A Framework for User Control on Stylised Animation of Gaseous Phenomena

Fabian Di Fiore Johan Claes Frank Van Reeth
Limburgs Universitair Centrum
Expertise Center for Digital Media
Universitaire Campus
B-3590 Diepenbeek, Belgium
e-mail: {fabian.difiore, johan.claes, frank.vanreeth}@luc.ac.be

Figure 1: Some snapshots of a barbecue rendered in a 'cartoon-ish' style.

Abstract

We present a novel approach to create stylised animations of gaseous phenomena. Existing computer-assisted systems employ computational physics-based approaches which generate 3D effects. Unfortunately, these effects usually don't look like traditional animation, nor can the user freely design the behaviour of the animation.

Our approach combines benefits from existing 2D and 3D approaches integrating them in an unusual but effective way. Extending the application of particle systems, the particle paths vary both with time and with the 3D viewing direction. Rendering features such as smoothly varying common outlines and speedlines help to preserve the animator's artistic style. The incorporation into a structured 2D modelling and animation environment enables a stylised animation exhibiting a convincing frame-to-frame coherence and allows 3D camera movement without aliasing artefacts.

The provided solution demonstrates how an animator can remain in full control of a stylised process for effects animation and how one framework is suitable for a wide range of effects.

Keywords: gaseous phenomena, stylised animation, computer-assisted animation, non-photorealistic rendering, particle systems

1 Introduction

Traditionally, stylised animations exhibit objects and characters with a hand-drawn look and resemble real life without trying to be an exact copy of reality. However, characters and objects are only parts of stylised animation; animated effects such as water, fire and smoke add realism — in a stylised manner — and make the whole more vivid.

The production of high-end feature films counts with enough resources to enable dedicated programmers and animators working closely together in an elaborate process of trial and error to achieve outstanding effects. In our research, we look for solutions to be used in smaller-scale productions where animators have to find their way more independently. Except from the traditional animation channel, we also target stylised effects for virtual worlds which due to the ever-increasing availability of high-bandwidth communications become more and more common-place.

Nowadays, for our targeted kind of productions gaseous phenomena are either drawn frame by frame (usually in short loops) or one has to revert to 3D computer generated effects and place these effects as layers over the main scene [1]. Our goals are quite different, as we want to offer animators full control over every aspect of a lively animation and to integrate these effects seamlessly into the rest of their artwork.

To establish these goals, we present a hybrid approach that combines existing 2D and 3D tech-

niques. On the one hand, necessary 3D information is incorporated by letting the animator draw the animation paths and acceleration curves. This is necessary to provide for frame-to-frame coherence. On the other hand, structured 2D modelling and animation techniques are used to preserve the animator's creative freedom and to create convincing 3D-alike animations starting from pure 2D information.

This paper is organised as follows. We start with an overview of related work on approaches involving explicit user control, as well as approaches exploiting physics based fluid dynamics. Then, modelling and animation in structured 2D is elucidated, followed by the central theme of our paper, rendering and animating stylised gaseous phenomena. We end with some clarifying results, our conclusions and topics for future research.

2 Related Work

2.1 User-controlled Behaviour

In stylised animation (and many other cases), realistic behaviour is not always desired, but there's a need for fake, yet very impressive or dramatic effects [2].

An appealing procedural paradigm for stylised animation of gaseous phenomena is due to Jinhui Yu [3, 4, 5]. He constructs cartoon effects by simulating the hand-drawing process (relying on reverse-engineering animation practices) through synthetic, computational means. Although this system emulates the stylish appearance and look and feel of traditional animation, little freedom is preserved for the animator: all cartoon effects are bound by fixed procedures with no possibility to influence the behaviour.

For the animated feature film *'The Prince of Egypt'* Patrick Witting presented a computational fluid dynamics system to produce smoke, water, and other effects [1]. Images or animation sequences are used to initialise temperature fields which cause dynamic buoyancy-driven vortices to evolve. Initially the animation is driven by the animators, since they are responsible for supplying these images. A major drawback, however, is that this freedom only counts at the start of the animation as the animator is limited to set up the system (i.e. supplying one or more initial images) after which physically accurate equations — time-dependent compressible Navier-Stokes — take over. As a result, the animator only has control at the start of the animation while the rest of the animation is guaranteed to be 'too' realistic. Also, this approach is less suited for a stylised cartoon look, as the sharp outlines get smoothed out by the diffusion process.

