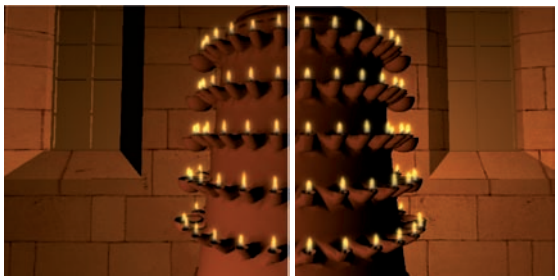
Eurographics Workshop on Natural Phenomena (2007)
D. Ebert, S. Mérillou (Editors)

Real-time Rendering and Animation of Plentiful Flames

Flavien Bridault, Michel Leblond, François Rousselle, Christophe Renaud

Laboratoire d'Informatique du Littoral
BP 719
62228 Calais cedex - France



(a) Without LOD (b) With LOD

Figure 1: *Lamp with 130 independent flames, up to 40Hz*

(a) Without LOD (b) With LOD

Figure 2: *12 torches, 10 visible, 5 flames per torch, 36Hz*

Abstract

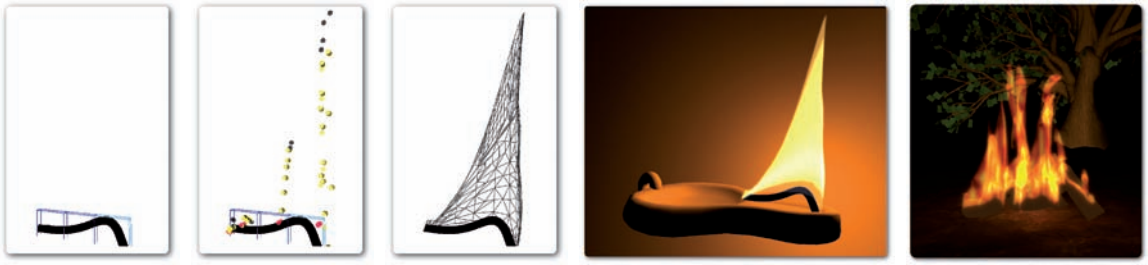
Rendering and animating flames in real time is a great challenge because of the complexity of the combustion process. While few models succeeded in simulating a single fire in real time, none tried to handle a large number of flames. In this paper we propose a set of techniques which aim at handling numerous flames like those of candles, torches or campfires in real time. We manage different levels of accuracy in the simulation, which is based on a fast fluid dynamics solver. We also consider many optimizations to reduce the rendering cost of a single flame. As a result, our approach is able to animate and render dozens of realistic flames in real-time, each one reacting independently to forces introduced at the scene level.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/image Generation - Display algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Animation

1. Introduction

Rendering and animation of flames is still a challenge for computer graphics. Lot of researches have been done in past years but modeling the underlying mechanisms of combustion remains complex and computationally demanding. Nevertheless flames are of great interest for entertainment industry, video games and virtual reality. According to these domains, the modeling and the simulation approaches greatly depends on the target application. Obviously off-line accurate simulations can provide realistic results but require high computation time. To the contrary, real-time applications require approximate models but render flame of poor quality.

First physically realistic flame models were proposed during the 90's. Simple laminar and static flames were described in [Ina90] and their dynamics in [Rac96]. Stam [SF95] used diffusion equations to simulate turbulent fires but the model parameters were difficult to adjust. Nguyen *et al* [NFJ02] used a set of two Euler's incompressible equations and a



(a) Wick split (b) Particles moving (c) Nurbs building (d) Final rendering (e) Campfire

Figure 3: *Build steps of NURBS based flames*

method known as the *level set equation* to reconstruct the shape of flames with implicit surfaces. The black-body approach is used in the rendering stage and the results look convincing. Lamorlette et al [LF02] proposed a model dedicated to production environments. This model allows the users to describe numerous types of fire (from a simple candle to a dragon breath) and offers some tools to make its use easier. The main drawback of all those methods lies in their high computational requirements due to the use of physics describing the combustion process.

