Real-time Rendering of Soap Bubbles Taking into Account Light Interference

Kei Iwasaki Keichi Matsuzawa Tomoyuki Nishita
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
Phone: +81.3.5841.4096 Fax: +81.3.5803.7288
{kei-i, keichi, nis}@nis-lab.is.s.u-tokyo.ac.jp

Abstract

In the field of computer graphics, simulation of physical phenomena is of great interest. We focus on the optical effects of soap bubbles. Soap bubbles have fascinating coloration and interesting physical properties. Therefore they are useful for the entertainment such as movies and games. Soap bubbles change their shapes by surface tension and external forces, and therefore their surface thickness also changes. Since the thickness of the soap bubble is several hundred nanometers, interference of the light occurs. This paper proposes a fast rendering method for the soap bubbles taking into account light interference and dynamics. In our method, the reflectivities of the thin film that is the cause of the light interference are calculated in advance and stored as textures. This makes it possible to render the deformable soap bubbles in real-time.

Keywords: light interference, graphics hardware, real-time animation

1. Introduction

The simulation of natural phenomena is one of the most challenging research fields in computer graphics. Soap bubbles have interesting physical properties and attractive coloration [15]. In this paper, we propose a real-time rendering method for soap bubbles taking into account light interference. Soap bubbles form very thin layers. Since the thickness of the soap bubble is close to the wavelength of the visible light, fringes due to the light interference can be seen. Several methods have been developed to render the fringes [1, 2, 11, 12]. These methods employed a ray tracing algorithm to render the soap bubbles. Although realistic images can be generated by the ray tracing method, the computational cost is expensive. Thus, it is only suitable for images and movies that do not require real-time response. However, applications such as virtual reality and games require real-time response. Therefore, the demand to accelerate the rate of rendering is high.

To address this problem, this paper presents a real-time rendering method for deformable soap bubbles taking into account light interference. In the proposed method, the reflectivities of the thin film are calculated in advance and stored as textures. Moreover, the rendering of objects reflected at the bubble surface is performed by using the dynamic cubical environment map method.

2. Previous Work

Several methods have been proposed to simulate and render the soap bubbles.

Dias [1] developed a method that modeled the interference light phenomenon and displayed the soap bubble with fringes. Li and Peng [11] extended Dias' method to take the polarization of light into account. Icart et al. [9] simulated two dimensional soap froth. Glassner [4, 5] modeled the clusters of two or three soap bubbles by using analytical solutions for the geometry of soap bubble clusters. Ďurikovič [3] treated a bubble as a set of particles and triangle patches. He simulated the deformation of soap bubbles due to wind forces and contacts with other objects. Kück et al. [10] simulated and rendered the liquid forms by representing foam bubbles as fixed-size spheres. Realistic images of soap bubbles can be generated by using these methods. However, these methods used a ray tracing algorithm whose computational cost is expensive. There have been several related work about rendering light interference. Dias [2] proposed a rendering method of optical phenomena of Newton's ring. Hirayama et al. [7, 8] developed a rendering method for the interference effect on multilayer films. They took into account the smoothness of the surfaces coated with multilayer films and demonstrated several materials such as alminum, silicon, and copper. Sun [14] presented a method for transforming colors into spectrum and rendering light interference. These methods also employed a ray tracing algorithm for the rendering.

In this paper, we accelerate the rendering of soap bubbles taking into account light interference. The proposed method can calculate the light interference caused by the variation

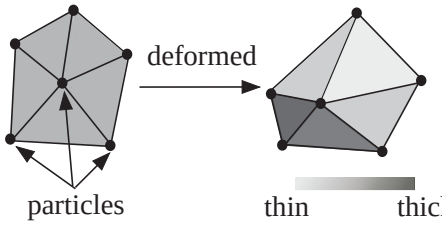


Figure 1. Representation of soap bubble by particles.

of the film thickness due to the interactions with wind and other objects in real-time.

