) $f = 80$ (f) $f = 100$

Figure 7.5: A kite train which consists of three sub-kite.



(a) $f = 1$ (b) $f = 20$

(c) $f = 60$ (d) $f = 80$

(e) $f = 100$ (f) $f = 120$

Figure 7.6: Change the segment size between sub-kites.

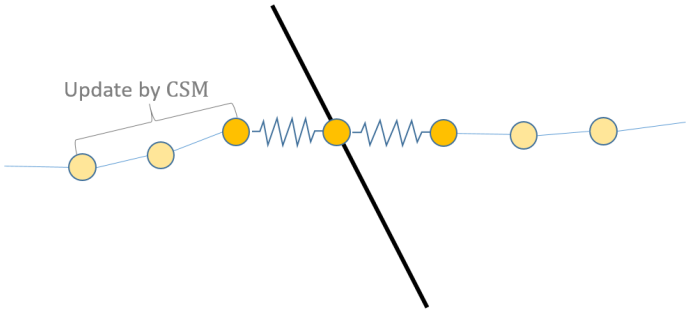


Figure 7.7: We calculate the force from two mass-spring segment on both sides of the sub-kite.

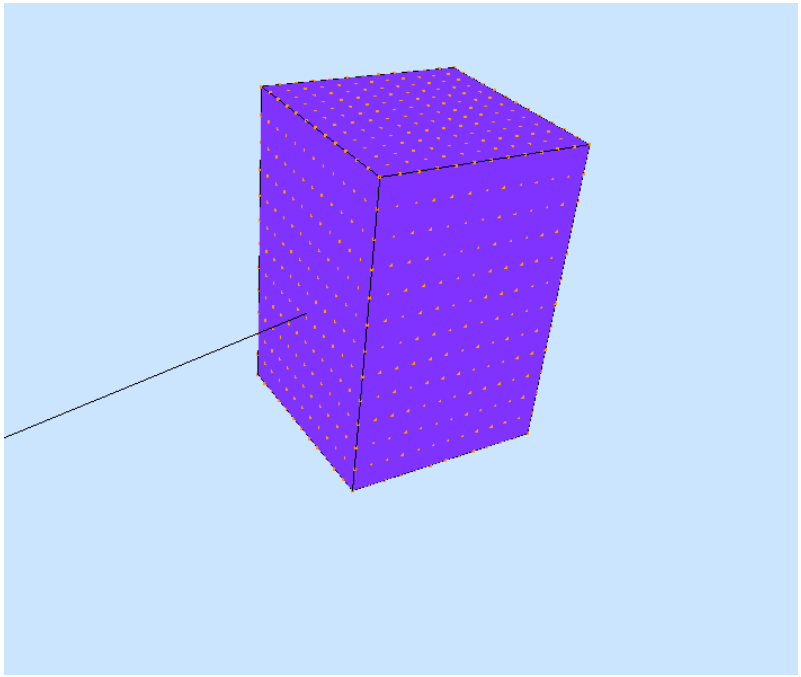
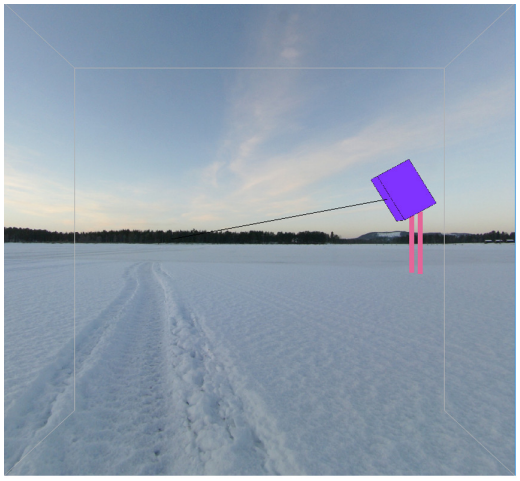
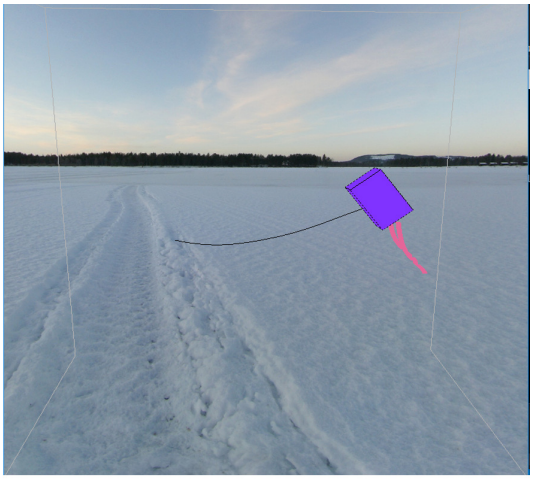