Some other methods thus try to get rid of physics or make some approximations in order to obtain faster simulations. Particles systems [Ree83] were introduced to simulate fires and are nowadays widely used to describe fuzzy phenomena with a low cost. Beaudouin *et al* [BPP01] used particle chains evolving in a velocity field. The shape of flame is then reconstructed from these chains through the use of a potential equation and implicit surfaces. Pszczolkowska [Psz04] used a similar approach but add some Perlin's noise [Per85] to simulate the turbulence effects. Then she used a Gauss function rather than a potential equation. Nevertheless none of these works are able to provide real-time flames.

Some approaches bypass the modeling problem by using some real data captures. In that way Chalmers *et al* [DC01, Cha02] measured the flame luminance through the use of a spectroradiometer. The flame animation is recorded and then incorporated in the virtual environments. Its illumination is approximated with several emitting spheres. Hasinoff and Kutulakos [HK03] developed a method for reconstructing 3D flames from two 2D orthogonal views. More recently Ihrke and Magnor [IM04] have proposed a tomographic method for the reconstruction of volumetric flames from several pictures. All these methods provide high quality flame images but they lack interactivity due to the use of a finite set of captured data.

In the real-time rendering context, three approaches have been proposed. Wei *et al* [WLMK02] solved air flows with a GPU implementation of the *lattice Boltzman Model*. Particles are released in the velocity field and textured splats are used to render the flames. However their results appear more like burned gas rather than well shaped flames. Fuller *et al* [FKM*07] proposed to model fire through the use of a B-spline based volume. The fire can thus be easily deformed and manipulated by simple curves and volume operations. Realism is reached by adding improved Perlin noise to a fire texture and a hardware-assisted volumetric rendering allows fire to be displayed in real time. Results are impressive but rather concern large fires than simple and independent flames. We previously proposed to model flame shapes using NURBS surfaces [BLLRR06]. Control points and orientation of the surface are obtained by using particles chains. These ones flow into a velocity field computed by a fluid dynamics solver. This allows to get realistic simple flames in real time, each one being animated independently. However these three approaches fail in handling large number of flames in real time.

In this paper we thus address the issue of integrating a real-time flame model in any 3D engine for which lot of flames could be required. We propose a method which introduces levels of accuracy and optimizations to render and to animate simultaneously dozens of flames with the real time constraint. Our main contributions are:

- An adaptation of the grid resolution according to the pixel coverage of the flame on the screen,
- A switch from the fluid dynamics solver to a simplified velocity field when this coverage reaches a threshold,
- A smooth transition to avoid visual artifacts when swapping between the accuracy levels,
- The consideration of flame visibility to activate the solvers,
- A global management of external forces applied to flames.

These techniques allow us to improve the resolution process and thus to manage the animation of many independent flames. Furthermore we studied the problem of rendering efficiently dozens of NURBS. Carefully adapting the tessellation levels according to the pixel coverage of the flame and the viewer allows us to improve the rendering frame rate.

Let us note that this work is dedicated to flames like those of candles, torches or campfires. It is not our purpose to han-

dle larger fires. Furthermore, we voluntarily neglect the issue of the lighting. Indeed, finding a way to handle hundreds of lights in the rendering pipeline is a real challenge and is out of the scope of the current article.

In the next part we will recall the main features of the NURBS based flame model. Then part 3 will analyze the main problems in simulating high number of flames. Parts 4 and 5 will describe some techniques we have studied to reach our goal of handling dozens of flames. A way of controlling the animation of all the flames globally is described in part 6. Some results will be presented in part 7 and we will finally give some perspectives to this work in the last part of the paper.

(a) Texturing (b) Depth Peeling (c) Glow

Figure 4: *Blending process for multiple flames*

2. A real time flame model

Using virtual flames in a real-time context requires an underlying model which provides the following characteristics: realism, control and speed. A good level of realism is obtained by creating and animating a velocity field using an optimized version of Stam's Navier-Stokes incompressible equations solver [Sta99, BLLRR06].