3. Simulation of Soap Bubble Dynamics

The simulation method of the soap bubble dynamics is based on Ďuricovič model [3]. The surface of the soap bubble is represented by triangular meshes and the vertices of the triangular meshes are treated as particles (see Figure 1). The soap bubble dynamics is simulated by solving the motion equation of each particle p_i.

$$m_i\ddot{x_i} = F_{internal} + F_{external} - \gamma\dot{x_i}, \tag{1}$$

where m_i is the mass of particle p_i and x_i the position of particle p_i. In our method, each particle p_i has the same mass m. $F_{internal}$ is the internal force and $F_{external}$ is the external force. The last term, $\gamma\dot{x}$, is the effect of damping, where γ is the damping constant. The internal force $F_{internal}$ consists of surface tension and the pressure difference inside and outside the bubble. The external force $F_{external}$ consists of gravity, drag force, force due to collisions with other bubbles, and force due to contacts with other objects. The calculation method for each force is described in [3].

To calculate the light interference, the film thickness of the soap bubble must be computed. As shown in Figure 1, thickness d_i at particle p_i is calculated from the areas of the triangles that include particle p_i in the following equation:

$$d_i = c_d \frac{m}{\sum_{j \in F(i)} A_j}, \tag{2}$$

where $F(i)$ is a set of indices of triangle patches including particle p_i, A_j is the area of triangle F_j, c_d is the scaling factor. The velocity $\dot{x_i}$ is calculated from $\ddot{x_i}$ by the explicit Euler method and then the position x_i is calculated from $\dot{x_i}$.

4. Rendering Soap Bubbles

4.1. Basic Idea

The radiance L_p from point P on the bubble surface to the viewpoint is expressed by the following equation.

$$L_p(\lambda) = (1 - R(\lambda,\theta)) \times L_{it}(\lambda) + R(\lambda,\theta) \times L_{ir}(\lambda) \tag{3}$$

where λ is the wavelength, θ is the incident angle of the incident light, $(1-R(\lambda,\theta))L_{it}$ the transmitted light, $R(\lambda,\theta)L_{ir}$ the reflected light, and $R(\lambda,\theta)$ the reflectivity. We use the previous method developed by Li et al. [11] to calculate reflectivity $R(\lambda,\theta)$. $R(\lambda,\theta)$ is expressed by the following equation:

$$\begin{aligned} R(\lambda,\theta) &= 2{R_\perp}^2 \frac{1-\cos(\delta(\lambda,\theta))}{1+{R_\perp}^4-2{R_\perp}^2\cos(\delta(\lambda,\theta))} \\ &+ 2{R_\parallel}^2 \frac{1-\cos(\delta(\lambda,\theta))}{1+{R_\parallel}^4-2{R_\parallel}^2\cos(\delta(\lambda,\theta))}, \end{aligned} \tag{4}$$

where $R_\perp$ and $R_\parallel$ are the amplitude reflectivities of perpendicular and parallel to the plane of incidence, respectively. The phase difference $\delta(\lambda,\theta)$ is calculated from the following equation:

$$\delta(\lambda,\theta) = \frac{4\pi}{\lambda} nd\cos\theta, \tag{5}$$

where n is the refractive index of the soapy water, d the thickness of the thin film. As shown in Equation (5), the phase difference δ depends on only d, θ and λ. Therefore, $R(\lambda,\theta)$ also depends on d, θ, λ, and is rewritten as $R(\lambda,\theta,d)$.

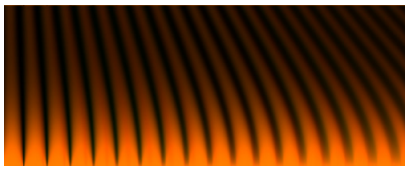
4.2. Reflected Light

Incident light $L_{ir}(\lambda)$ consists of light from the light source and light from the environment. We calculate the reflection of the light sources (in this paper, we deal with very small area light sources) and that of the environment separately since the dynamic ranges are different. We first describe the rendering method for the reflection of the light sources, then the rendering method for that of the environment is discussed.

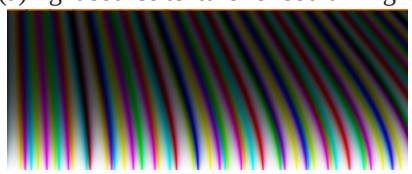
4.2.1 Reflected light from the light sources

The energy distribution of the light source $E(\lambda)$ is prepared since some light sources are difficult to represent at only RGB components (e.g. sodium light). $E(\lambda)$ is sampled at every 10 [nm] from 350 [nm] to 800 [nm]. We calculate $R(\lambda,\theta,d)E(\lambda)$ for each wavelength and finally convert the reflected light to RGB components to display on a color monitor. We calculate RGB components $R(\lambda,\theta,d)E(\lambda)$ by integrating the product of $R(\lambda,\theta,d)E(\lambda)$ and the color matching function over the entire visible spectrum [6]. The integration $R(\lambda,\theta,d)E(\lambda)$ over the entire visible spectrum is calculated for each d and θ. Finally, $R(\lambda,\theta,d)E(\lambda)$ is stored as a texture that is called *light source texture* whose parameters are d and θ.