For another feature film, Lamorlette and Foster published a system to control the behaviour of physically based fire [6]. They also argue that physics-based simulations are not well fit for an artistic environment, as such simulations are too slow and difficult to control. Hence, they developed a fire animation system with eight stages which is built up from single flames modelled on a natural diffusion flame. This tool is effective because many stages can either be directly controlled by an animator or driven by a physics-based mode. However, the focus of this tool is on just one particular gaseous phenomenon and involves a quite elaborate workflow.

Another type of animator controlled behaviour was described by Treuille et al., who started from user-specified key frames to direct smoke animations [7]. There, the animator specifies smoke density and velocity key frames while an optimisation process determines the appropriate wind forces needed to satisfy the constraints. In this approach, the animator has a lot of control; however, this is heavily related to the amount of key frames used. That is, between every two key frames a set of parameterised forces comes into play to produce a realistic simulation that best matches the animator's goals. Consequently, the only guarantee the animator has is that the animation will pass 'through' the defined key frames, but it is unpredictable between two consecutive key frames.

2.2 Physics-based Realistic Behaviour

Computational fluid dynamics can create convincing dynamic gaseous simulations. There is a consensus that the Navier-Stokes equations are an adequate model for fluid flow. This set of differential equations governs the motion of a fluid, expressing conservation of mass, linear momentum and energy for general motions.

Foster and Metaxas were among the first to produce good-looking results of animated gasses and fluids using the Navier-Stokes equations [8]. Jos Stam improved on their results and proposed an implicit, semi-Lagrangian technique for updating the grids, resulting in much simpler computations that were guaranteed to be stable [9]. These algorithms were extended by many researchers, for example by Fedkiw et al., optimising the techniques for simulating smoke [10].

Very recently, Selle et al. described a system to render smoke in a cartoon style [11]. They displayed particles generated by a dynamic fluid flow simulation (Fedkiw et al. [10]), using depth differences in the image buffer to calculate the position of cartoon outlines, analogous to Deussen en Strothotte's tree rendering algorithm [12]. They introduced rotated and stretched particles, which indicate the flow better and obtained good-looking

results of smoke colliding with a ceiling. However, concerning the rendering and compared to our approach, drawbacks are that aliasing artefacts can not be avoided (due to an aliased depth buffer), it is difficult to control varying outline thicknesses, and viewpoint changes loose frame-to-frame coherence.

Such physics-based approaches can produce very appealing and realistic results. However, from a stylised animation point of view, an important drawback is that the user has little influence on the behaviour of the animation. One may manipulate some initial parameters (such as viscosity, location, quantity, ...) but it is extremely hard to predict if and how the animation is altered by these changes. Moreover, the high computation and memory costs make these techniques less appropriate for an interactive trial-and-error approach of animators trying to engineer their best shot.

2.3 Simulated Realistic Behaviour

In contrast to taking resort to computational fluid dynamics, many approaches use simpler and computationally less expensive methods.

For instance, King et al. presented a technique to animate amorphous materials on graphics hardware with dedicated texture memory [13]. Object dynamics are achieved using procedural animation techniques while texture cycling is used to create local and global dynamics.

Many other approaches focus on just one particular gaseous phenomenon, often coupled with obliged movements. Dobashi et al. propose a simple, efficient method for animation of clouds [14]. Cloud evolution is simulated using a cellular automaton while the dynamics are expressed by several transition rules. Perlin and Neyret extended *Perlin Noise* so that shaders that make use of it can be animated over time to produce flow textures with a swirling quality [15]. Recently, Raghavachary and Benitez used several existing commercial packages to construct a painterly wall of fire for the animated feature film *'Spirit – Stallion of the Cimmaron'* [16].

Contrary to fluid dynamics, simple physics-based or heuristic approaches offer the user a certain degree of control and can be rendered quickly. However, often changes to the implementation have to be made in order to really control the animation or to increase the variety of stylised effects. Eventually, one ends up with a collection of autonomous dedicated systems. Moreover, none of the discussed approaches seems to be easily adaptable to generate qualitative stylised animation.

To summarise, physics-based algorithms and their parameters are very hard to control and hence can realise only a crude approximation of the fake behaviour the animator tries to achieve. Procedural approaches, on the other hand, deliver the look and feel of traditional animation, but at the expense of the animator's freedom as the fake behaviour is strongly bound by fixed procedures.