An easy control is obtained by introducing the notion of virtual wick, a simple and efficient tool for generating, positioning and handling flames. A candle or an oil lamp use only one virtual wick (figure 3(d)) while more complex flames, like torch or campfire (figure 3(e)), use several ones. They can be visible or not, depending on whether the flames are produced by real wicks or not. Moreover, a user defined fuel distribution function (FDF) mapped on each wick allows us to introduce vertical forces simulating buoyancy in the velocity field. A periodical or a noise function is also added to these vertical forces to highlight the dynamic of the flame by producing a flickering effect.

Then, several points uniformly distributed along and around a wick are defined as origins of particles (figure 3(a)). At each origin point, a vertical force computed from the FDF is added, and particles chains (called here particles skeletons) are generated (figure 3(b)). This means that the shape of the flame is directly linked to, and control by the FDF. Speed of construction and rendering of the flame geometry are ensured by building a NURBS surface using existing graphics API. Particles of the chains are used as control points of the NURBS surface (figure 3(c)). Last, to improve realism, parts of some particles skeletons can randomly break away and form small detached flames.

Several wicks are combined to simulate a set of flames, however they remain independent in order to avoid the overload which would appear in trying to merge the flames. Instead, a visual merging is done by handling transparency in a post-process using texture mapping (figure 4(a)) and the Depth Peeling technique [Eve02] (figure 4(b)). Moreover, all the light sources are distinguishable from other objects by a glow that surrounds them. This effect, particularly important for flames perception, is created by a Gaussian blur applied on screen area surrounding flames. This also permits to improve blending of overlapping flames (figure 4(c)). For more details, see [BLLRR06, BLRRL06].

3. Performance analysis

In the case of real time rendering (about 30Hz), our experiments show that only 4 solvers can be handled per CPU using this approach. Actually two problems prevent to get better performances.

A grid of size $15 \times 15 \times 15$ is usually employed in order to solve the incompressible Navier-Stokes equations numerically. One solver is attached to each fire source. This produces realistic appearing flames and is sufficient for one candle, one torch or one campfire but it is quite expensive and it is actually the bottleneck of this model. Some optimization techniques could be applied to increase this number such as SIMD programming or a GPU implementation [Har04]. However, we don't believe that putting all computation on the GPU is the best solution in this case, especially because some GPU time is needed to render the flames. This cost is lower than the solver, but it can't be neglected. Thus we prefer to balance the load of the CPU and the GPU at the moment.

A more interesting thing to note is that the flames do not always need this level of precision to be *visually* realistic. When computing a frame, invisible flames do not need to be animated and rendered. Furthermore, if a hundred of flames are visible at the same time, most of them are necessarily of small size on the screen and do not require as much precision as the ones being close to the viewpoint. Thus we propose to manage level of details in flames animation and rendering. We carefully control the tessellation accuracy of the NURBS surface. We also introduce a method to adapt the level of precision in particles animation according to the size of the flame projected on the screen.

4. Level-of-details management

As we told previously, the main bottleneck of the model is the Navier-Stokes equations solver, even with the optimizations described in [BLLRR06]. However we quickly figured out that the same level of precision is not needed depending on the distance between the observer and the fire. Thus we first propose to adapt the resolution of the grid according to the pixel coverage of the flame and then to bypass the Navier-Stokes equations. We derive a simplified equation, that can be computed only for the particles instead of on the whole grid. After describing these two techniques, we will explain how we use them it practice.

4.1. Multi-Resolution

The quality of the flames shape depends on the resolution of the solver grid: The finer the grid, the smoother the shape is. But, in contrast, the slower the solver works. That is the reason why our first attempt to improve the speed of our algorithm consisted in adaptation of the grid resolution with respect to the pixel coverage: We reduce it when the coverage increases and vice versa.

We thought it is cheaper to solve a system with an iterative method (i.e. it takes fewer iterations) if the initial guess is close to the solution. To get such initial guesses when we changed the grid resolutions, we first tried the inter-grid transfer operators commonly used in multi-grid methods [Hac85, Bri87]. The restriction operator projects vectors on a grid to vectors on a coarser one by means of a 27-point full weighting scheme. Conversely, the prolongation operator transforms vectors on a grid to vectors on a finer one. A tri-linear interpolation is used. An adequate use of one of these operators to the solution returned by an actual solver gave us an initial guess for a solver with finer or coarser resolution.