Figure 2 shows the textures for calculating reflected light. Figure 2(a) shows the light source texture for a sodium light. Horizontal axis represents the thickness of the film (from 0 [nm] (left) to 2000 [nm] (right)). Vertical axis represents the cosines of incident angles (from 0.0 (bottom) to 1.0 (top)).



(a) light source texture for sodium light



(b) reflectivity texture

Figure 2. Textures for calculating reflected light.

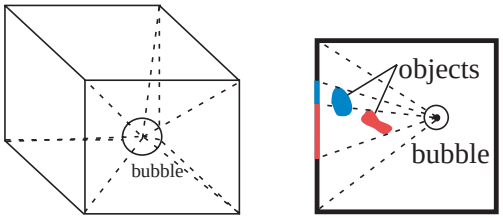
4.2.2 Reflected light from the environment

In the proposed method, the scene is framed by an adequately large cube as shown in Figure 3(a). Incident light from the environment is represented by environment map textures that is usually sampled at RGB components. Therefore, reflectivity $R(\lambda, \theta, d)$ is also sampled at RGB components and stored as a texture whose parameters are d and θ (see Figure 2(b)). Although several methods [13, 14] of transformation from RGB color to spectrum may be applied to the cube map textures, the proposed method can render visibly convincing soap bubbles by taking account of RGB components of reflected light from environment. To render the reflected object near the soap bubble, we dynamically generate cube map textures by rendering the background and neighbour objects except for bubbles as seen from the current center of the bubble (see Figure. 3(b)). This makes it possible to render both the neighbour objects and the distant environment. The creation of dynamic cube map textures is performed by texture rendering. We use an off-screen buffer technique proposed by Wynn [16] to accelerate the process of texture rendering. Reflected radiance at the bubble surface is calculated by multiplying the texture for the reflectivities by dynamic cube map textures.

Though this technique can not express multiple reflections or reflected bubble images in other bubbles, the reflectivity of soapy water is small and therefore the effect of multiple reflection can be ignored. The area of the soap bubble surface where the reflectivity is high is almost parallel to the viewing ray (that is, the reflectivity is high if the viewing ray is almost perpendicular to the normal of the soap bubble surface) and the projected area onto the screen is small, and therefore the effect of multiple reflection can be ignored.

4.3. Transmitted Light

Transmitted light $(1 - R(\lambda, \theta, d))L_{it}(\lambda)$ is the light that comes from the background of the soap and is partially



(a) scene framed by cube (b) dynamic cube map

Figure 3. Dynamic cubical environment map.

transmitted through the thin film of the soap bubble. Transmitted light is stored in the frame buffer after the objects to the rear of the soap bubbles are rendered. Since the film of the soap bubble is very thin, the difference between the direction of the refracted ray and the direction of the incident ray can be ignored. Therefore, we can ignore the effects of the refraction of the viewing ray.

4.4. Rendering Process

Since the soap bubble is transparent, we must render surfaces of the soap bubbles in order of decreasing distance from the viewpoint. Therefore, polygons of the bubble have to be sorted by each depth from the viewpoint. The bubble surfaces, generated by recursively subdividing an icosahedron three times, are 1,280 triangles and if there are several such bubbles, the computational time to sort becomes large. However, bubbles have the following properties.

- Bubbles can not penetrate each other.
- The whole shape is almost convex.

The viewing ray can rarely intersect the surface of the same bubble more than three times according to the properties. The bubbles are sorted by the position of their center of mass in the proposed method. Back faces of the soap bubble are rendered first, then front faces are rendered afterward.

The procedure of the rendering soap bubbles is summarized as follows.