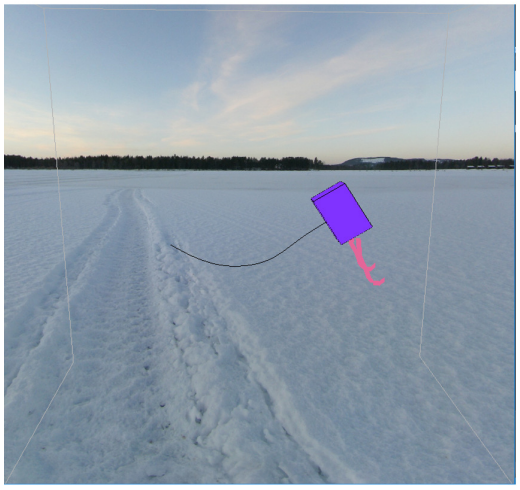
Figure 7.8: Sensor particles setting on polygon of the box shape.



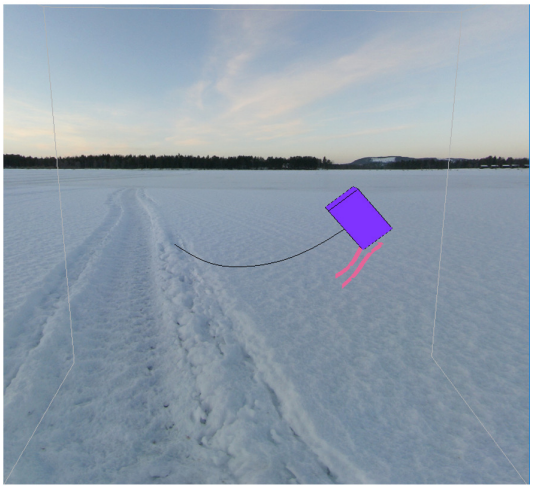
(c) $f = 1$



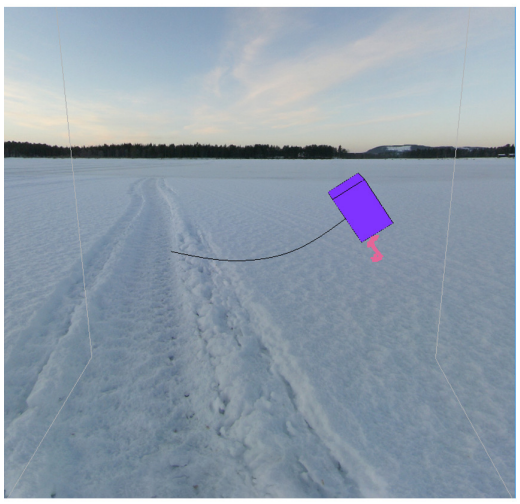
(c) $f = 25$



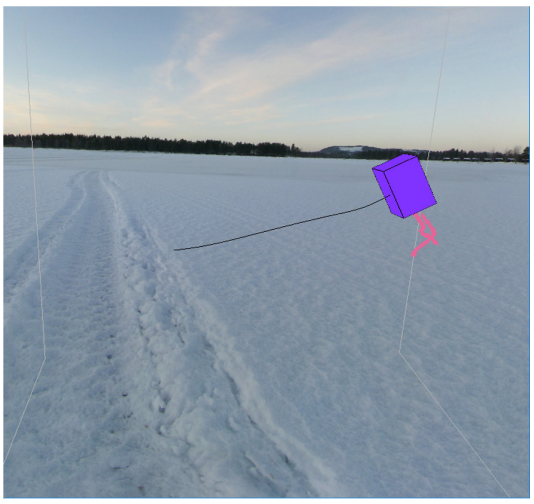
(c) $f = 50$



(d) $f = 75$

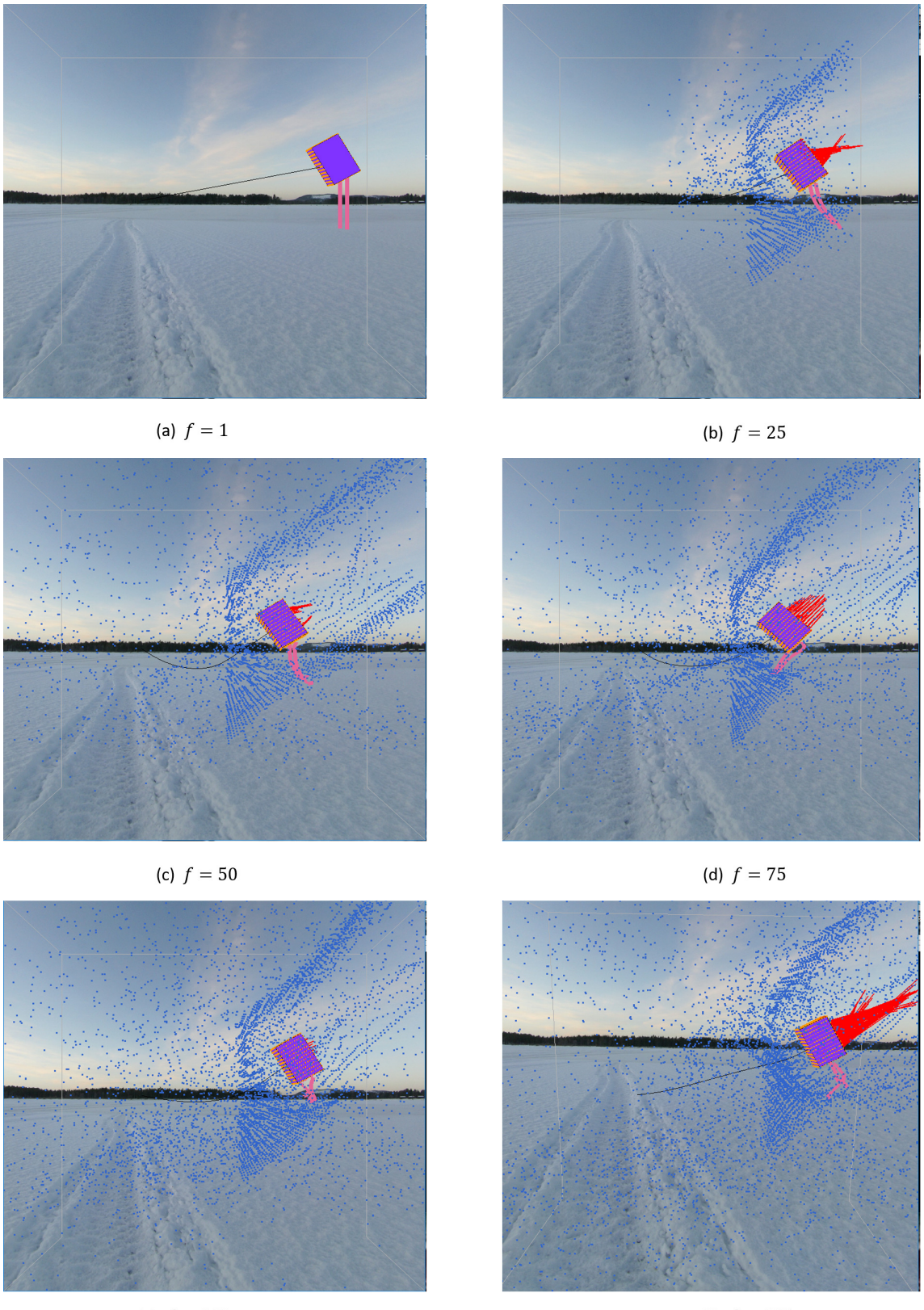


(e) $f = 100$



(f) $f = 125$

Figure 7.9: A 3D cube to represent a box kite.



(a) $f = 1$ (b) $f = 25$

(c) $f = 50$ (d) $f = 75$

(e) $f = 100$ (f) $f = 125$

Figure 7.10: Wind flow around the kite.

Chapter8

Discussion and Future Work

We have proposed a kite-flying simulation system with taking various kite shapes into account. We simulate the kite string by using the CSM method, it makes the kite string stable and more controllable, compared to the previous mass-spring model. We model the wind flow by SPH particles, each particles is trackable and is easy to calculate the interaction with the kite surface. We use a one-layer set of sensor particles to represent the shape of the kite, and calculate the exerted forces with the use of symmetric forces acting on wind particles. We also add tails to balance the kite if it is necessary. Because of the maldistribution of the one-layer sensor particle setting, we introduce the virtual volume to modify the density, pressure and friction of the wind particles correctly. We also use different settings of sensor particles to matching with the kite models. Influenced by the total forces exerted on kite, such as tension forces, aerodynamic forces and gravity force, we update the position and rotation of the kite in every time step.