3 Our Approach

Considering traditional hand-drawn animation from an artist's standpoint; preserving shapes, getting perspective right and ensuring frame-to-frame coherence are major problems [17]. Existing software to assist traditional animation either lacks the 3D representation needed to tackle this kind of shortcomings, or imposes too many constraints hampering the animation artists' creativity.

The novelty of our approach lies in how we combine benefits from 2D and 3D techniques. Necessary 3D information — required for frame-to-frame coherence and a correct drawing order — is incorporated by having the animator draw time and view-dependent 2D animation paths and acceleration curves. On the other hand, we preserve the animator's artistic style by clearly distinguishing between a structured 2D modelling phase and a separate animation phase. This is similar to the 3D animation process and has been proven to be very useful for the purpose of creating convincing 3D-like animations starting from pure 2D drawings [18, 19].

3.1 Modelling and Animating in Structured 2D

This section briefly summarises our previous work in which we defined a structured 2D method for automatic in-betweening [18].

Considering 2D animation from a technical standpoint, two different categories can be distinguished: (i) transformations in a plane parallel to the drawing canvas (the XY plane), and (ii) transformations outside the drawing plane, especially all rotations around an axis different from the Z-axis.

The former category of transformations is relatively easy to deal with, whereas the latter is the main cause of all the trouble in automating the in-betweening process (i.e. the underlying subproblems of silhouette changes as well as self-occlusion). It is in the latter type of animation where the 3D structure comes into play that is underlying the objects and characters in traditional animation (and which is present in the animator's — and viewer's — mind), but which is not present in the 2D drawings.

To tackle this without introducing too much 3D information, we developed a solution based on structured 2D modelling and animation techniques. This is implemented as a multi-layered

system. At level 0, objects are modelled as sets of depth-ordered 2D drawing primitives (e.g. subdivision curves). Level 1 manages and processes explicit 2D modelling information and is fundamental in the realisation of transformations outside the drawing plane: for each set of 'important' x-y-rotations of the object relative to the virtual camera, the animator draws a set of ordered 2D primitives. This is functionally comparable to the extreme frames in traditional animation [17, 20]. Level 2 incorporates 3D information by means of 3D skeletons or approximate 3D objects, while level 3 offers the opportunity to include high-level tools (for example a deformation tool or a sketching tool).

Multi-level 2D strokes, interpolation techniques and on-the-fly resorting are used to create convincing 3D-like animations starting from pure 2D information. Unlike purely 3D-based approaches, the resulting animations still have many lively aspects akin to 2D animation.

This structured-2D approach (i.e. explicit 2D modelling and automatic in-betweening) is the starting-point of our current paper, where we investigate how to integrate turbulently moving 3D information, concentrating both on user control and on flicker-free stylised rendering.

3.2 Modelling Gaseous Phenomena

Incorporating 3D Information: Animation Flow. For many years, particle systems have been employed successfully to get flexible simulations/animations representing gaseous phenomena [21]. Particles or particle clusters are being moved through the animation space according to certain physics-based laws and directly rendered to the screen. Recently, Ilmonen and Kontkanen defined a *second order particle system* as an extension to the classical particle system in which, besides the visual particles, the particle sources and force generators are subject to the forces as well [22].

In our approach we employ a variant of a *second order particle system* to represent gaseous phenomena. The physics-based laws being responsible for the motion of the particles are replaced by time and view dependent drawn versions of the animation path (i.e. trails particles should follow) and drawn curves representing the particles' acceleration of speed. This way, the animator is able to create more aesthetically pleasing animations in an easy and rapid way.

For creating an animation path, the animator just draws one or more strokes indicating the flow of particles. In order to achieve convincing 3D-like animations, time and view-dependent versions of the path can be modelled. All these different versions can be regarded as extreme frames [17] and will be used by our in-betweening method in the animation phase. Figure 2 depicts an example of a time-dependent animation path drawn by a user. As one can see, animation paths can take any free-form shape the animator is bearing in mind. The boundaries of this fat free-form shape actually reflect the boundaries and movement of the trails that the visual components are allowed to follow.

Each 2D animation path gets 'inflated' to a 3D trajectory. We use an adaptation of the gesture-based sketching interface (TEDDY) presented by Igarashi et al. in which 3D polygonal objects are constructed by inflating the region surrounding the silhouette, making wide areas fat and narrow ones thin [23]. In our approach, we start with two drawn boundary paths in 2D. For each particle, first a path is chosen that is some (randomly generated) percentage halfway between these two drawn boundary paths. Then, this path is also translated in the direction perpendicular to the drawing plane, creating a 3D path. This perpendicular translation is given by another random value, in such a way that the volume containing all paths will form a 3D volume similar to Igarashi's volumes.