1. Render the background.
2. Repeat the following process for each soap bubble in descending order from the viewpoint.
   (a) Create the cube map textures.
   (b) Map the reflectivity texture onto the soap bubble. Render the bubble by multiplicative blending with the frame buffer (set the blending factor `GL_ONE_MINUS_SRC_COLOR` to calculate $1 - R(\lambda, \theta, d)$).
   (c) Map the cube map textures and the texture of reflectivity to the soap bubble by using multiplicative multitexturing function. Render the bubble by additive blending with the bubble rendered at step (b).

(d) Map the light source texture to the soap bubble. Render the bubble by additive blending with the bubble rendered at step (c).

5. Results

The following images are rendered using a windows machine (PentiumM 1.7GHz, 512MB RAM, and QuadroFX Go700). Each bubble is created by recursively subdividing an icosahedron three times.

5.1. Comparison between Proposed Method and Ray Tracing

Figure 4 shows the comparison of images rendered by using the ray tracing method and the proposed method. There are two bubbles and one plane. The sizes of the images are 400×400. In Figure 4(a), the rendering time is about 34[sec] by using the ray tracing method. In Figure 4(b), the rendering time is about 0.03[sec] (19fps including the simulation time 0.02[sec]) by using our method. That is, the proposed method is 1000 times faster than the ray tracing method. As shown in these figures, the quality of Figure 4(b) is almost the same as that of Figure 4(a). In Figure 4(c), the rendering time is about 0.01[sec] (29fps including the simulation) by using our method without dynamic texturing. This result suggests that we can render the soap bubbles in real-time.

5.2. Interaction with Objects

Figures 5, 6, and 7 show stills from animations. Figure 5 shows stills from an animation of deformed bubbles due to the wind force. Wind blows from left to right. As shown in this figure, the variation of the color due to light interference can be rendered. Figure 6 shows examples of a collision of bubbles and a plane. Figure 7 shows stills from an animation of a collision of two bubbles. The computational time for simulating soap bubble dynamics is about 0.02 [sec].

These results demonstrate that the proposed method can generate realistic images of soap bubbles almost in real-time. The accompanying movies[1] show more impressive results and demonstrate the usefulness of our method. The accompanying movies consist of four animations (animation of deformed soap bubbles due to wind, animation of collision with plane, animation of collision with bubbles, and screen captured animation of demonstration).

6. Conclusion & Future Work

In this paper, we have proposed a rendering method for soap bubbles. Since the thickness of the soap bubble becomes thinner than the wavelength of the visible ray, interference of the light occurs. This optical effect must be taken into consideration to create realistic images of the soap bubbles. In our method, textures that store the reflectivities are precalculated. This makes it possible to achieve the interactive rendering of the soap bubble by using graphics hardware. Moreover, our method can render the reflection of both a distant environment and of a reflected object that is near the soap bubbles. An object reflected in the soap bubble is rendered efficiently by a hardware-accelerated dynamic cubical environment map.

In future work, we want to apply our rendering method to other materials with thin films such as pearls and opals.

[1]The movies can be seen at the following URL.
http://nis-lab.is.s.u-tokyo.ac.jp/~kei-i/bubble/

Acknowledgments

We would like to thank Prof. Nelson Max (Lawrence Livermore National Laboratory) for providing useful comments on the manuscript.

References

[1] L.M. Dias,"Ray Tracing Interference Color," *IEEE Computer Graphics and Applications*, Vol.11, No.2, pp.54-60, 1991.

[2] L.M. Dias, "Ray Tracing Interference Color: Visualizing Newton's Rings," *IEEE Computer Graphics and Applications*, Vol.14, No.3, pp.17-20, 1994.

[3] R. Ďurikovič, "Animation of Soap Bubble Dynamics, Cluster Formation and Collision," *Computer Graphics Forum*, Vol.20, No.3, pp.67-76, 2001.

[4] A. Glassner, "Soap Bubbles: Part1," *IEEE Computer Graphics and Applications*, Vol.20, No.5, pp.76-84, 2000.

[5] A. Glassner, "Soap Bubbles: Part2," *IEEE Computer Graphics and Applications*, Vol.20, No.6, pp.99-109, 2000.

[6] R. Hall, "*Illumination and Color in Computer Generated Imagery*," Springer-Verlag, Berlin, 1989.

[7] H. Hirayama, K. Kaneda, H. Yamashita, Y. Monden, "An Accurate Illumination Model for Objects Coated with Multilayer Films," *Computers & Graphics*, Vol.25, pp.391-400, 2001.