These operators are available only if the format of the grid is $n_x \times n_y \times n_z$ where n_x, n_y, n_z are all numbers of the type $2^p - 1$. In our experiments we use $nx = ny = nz = 15$, 7 or 3. We swapped from any grid to a coarser one by dividing nx, ny and nz by 2. Conversely we swapped to a finer one by applying the mapping $x \rightarrow 2x + 1$ to these numbers.

In our experiments we used either null vector or vector obtained by means of the inter-grid transfer operators as initial guess. We did not noticed significant differences. The transfer operators did not slow down our algorithm. The use of a null vector did not alter the quality of the flame. Thus we had the choice of the method. We chose null vector as initial guess to get rid of the type of the number in the format of the grid resolution. We could then adopt a new scheme for changing the grid size. We switched to a finer grid by adding 2 voxels in each direction and to a coarser one by subtracting 2 voxels in each direction. As a result we now have more levels of approximation: The set of grid sizes we can now use is equal to $\{15, 13, 11, 9, 7, 5, 3\}$. Then we can now adapt our algorithm to the pixel coverage of the fire more precisely. However it's not judicious to use values less than 5 because of the poor resulting quality.

4.2. Simplified Resolution

When the camera is really far from the flame, this is a waste of time to compute the Navier-Stokes equations, even on a small grid, because no one can really distinguish the details. Thus the basic idea is to find a simplified scheme that can be used when the flame is far. However it should visually and roughly match the animation of the particles under the Navier-Stokes equations. As the flame model is based on particles, a big saving can be achieved by working directly with the particles instead of computing a velocity field on a grid. Thus at this point, we will consider a particle system. Each particle follows Newton's Second Law :

$$\vec{F} = m\vec{\gamma}. \tag{1}$$

where $\vec{F}$ is the resultant force, m the mass of a particle and $\vec{\gamma}$ the acceleration. Considering at the moment t the position $S(t)$ of a particle, $V(t)$ its velocity and $A(t)$ its acceleration we can write, using first-order approximation :

$$V(t) = \frac{dS(t)}{dt} \approx \frac{S(t+\Delta t) - S(t)}{\Delta t}, \tag{2}$$

$$A(t) = \frac{dV(t)}{dt} \approx \frac{V(t+\Delta t) - V(t)}{\Delta t}. \tag{3}$$

We can then rewrite equation (1) as :

$$F(t-\Delta t) = mA(t-\Delta t) = m\frac{V(t) - V(t-\Delta t)}{\Delta t}. \tag{4}$$

Combining equations (2) and (4) we derive the following recurrence relation between the positions of a particle at the moments $t-\Delta t$, t and $t+\Delta t$:

$$S(t+\Delta t) = 2S(t) - S(t-\Delta t) + \frac{\Delta t^2}{m}F, \tag{5}$$

with F the external forces defined as :

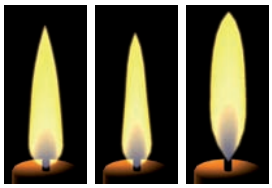
$$F = (fx, fy + c \times k(y), fz)^T, \tag{6}$$

$f_{x,y,z}$ denoting the wind, c a constant corresponding to the buoyancy, and $k(y)$ a coefficient varying from $[0;1]$ according to the normalized height in the velocity field.

In our experiments a flame placed far from the point of view, and obtained from this simplified resolution seems quite similar to a flame built from the solution of the Navier-Stokes equations. This is also the case for a far moving flame. However when coming nearer, such a flame looks more rigid. This is why the Navier-Stokes equations are still a better solution when the observer is close to the flame.

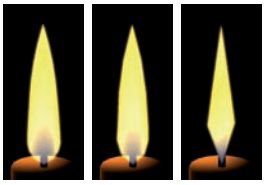
4.3. Application

Now that we have two different techniques to adapt the velocity field computation, we have to determine how to employ them practically. An observation force us to consider



(a) N=15 (b) N=7 (c) N=3

Figure 5: *Candle flame shape with different grid sizes N*



(a) None (b) Medium (c) Full

Figure 6: *Candle flame shape with different nurbs simplifications*

(a) (b) (c) (d) (e) (f)

Figure 7: *Candle flame behavior with the solver (a) (b), the simplified equation (c) (d), and then simplified equation combined with full nurbs simplifications (e) (f)*

two different cases, unique flames and multiple flames; actually we noticed that the variation of the grid size is not applicable for a unique flame like a candle flame. The velocity field is indeed quite stable whereas it is choppy for the bigger turbulent flames. Furthermore, the contour of the flame is clearly defined. Therefore its shape greatly suffers from the inaccuracy of low resolutions (see figure 5).