So, the trajectories the particles will follow are entirely defined by the animator in a familiar way. Via pure 2D input, necessary 3D information is generated, which helps preserve the overall shape and to ensure frame-to-frame coherence.

Figure 2: Examples of a time-dependent animation path drawn by the animator. a) Time t_0. b) Time t_1. c) Time t_2.

In addition, acceleration curves define how the particles are accelerated on their way along the animation path. This allows the animator better to control the animation and also permits all kinds of natural *and* 'unconventional' behaving animations which otherwise are too cumbersome or even impossible using existing approaches. Moreover, as the system gives instantaneous visual feedback, the user can 'polish up' the animation at runtime. The acceleration curves attributed to the animation paths of figure 2 are shown in figure 3.

Finally, some random 'deviation parameters' are assigned to each particle, including speed deviation, orientation deviation and mass' influence. These 'deviation parameters' prevent the particles' flow from looking artificial. As a result, particles can for example diverge from the trails bounded by the animation path.

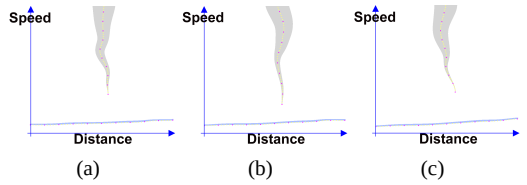

Figure 3: Examples of a time-dependent acceleration curve specified by the animator. These curves are attributed to the animation paths of figure 2. a) Time t_0. b) Time t_1. c) Time t_2.

Structured 2D Modelling Information: Visual Components. Instead of modelling the amorphous structure at once, we have the basic visual components (flames, drops, puffs, 'speedlines', ...) drawn by the animator independent of the animation path. In this way, the animator draws a stylised repository of the required basic gaseous components — including view-dependent versions — which does not depend on the underlying model and is highly reusable.

Figure 4 shows some extreme frames (i.e. combination of viewpoint and moment in time) of gaseous components. Notice that neither of the depicted elements is modelled true-to-nature. Moreover, depending on the extreme frame, the same component can change shape, colour, size, opaqueness, ...

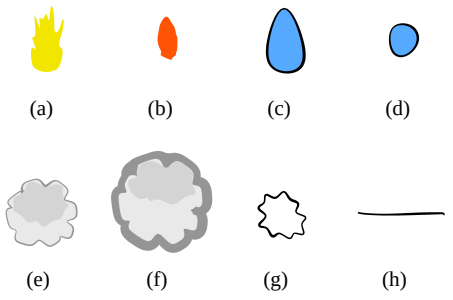

Figure 4: Some extreme frames of gaseous components. Note that for illustrative purposes all components are enlarged. a–b) The same flame at different ages. c–d) Drop of water as seen from different viewpoints. e–f) Cloud of smoke at different moments in time. g) Foam. h) Speed line indicating the animation flow.

3.3 Animating Gaseous Phenomena

In the previous section we introduced the incorporation of 3D information by means of modelled animation paths and acceleration curves, and the separate modelling of the basic visual gaseous components.

This section shows how gaseous phenomena can be animated in a stylised way using this 3D information and repository of gaseous components. An overview of the rendering process is given in listing 1.

```
Draw( frame f)
for each gaseous phenomenon, g_i, do
    inflate ap_{g_i}^{2D} to a 3D trajectory
    read key frame information
    generate in-between 3D trajectory, ap_{g_i}^{3D}
    generate in-between 2D acceleration curve, ac_{g_i}^{2D}
    Process Particles( particles p_{g_i}^{3D} )
end for
sort all particles p_{g_1..g_G}^{3D}
Draw Particles( p_{g_1..g_G}^{3D} )
```

Listing 1: Overview of the drawing process.

Consider an arbitrary gaseous phenomenon g_i at frame f.

As explained in previous section, before the animation, each 2D animation path first gets 'inflated' to a 3D trajectory.

During the animation, for each frame, we first parse the view-dependent and time-dependent key frame information specified by the animator. Typically, this information consists of a desired orientation, position and age of the animation path. This information then gets streamed to our in-betweening algorithm which creates an in-between 3D trajectory, $ap_{g_i}^{3D}$, and an in-between 2D acceleration curve, $ac_{g_i}^{2D}$.