[8] H. Hirayama, K. Kaneda, H. Yamashita, Y.Yamaji, Y. Monden, "Visualization of Optical Phenomena Caused by Multilayer films Based on Wave Optics," *The Visual Computer*, Vol.17, No.2, pp.106-120, 2001.

[9] I. Icart, D. Arquès, "An Approach to Geometrical and Optical Simulation of Soap Froth," *Computers & Graphics*, Vol.23, No.3, pp.405-418, 1999.

[10] H. Kück, C. Vogelgsang, G. Greiner, "Simulation and Rendering of Liquid Foams," *Proc. Graphics Interface'02*, pp.81-88, 2002.

[11] J. Li, Q. Peng, "A New Illumination Model for Scenes Containing Thin Film Interference," *Proc. Pacific Graphics'96*, pp.133-146, 1996.

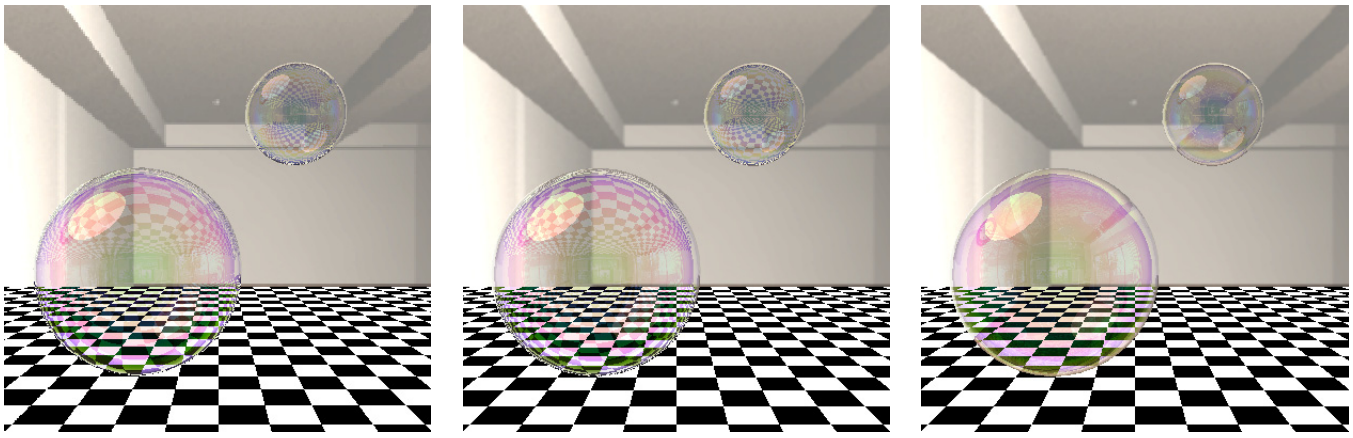
[12] B. Smits, G. Meyer, "Newton's Colors: Simulating Interference Phenomena in Realistic Image Synthesis," *Proc. Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, pp.185-194, 1990.

[13] B. Smits, "An RGB to Spectrum Conversion for Reflectances," *Journal of Graphics Tools*, Vol.4, No.4, pp.11-22, 1999.

[14] Y. Sun, F.D. Fracchina, T.W. Calvert, M.S. Draw, "Deriving Spectra from Colors and Rendering Interference," *IEEE Computer Graphics and Applications*, Vol.19, No.4, pp.61-67, 1999.

[15] D. Weaire, S. Hutzler, "*The Physics of Foams*," Clarendon Press, Oxford, 1999.

[16] C. Wynn, "Using P-Buffers for Off-Screen Rendering in OpenGL," http://developer.nvidia.com/.



(a) ray tracing method (b) proposed method (c) proposed method without dynamic texturing

Figure 4. Comparison of images rendered by different methods.