4.3.1. Unique flames

Thus in this case, the multi-resolution method can't be applied. So we chose only to switch between the Navier-Stokes solver and the simplified resolution. However as the simplified equation matches very roughly the field computed with the NS equations, the shape of the flame is not exactly the same, especially under the wind (see figure 7). To solve this problem, we propose to switch smoothly between the two vector fields. Over a given amount of time (typically 1 or 2 seconds), the position of a particle is computed with a linear interpolation taking the source field and the target field as inputs. The weight of the current field decreases whereas the one of the target field increases over the time. This can be formulated as :

$$p(t) = \frac{n * p_{src}(t) + (N-n) * p_{tgt}(t)}{N}, n \in [0;N] \quad (7)$$

where p_{src} is the position computed with the current vector field, p_{tgt} the position computed with the target vector field, and n a counter allowing to perform this kind of morphing.

4.3.2. Multiple flames

When using several NURBS flames to model a fire, the shape of any individual flame is hard to distinguish in the fire; notably because the velocity field used for multiple flames is very turbulent. Thus switching between different grid resolutions does affect the shape of the fire but the animation is so fast that it is hard to perceive the change. In addition, when the swap occurs, the camera is moving so at worst, it can be assimilated to an additional cause of turbulence. As a result, it's possible to use all the grid sizes we told earlier before switching to the simplified resolution. When the grid size can no more be decreased, we eventually switch to the simplified resolution. After many experiments we found the thresholds presented in Table 1, with the pixel coverage expressed as a percentage of the screen.

pixel coverage %	[100, 25[	[25, 5[	[5, 2.5[
resolution	15	13	11
pixel coverage %	[2.5, 1.5[	[1.5, 1[	[1, 0^+[
resolution	9	7	no solver

Table 1: *Resolution accuracy according to the pixel coverage*

5. Improving Performances

In this section, techniques used to optimize the rendering algorithm will be discussed. As it highly depends on the visibility and the pixel coverage of the flames, a good start is to explain how to compute them.

5.1. Visibility of the flames

The classical frustum culling can be used efficiently provided we define good bounding volumes for the flames. The bounding sphere approach has been preferred because the visibility test is cheapest. Despite its low cost, we decided to not recompute it on each frame. Indeed, we know that a flame can't go outside the grid of the solver. Thus in the most common case of a cubic grid, we can easily define a bounding sphere centered at the center of the solver with a radius of $\sqrt{c^2+\sqrt{c^2+c^2}}$ where c is the half length of a side of the grid. It can be computed only at the beginning and when the flame moves. In the same manner, the visibility test should only be updated when the flames or the camera moves.

This visibility test can be used to activate or deactivate the computing of a flame : solver, moving of the particles, and of course drawing. However, a special care should be taken when activating a flame. It should be done a short while before it becomes visible, otherwise the observer may see the flame starting moving. An easy way to achieve that is to enlarge slightly the radius of the bounding sphere by about 10%. Last the pixel coverage of the fire must be computed quickly. Thus we don't retrieve it accurately but we rather use the bounding sphere to quickly approximate it.

5.2. Reducing geometrical complexity

We previously saw that in our model, the shape of the flame is governed by a NURBS surface. We use the OpenGL GLU evaluators to tessellate the NURBS surface into a mesh. In order to get a fast rendering, it is crucial to keep the number of polygons as low as possible. And in the context of this paper, we obviously want to decrease the number of polygons when the camera move away from the flame.