Next, 3D particles, p^{3D}, are emitted into these trajectories (listing 2(top)). First, N new particles are emitted to replace the ones that 'finished' the animation. This is a user-controlled option that can be used to ensure a continuous fluent animation. Then, for each new particle p_n^{3D}, a random 2D component, p_n^{2D} is taken from the animator's repository. From now on, this structured 2D component is attributed to its 3D counterpart for its life span. After all new particles have been added to the current pool of particles, the pool gets updated. As mentioned in the previous section, the process of updating each particle p_j^{3D} basically resembles the conventional way [21] but in which user-defined animation paths and acceleration curves are used instead of physics-based or heuristic methods. Furthermore, we also store the depth, orientation and position for later use.

Once all particles have been processed for all animation paths, we need to determine the current drawing order of all visual components since we are working with 2D objects which contain no explicit depth information. This drawing order can directly be derived from the depth data that we stored for each p_j^{3D} in the processing step.

Next, for each particle, starting from the back to the front, an in-between version of the corresponding 2D gaseous component is calculated (listing 2(bottom)). Take for example particle p_j^{3D}. Given the co-ordinates of its position and orientation in world space, the absolute rotations in screen space around the vertical (Y) and horizontal (X) axis are calculated. Also, the current point in time is determined. This information (current viewpoint, moment in time) is then streamed to our in-betweening algorithm which creates an in-between 2D particle, p_j^{2D}, that actually reflects the animator's artistic style. After that, we put it into the right place in screen space. This requires only affine translations and Z-rotations. As a result, depending on the viewpoint and moment in time, each p_j^{2D} will change shape, position, orientation, colour, size, opaqueness, ... Finally, the particle/component is drawn on top of the others. As a result, we achieve an animation that does not suffer from temporal aliasing, while reflecting the user's style.

```
Process Particles( particles p^3D)
emit N new particles
for each new particle, p_n^3D, do
   assign random 2D component, p_n^2D
   add p_n^3D to p^3D
end for
for each particle, p_j^3D, do
   evolve
   store depth
   store orientation and position
end for
```

```
Draw Particles( particles p^3D )
for each particle, p_j^3D, do
   read orientation and position
   generate in-between 2D particle, p_j^2D
   orient and position p_j^2D in screen space
   draw p_j^2D on top
end for
```

Listing 2: Top) The processing procedure of particles. Bottom) The drawing process of particles.

Artists can configure a level of detail, depending on the particular effect they want to achieve for a specific animation. At the lowest level, the outlines of each component are shown while at the highest level outlines are shown only for the most prominent parts. A common outline is achieved by first drawing all the outlines of all the particles that are more or less on the same depth, and afterwards drawing the interior surface of these particles, effectively only erasing the outlines where particles overlap. Note that the particles have been partially depth-sorted to allow a correct treatment of transparencies. This sorting can be achieved in $O(n)$ time, because particles in a user-definable depth range are considered together.

4 Examples

We used the components (and variations of them) displayed in figure 4 to create a set of example animations. These examples contain gaseous phenomena (which are structured 2D models) as well as individual components which in turn are part of a higher-level structured 2D model (i.e. the scene).

Figure 5 shows some snapshots of a house with a smoking chimney, taken at different viewing angles. In this example we used components, depicting smoke puffs, which gradually (depending on their age) become larger and fade out. During the animation, the smoke puffs remain individual puffs or merge into one another (using blending operations) [24].

The images depicted in figure 6 depict a closer look at the barbecue party shown in figure 1.

Some images depicting a garden hose are shown in figure 7. We also included the possibility to let procedural approaches (for example, physics-based) take over (part of) the animation. This choice is entirely up to the animator and is fully configurable using the GUI of our program. In this example, the animator chose the bouncing effect of the water to be controlled by a physics-based procedure: once the particles bounce (i.e. arrive at the end of the trajectory) simple physics rules take over the animation. At the same time, the visual components depicting drops of water are replaced by foam. Note also that some particles have been marked as being 'speedlines': they're always drawn on top of the 'regular' particles. In this case they indicate the flow of the water.

Figure 8 illustrates the ease of use and large-scale applicability of our approach. It depicts the word 'CARTOON' catching various artistic kinds of fire: C = traditional fire, A = cartoon fire with thick outlines, R = smokey fire, T = bluish fire, first O = cartoon fire with thick outlines, second O = smokey fire, N = traditional fire.