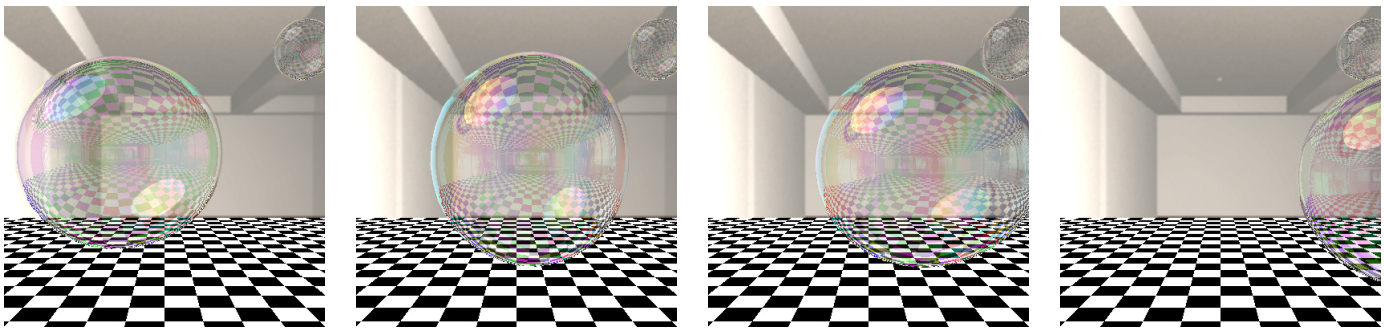


Figure 5. An animation of deformed bubbles due to wind.

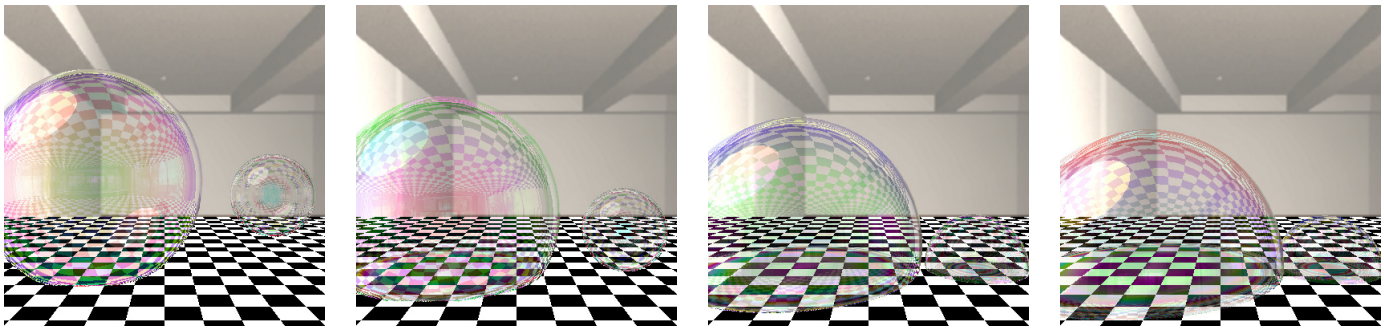


Figure 6. An animation of collision with plane.

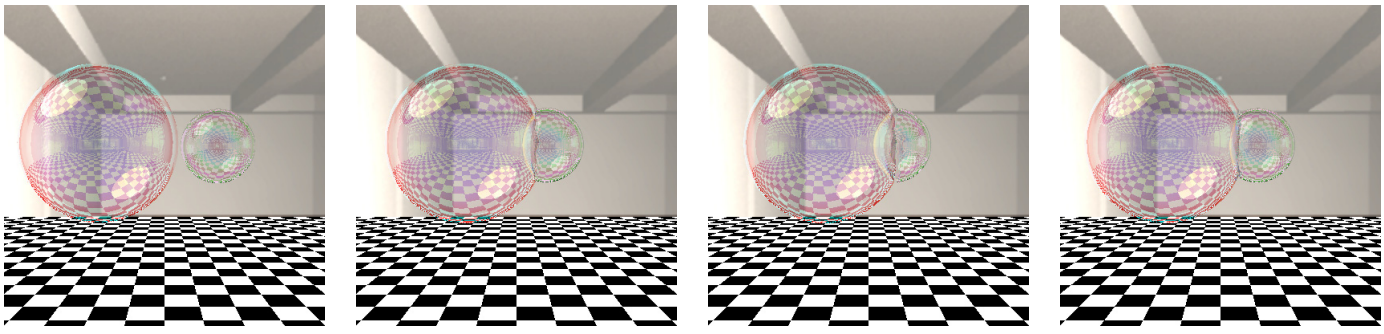


Figure 7. An animation of collision with bubbles.