Several sampling methods are available, and they give very different results. The default method GLU_PATH_LENGTH defines the maximum length in pixels of the polygons edges approximating the NURBS. It is view dependent but even with a very high length it doesn't succeed in getting a really coarse mesh. In fact, only the GLU_DOMAIN_DISTANCE (GDD) method, which sets the number of sample points per unit length in each direction u, v achieves that. We can use this method with one or two sample points when the flame is far, but it is preferable to use GLU_PARAMETRIC_ERROR (GPE) method when the flame is close; even with dozens of sample points, the flame is not as smooth as with this sampling method.

Reducing the number of polygons approximating the NURBS surface is important, however the cost of the sampling itself can not be neglected. We propose to reduce this cost by passing only a subset of the particles to the evaluator. When the flame is far enough, we only use one particle over two as control point. All the particles continue to evolve normally in the velocity field, ensuring the continuity of the shape of the flame.

Table 2 summarizes the parameters we determined after many experiments : pixel coverage thresholds, sampling methods, parameter value for the sampling method and numbers of particles considered to build a NURBS surface. Figures 6 and 7 show how these simplifications alter a candle flame.

pixel coverage %	[100, 5[	[5, 1.5[	[1.5, 0^+[
sampling method	GPE	GPE	GDD
parameter value	10	30	1
particles used	all	half	half

Table 2: *Summary of the NURBS sampling parameters according to the pixel coverage*

6. Global control of the animation

Far from here, we only talked about rendering and animation of dozens of flames. Nevertheless within a 3D engine, it is also important to get some control over the flames. While it is possible to add external forces to them individually, it can quickly become tedious. So we think that a more global control can be useful.

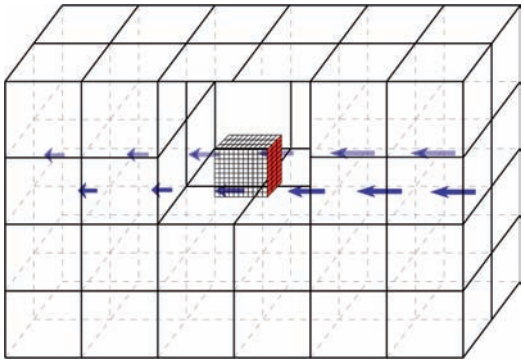


Figure 8: *Global solver*

In this purpose, we propose to use an extra fluid solver at the scene level. This fluid solver has also a coarse resolution to allow real-time performances. Each solver containing flames are located in this grid. The animator can then input forces in any voxel of the grid, for instance when a door is opening, or when a character is walking. This force will of course be convected in the grid thanks to the fluids computation. Each time the velocity in a voxel containing a local solver has a non-null value, forces are added on several of the six faces of the local solver, according to the directions of the components u, v, w of the velocity.

On the figure 8, a force is coming from the right, so $-x$. The global solver informs the local solver that a velocity is present. An average of the velocities of the voxels surrounding the local solver is computed, and thus here, forces will be added on the right face (in red on figure 8) in the same $-x$ direction. This process allows quick computations of the forces, and simplify the management of the wind when dozens of flames are present in a scene.

However we can imagine a more complete system. On the first hand for very large scenes, this approach can be combined with portals. Each room owns its global solver and when a room becomes visible, its solver is activated. The portal is used as an interface between the solvers to transmit forces. On the other hand, it would be really interesting to consider all objects in the scene and add boundary conditions in the global solver. But raising to such a level of realism would imply to increase drastically the resolution of the grid, which would prevent real-time performances.

Using an octree-based grid [LGF04] could allow to use a unique solver, with detailed zones around the flames and a rougher resolution elsewhere. But it would still be too expensive if we want to consider physical objects as boundary conditions. A better solution at this point would be to consider model reduction [TLP06], because this approach, despite the long pre-computation phase, can be computed in real-time.

7. Results

The benchmarks were performed on a Intel Core 2 Duo 6300 equipped with a NVIDIA GeForce 8800 GTX graphics card, and a screen resolution of 1024×768 pixels.