Each example took an unskilled animator only few minutes to model the animation paths and gaseous components. All examples run at an interactive frame rate on a commodity personal computer (Pentium IV 3.06 GHz, RADEON 9000).

5 Discussion and Future Work

In this paper, we presented a novel approach to create stylised and believable animations of gaseous phenomena such as water, fire and smoke. We described how the animator draws view dependent and time dependent animation paths and acceleration curves which provide our system with the necessary 3D information to assure a frame-to-frame coherent animation. Furthermore, we showed how to retain the artist's freedom of creativity by exploiting structured 2D modelling and animation techniques: instead of modelling the amorphous

structure (or all the particles) at once, we have the basic visual components (flames, drops, puffs, 'speedlines', ...) drawn by the animator independent of the animation path.

Consequently, our approach shows how an animator can remain in charge of the entire animation process and demonstrates how one mechanism supports a complete range of effects. Moreover, the rendering part of our system also works with other simulation techniques (e.g., [9, 5]) as input.

In the future we want to explore the suitability of this technique to other cartoon effects such as water ripples, explosions, etc. The inflation process could also be improved by applying different densities for the boundaries and centre of the animation paths. Furthermore, we want to extend the framework to also support interaction between objects and effects, such as colour bleeding of fire on a wall.

Acknowledgements

We gratefully express our gratitude to the European Fund for Regional Development, the Flemish Government, and the European research project IST-2001-37116 'CUSTODIEV' which are kindly funding part of the research reported in this paper.

Furthermore, we would like to thank 'Xemi' Morales and Joan Cabot for their artistic advice.

References

[1] Patrick Witting. Computational fluid dynamics in a traditional animation environment. In *Proceedings of SIGGRAPH*, pages 129–136, 1999.

[2] Ronen Barzel. Faking dynamics of ropes and springs. *IEEE Computer Graphics and Applications*, 17:31–39, 1997.

[3] Jinhui Yu. A hierarchical flowing water model. In *Proceedings of the 7th National Conference on Imagery and Graphics*, 1994.

[4] Jinhui Yu and John W. Patterson. A fire model for 2D computer animation. In *Proceedings of Computer Animation and Simulation*, pages 49–60, 1996.

[5] Jinhui Yu. *Stylised Procedural Animation*. PhD thesis, University of Glasgow, 1999.

[6] Arnauld Lamorlette and Nick Foster. Structural modeling of flames for a production environment. In *Proceedings of SIGGRAPH*, pages 729–735, 2002.

[7] Adrien Treuille, Antoine McNamara, Zoran Popovic, and Jos Stam. Keyframe control of smoke simulations. In *Proceedings of SIGGRAPH*, pages 716–723, 2003.

[8] N. Foster and D. Metaxas. Modeling the motion of a hot, turbulent gas. In *Proceedings of SIGGRAPH*, pages 181–188, 1997.

[9] Jos Stam. Stable fluids. In *Proceedings of SIGGRAPH*, pages 121–128, 1999.

[10] Ron Fedkiw, Jos Stam, and Henrik Wan Jensen. Visual simulation of smoke. In *Proceedings of SIGGRAPH*, pages 23–30. ACM, 2001.

[11] Andrew Selle and Alex Mohr. Cartoon rendering of smoke animations. *NPAR2004: Symposium on Non-Photorealistic Animation and Rendering*, June 2004.

[12] Oliver Deussen and Thomas Strothotte. Computer-generated pen-and-ink illustration of trees. In *Proceedings of SIGGRAPH*, pages 13–18. ACM, 2000.

[13] Scott A. King, Roger A. Crawfis, and Wayland Reid. Fast animation of amorphous and gaseous phenomena. In *Proceedings of Volume Graphics (VG99)*, pages 333–346, March 1999.

[14] Yoshinori Dobashi, Kazufumi Kaneda, Hideo Yamashita, Tsuyoshi Okita, and Tomoyuki Nishita. A simple, efficient method for realistic animation of clouds. In *Proceedings of SIGGRAPH 2000*, pages 19–28, 2000.

[15] Ken Perlin and Fabrice Neyret. Flow noise. *SIGGRAPH. Technical Sketches and Applications.*, page 187, August 2001.

[16] S. Raghavachary and F. Benitez. Painterly fire. *SIGGRAPH. Technical Sketches and Applications.*, page 225, 2002.