Our approach can handle complicated kite shapes such as kite train and box kite according to the sensor particle settings. However, the kite model should be elaborated to keep the kite flying high. For example, add several tails or modify the kite model based on the reality types. Therefore, it is necessary to assist the user design of kite based on actual data. In our system, we weaken the influence of the moment of kites, because there are no particles behind the kite at the initial state, so each wind particle will influence the rotation of kite easily, which would lead to the kite flying unstable. In the future, we will add several kite springs (see the strings in Figure 1.2), just the same as the real kites, and account for the deflection of the kite body in order to enhance visual realism and improve the stability of the kite.

To make our system more active, we will take the environment influence into consider, such as trees and slope area which will lead to multi direction wind flows. And we can add the effect of turbulence around the kite edges to make the simulation more realistic. Another direction is to provide our system to an interactive kite design tool, in which users can create simple or complicated kites by visualizing the flying effect in real time. We can also expand the system by introducing the user-interaction function, such as using a haptic interface device like Leap Motion, to make the kite control more enjoyable.

Chapter9

Acknowledgements

First of all, I would like to thank Professor Makoto Fujisawa for inviting me to the Physics-based Computer Graphics Lab and giving me the chance to complete my master course in Japan. I am very happy to have an instructor like him not only for his kindness and patient, but also his great knowledge on the field of computer graphics.

Then, I would like to thank Professor Masahiko Mikawa for his valuable comments on our co-seminars, and interesting conversations about Japanese life.

I am happy to meet all the members in Physics-based Computer Graphics lab and intelligent robot lab, I feel very lucky to have you as my friends.

Finally, I would like to thank my family for their understanding and support, I am grateful to have you all.

References

[Akinci2012] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph.*, 31(4):62:1–62:8, July 2012.

[Baraff1998] David Baraff and Andrew P. Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 43–54, July 1998.

[Debunne2001] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 31–36, 2001.

[Desbrun1996] Mathieu Desbrun and Marie-Paule Gascuel. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96*, pages 61–76. Springer-Verlag New York, Inc., 1996.

[James1999] Doug L. James and Dinesh K. Pai. Artdefo: Accurate real time deformable objects. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 65–72. ACM Press/Addison-Wesley Publishing Co., 1999.

[Losasso2004] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 457–462. ACM, 2004.

[Martin2015] Tobias Martin, Nobuyuki Umetani, and Bernd Bickel. Omniad: Data-driven omni-directional aerodynamics. *ACM Trans. Graph.*, 34(4):113:1–113:12, July 2015.

[Muller2002] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable real-time deformations. In *Proceedings of*

the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '02, pages 49–54. ACM, 2002.

[Muller2003] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 154–159. Eurographics Association, 2003.

[Muller2005] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. *ACM Trans. Graph.*, 24(3):471–478, July 2005.

[Muller2007] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *J. Vis. Comun. Image Represent.*, 18(2):109–118, April 2007.

[Mller2004] Matthias Mller, Matthias Teschner, and Markus Gross. Physically-based simulation of objects represented by surface meshes. In *In Proc. Comput. Graph. Int*, pages 156–165, 2004.

[Nealen2006] Andrew Nealen, Matthias Mueller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 2006.

[Okamoto2009] Taichi Okamoto, Makoto Fujisawa, and Kenjiro T. Miura. An interactive simulation system for flying japanese kites. In *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games*, Sandbox '09, pages 47–53. ACM, 2009.

[Pirk2014] Sören Pirk, Till Niese, Torsten Hädrich, Bedrich Benes, and Oliver Deussen. Windy trees: Computing stress response for developmental tree models. *ACM Trans. Graph.*, 33(6):204:1–204:11, November 2014.

[Ramakrishnananda1999] Balajee Ramakrishnananda and Wong Kok Cheong. Animating bird flight using aerodynamics. In *ACM SIGGRAPH 99 Conference Abstracts and Applications*, SIGGRAPH '99, pages 230–. ACM, 1999.

[Rungjiratananon2010] W. Rungjiratananon, Y. Kanamori, and T. Nishita. Chain shape matching for simulating complex hairstyles. *Computer Graphics Forum*, 29(8).

[Stam1999] Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.

[Teran2003] J. Teran, S. Blemker, V. Ng Thow Hing, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 68–74, 2003.

[Umetani2014] Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph.*, 33(4):65:1–65:10, July 2014.

[Wang2007] Z. Jane Wang. Aerodynamic efficiency of flapping flight: analysis of a two-stroke model. 211(2):234–238, 2007.

[Wu2003] Jia-chi Wu and Zoran Popović. Realistic modeling of bird flight animations. *ACM Trans. Graph.*, 22(3):888–895, July 2003.