It is important to distinguish on one hand the cost of the computing of the fluid dynamics, either with Navier-Stokes or simplified equations, and on the other hand the cost of the flame display. Table 3 shows the number of iterations of the building process (fluids computing, particles moving and control points matrix building) that can be done in one second. Considering the minimum rate is 25Hz, we can see that with only solvers of $15 \times 15 \times 15$ voxels, we can animate at most 16 flames simultaneously. With the adaptation of the resolution method, we can build at most 256 flames in the best case.

resolution	15	13	11	9	7	0
iterations	410	610	940	1570	2300	6414

Table 3: *Fluid dynamics cost for various resolution sizes (resolution = 0 means simplified equations)*

However the display cost, including the NURBS sampling, is now more important than the fluids computing cost and prevents to reach this number. Figure 9 shows the variation of the frame rate according to the pixel coverage for a single NURBS flame with 28 control points, which is the average for a candle flame. Methods and settings presented in section 5 are tested individually, then together. They are activated according to the pixel coverage as shown on table 2. Thus under a coverage of 5% of the screen, the frame rate is obviously identical for all methods. Above this threshold, the gain is evident. Adapting the sampling method and reducing the number of control points give the best results.

Although we get at best a frame rate of 1300Hz for a single flame, this is obviously under the 6400Hz of the building process. It is interesting to note that now, the display cost is higher than the solvers cost. In other words, the GPU is more loaded than the CPU. Thus it confirms our assumption that computing the Navier-Stokes equations on the GPU would not increase the performances in this model.

Note also that the display cost is not linear. For instance the simulation on figure 1 shows an ancient lamp with 130 candles at a frame rate varying from 20Hz to 40Hz. Figure 2 shows 10 torches at 36Hz. It is important to note that torches are made of 5 NURBS flames, so the display cost is equivalent to 50 NURBS. Without all the optimizations we presented in this paper, this scene runs at 5Hz and the ancient lamp at 7Hz, and of course whatever the point of view. Videos are available on `http://lil.univ-littoral.fr/~bridault`. The model for the scene was obtained from `http://hdri.cgtechniques.com`.

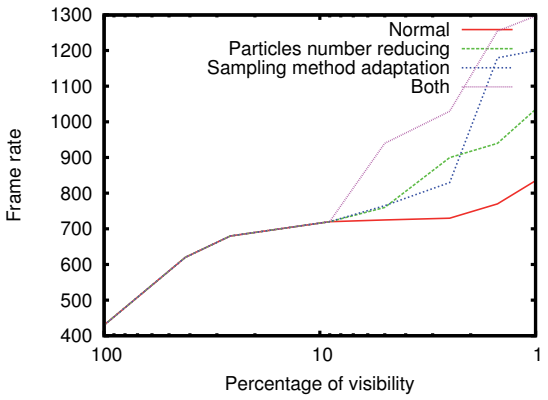


Figure 9: *Frame rate variation according to the NURBS sampling methods*

8. Conclusion

In this paper we have been interested in studying techniques allowing us to render and animate realistically and at real-time frequencies large number of flames. The use of a fluid dynamics solver for each flame was the main drawback to reach this goal, but it is still necessary for the purpose of a realistic animation. We thus proposed to reduce the solving cost of these solvers by decreasing their resolution according to the pixel coverage of the flame. When the observer is far enough a solver is even replaced by a simplified resolution and care has been taken to ensure visual coherency between successive levels of the resolution process. Techniques to improve rendering performances have been presented. Finally at the scene level an external forces solver is used in order to be able to approximate the forces propagation into the scene and thus to improve the realism of the animation. The combination of all these works allows us to render and to animate in real-time several dozens of visually realistic flames.

Several new improvements to our work are currently under investigation. The first one is to study ways to allow all the flames to illuminate the environment. We previously proposed to approximate the illumination due to a flame through the use of a light shader implementing a photometric solid. But this approach is restricted to a small number of flames due to the high cost of each shader. One way of bypassing this problem for large number of flames could be to use some clustering techniques in order to reduce the number of shaders that have to be used. Another problem lies in handling shadows. But even if shadows are a consequence of illumination, their simulation for a large number of sources in real-time is nowadays a open problem and it requires deep investigations.

Up to now our approach does not take into account interactions between the flames and any object of the environ-

ment. This would imply to take into account boundary conditions in the solver. Obviously the inaccuracy of the simulations increases while the solvers resolution decreases and may then lead to visually unpleasant effects. These effects would probably be emphasized when using the simplified resolution only and collision detection techniques could conceivably be helpful in this case.