[17] Preston Blair. *Cartoon Animation*. Walter Foster Publishing Inc., ISBN: 1–56010–084–2, 1994.

[18] Fabian Di Fiore, Philip Schaeken, Koen Elens, and Frank Van Reeth. Automatic in-betweening in computer assisted animation by exploiting 2.5D modelling techniques. In *Proceedings of Computer Animation (CA2001)*, pages 192–200, November 2001.

[19] Fabian Di Fiore, William Van Haevre, and Frank Van Reeth. Rendering artistic and believable trees for cartoon animation. In *Proceedings of Computer Graphics International (CGI2003)*, pages 144–151, July 2003.

[20] J. W. Patterson and P. J. Willis. Computer assisted animation: 2D or not 2D? *The Computer Journal*, 37(10):829–839, 1994.

[21] W. T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. In *Proceedings of SIGGRAPH*, pages 359–376, 1983.

[22] Tommi Ilmonen and Janne Kontkanen. The second order particle system. In *Journal of Winter School on Computer Graphics (WSCG2003)*, pages 240–246, 2003.

[23] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3D freeform design. In *Proceedings of SIGGRAPH*, pages 409–416. ACM, August 1999.

[24] Harold Whitaker and John Halas. *Timing for Animation*. Focal Press, ISBN: 0–240–51714–8, 1981.

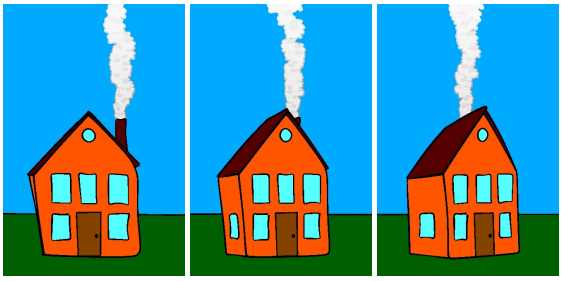

Figure 5: Snapshots of a house with a smoking chimney, taken at different viewing angles.

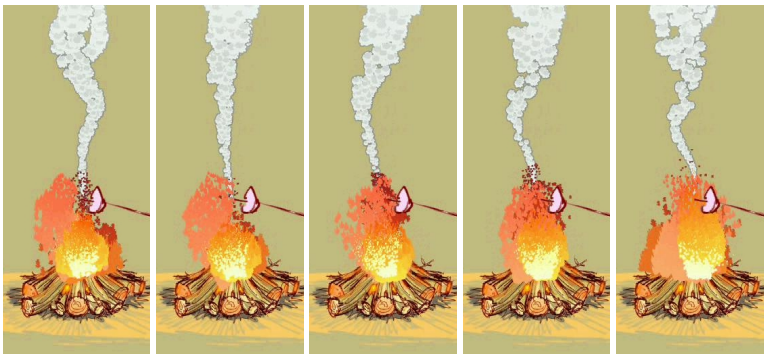

Figure 6: These pictures give a closer look at the animation depicted in figure 1.

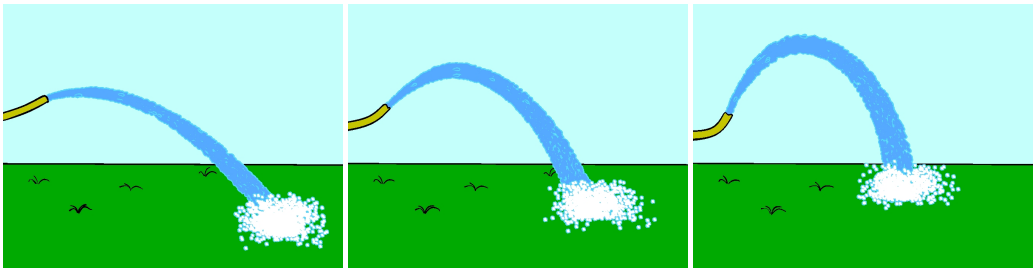

Figure 7: These pictures depict a stylised rendering of an animated garden hose. This example also demonstrates the use of 'speedlines' to indicate the flow of the water.

Figure 8: a) The word 'CARTOON' catching various artistic kinds of fire: C = traditional fire, A = cartoon fire with thick outlines, R = smokey fire, T = bluish fire, first O = cartoon fire (thick outlines), second O = smokey fire, N = traditional fire.