References

[BLLRR06] BRIDAULT-LOUCHEZ F., LEBLOND M., ROUSSELLE F., RENAUD C.: Enhanced illumination of reconstructed dynamic environments using a real-time flame model. In *Afrigraph '06: Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa* (New York, NY, USA, january 2006), ACM Press, pp. 31–40.

[BLRRL06] BRIDAULT-LOUCHEZ F., ROUSSELLE F., RENAUD C., LEBLOND M.: Real-time animation of various flame shapes. In *VAST 2006: Proceedings of the 7th International Symposium on Virtual Reality, Archeology and Cultural Heritage* (october 2006).

[BPP01] BEAUDOIN P., PAQUET S., POULIN P.: Realistic and controllable fire simulation. In *Proceedings of Graphics Interface 2001* (2001), Watson B., Buchanan J. W., (Eds.), pp. 159–166.

[Bri87] BRIGGS W. L.: *A Multigrid Tutorial*. SIAM Books, 1987.

[Cha02] CHALMERS A.: Very realistic graphics for visualising archaeological site reconstructions. In *Proceedings of the 18th spring conference on Computer graphics* (Avril 2002), ACM Press, pp. 43–48.

[DC01] DEVLIN K., CHALMERS A.: Realistic visualisation of the pompeii frescoes. In *Proceedings of the 1st international conference on Computer graphics, virtual reality and visualisation* (2001), ACM Press, pp. 43–48.

[Eve02] EVERITT C.: Interactive order-independent transparency. tech. rep., 2002.

[FKM*07] FULLER A. R., KRISHNAN H., MAHROUS K., HAMANN B., JOY K. I.: Real-time procedural volumetric fire. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2007), ACM Press, pp. 175–180.

[Hac85] HACKBUSCH W.: *Multi-grid methods and applications.* Springer Verlag, 1985.

[Har04] HARRIS M. J.: Fast fluids dynamics on the gpu. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics* (2004), 343–361.

[HK03] HASINOFF S. W., KUTULAKOS K. N.: Photo-consistent 3d fire by flame-sheet decomposition. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision* (Washington, DC, USA, 2003), IEEE Computer Society, p. 1184.

[IM04] IHRKE I., MAGNOR M.: Image-based tomographic reconstruction of flames. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2004), ACM Press, pp. 365–373.

[Ina90] INAKAGE M.: A simple model of flames. In *Proceedings of the eighth international conference of the Computer Graphics Society on CG International '90: computer graphics around the world* (Institute of Systems Science, Singapore, july 1990), pp. 71–81.

[LF02] LAMORLETTE A., FOSTER N.: Structural modeling of flames for a production environment. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 729–735.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM Press, pp. 457–462.

[NFJ02] NGUYEN D. Q., FEDKIW R., JENSEN H. W.: Physically based modeling and animation of fire. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 721–728.

[Per85] PERLIN K.: An image synthesizer. *SIGGRAPH Comput. Graph. 19*, 3 (1985), 287–296.

[Psz04] PSZCZOLKOWSKA D.: Visual model of fire. In *Eurographics 2004 Short Presentations* (Warsaw University of Technology Poland, 2004), 2004 E. A., (Ed.).

[Rac96] RACZKOWSKI J.: Visual simulation and animation of a laminar candle flame. In *International Conference On Image Processing and Computer Graphics* (1996).

[Ree83] REEVES W. T.: Particle systems : a technique for modeling a class of fuzzy objects. *ACM Trans. Graph. 2*, 2 (1983), 91–108.

[SF95] STAM J., FIUME E.: Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM Press, pp. 129–136.

[Sta99] STAM J.: Stable fluids. In *Siggraph 1999, Computer Graphics Proceedings* (Los Angeles, 1999), Rockwood A., (Ed.), Addison Wesley Longman, pp. 121–128.

[TLP06] TREUILLE A., LEWIS A., POPOVIĆ Z.: Model reduction for real-time fluids. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 826–834.

[WLMK02] WEI X., LI W., MUELLER K., KAUFMAN A.: Simulating fire with texture splats, 2002.