Animation of Water Droplets and their Dynamics on Glass Panes

Seema Jogoo, Msc Computer Animation, NCCA

5th September 2005

Contents

List of Algorithms

List of Figures

List of Tables

A model for simulating the flow of liquid droplets on a flat surface is presented. The patterns created by droplets rolling down a surface are very complicated and difficult to model accurately. The method that is proposed uses the underlying surface properties and interfacial dynamics to control the motion of the droplets. A discrete method is employed as a base layer for determining the flow and a texture map layered on top provides added accuracy and detail. Merging and splitting behaviors are simulated by relating aspects of the flow, for example, position, to a shader model. The rendering process includes a texture mapping technique that allows droplets to be located rapidly, and a simple refraction method defined in the shader. Refraction is based on attibutes distributed across the surface and on the positions of each droplet.

1 Introduction

As water droplets flow along a surface, they leave a trail of water behind, marking their path. The unpredictability of the shape of these paths, that is, of what route a droplet will take is the most intriguing aspect visually and a main reason for the development of such a simulation. Each run would hence generate a different pattern, depending on the state of the surface at that moment in time. For example, if the surface contains a static droplet of water at some position, p, then it is most likely that another droplet arriving at p would interact with the static one rather than continuing down its path. But it is hard to tell, since there are various other properties that influence the behavior of a droplet. By addressing some of these factors and simulating them over time, it is possible to retrieve the key features of a flow such that a realistic and believable motion is obtained.

A lot of methods for modelling and rendering water elements have been developed over the past years. When water first started getting discovered in computer graphics, only large bodies without boundary, for example the ocean, were focused on. From then on, refinements were made by adding features such as a boundary representing a seashore. Fluid dynamics were taken to higher levels by considering liquid in containers, where improvements to classic Navier-Stokes Equations achieved realistic wave behaviors and rotational effects together (referring to the paper presented by N.Foster and D.Metaxas, [19]).

The modelling of small scale water entities, for example, droplets, involves more complex natures. Efforts were being made into understanding the phenomena between liquid and solid and the dynamics of water droplets[1]. It was only in 1993 that the subject of water droplets, specifically, was approached by K.Kaneda, T.Kagawa and H.Yamashita[2]. The solution presented in their paper defines the flow of droplets based on the underlying surface properties, which is a logical relation to make. Consider a droplet stuck to a surface. At some point in time, the droplet starts moving down. If the surface is dry or dirty, the droplet would move slowly and eventually deccelerate to a stop. On a slick surface, the droplet flows smoothly, and the mass of water they leave behind beads up into tiny droplets. It does not run straight down but rather meanders along, making decisions on which way to go. If another droplet is close by, the droplet would deviate and merge with the it. If impurities are added to the surface, the droplet's motion becomes more jerky.

The motion of water droplets over a surface is affected by countless factors. Trying to take all of these different elements into account at the same time so as to obtain an exact physical animation is almost impossible, and would require an accuracy that goes far beyond the scope of this project. Instead, a method is developed that generates the animation of water droplets down a glass pane, based on gravity and a set of surface parameters defining interfacial tensions, friction due surface roughness affinity for water and a wetness factor. Merging is achieved at the rendering stage, where a shader model is designed to provide a believable shape to the droplets. A realistic wet look is achieved by including refraction based on the amount of water present on the surface. To complete an accurate simulation, droplet trails are generated which depend on the splitting of droplets.

The inclusion of water droplets on glass surfaces in a scene is important for uses in drive simulators as it creates rainy conditions, something that every driver has to deal with. Water droplets is increasingly being used as an element to visual effects. For example, the recent film 'Sin City' depicts a shower of rain and droplets flowing along the side of car windows. 'Road to Perdition' creates the effect of water streaming down a window, reflecting over a wall. Thus, the project aims towards creating a realistic effect of water droplets flowing down a glass surface.

The remainder of the report is structured as follows:

Section 2 looks at previous works that relate to the project. A background of the terms and techniques leading to the solutions achieved by the project is given.

Section 3 outlines an overview of the project.

Section 4-11 describe the models derived for the components of the system that relate the flow, shape and appearance of the droplets.

Section 12 discusses the overall pipeline of the system.

Sections 13,14 and 15 build on the design of the system.

Section 14 describes the algorithms implmented in the system.

Section 16 lists the key steps in the system's implementation.

Section 17 addresses the rendering technique employed.

Sections 18-21 discusses the technical problems encouters, the solutions derived and optimization steps taken throughout the project.

Section 22 gives some animated sequences illustrating the effects produced with the system.

The rest of the report evaluates and concludes the project, discussing future works to be done on the project.

2 Background

2.1 Definitions

Below are some definitions and terms used throughout the report.

The conservation of momentum states that the amount of momentum remains constant throughout a problem domain. Momentum is neither created nor destroyed, but only changed through the action of forces as described by Newton's laws of motion.

Term	*Definition*
Contact Angle θ	Formed when a liquid does not completely spread on a substrate(usually solid).
Interfacial tensions γ	A measure of the excess energy present at an interface arising from the imbalance of forces between molecules at an interface e.g at liquid/gas or liquid/solid.
Spreading coefficient S	The energy difference between the solid and the contacting gas/liquid phases.
Thin film flow	An expanse of liquid partially bounded by a solid substrate with a free surface where the liquid is exposed to another fluid.
Reynold's number Re	The ratio of inertial forces to viscous(heavy and gluey) forces. That is, Re determines the stability of flow.
Index of refraction, n	The index of refraction of a material is the ratio of the speed of light in vacuum to the speed of light in that material.

2.2 Previous Works

1. J.Dorsey, H.K.Pedersen and P.Hanrahan [10] presented a model for the changes in appearance caused by the flow of water over complex surfaces. The model produces convincing results of weathering effects, taking into account chemical interactions due to evaporation, staining, deposition and different materials. The flow is modelled as a particle system under equations of motion, where each particle is constrained to the surface by projecting the resulting force vector onto the tangent plane of the surface. Several maps are placed onto the surface geometry and are used to retrieve values needed for rate equations governing, for example, the absorption of water on the surface.

2. Y.Yang, C.Zhu and H.Zhang[14], proposed a model that applies a partial-distortion method to approximate water droplets and their visual effects on glass. A droplet acts as an optical lens, and distorts the area it covers using a nonlinear deformation approach. As a result, the simulation can be achieved in real-time. Instead of modelling 3D shapes, the rendering pipeline takes all 3D scene objects, projects, clips and rasterizes into the framebuffer. The 2D images are then distorted where the droplet is. Simulating different water thicknesses is possible by altering a distortion coefficient. The tail left by a droplet is created by distorting an irregular narrow area.

3. K.Kaneda, T.Kagawa and H.Yamashita [2] presented a method for the animation of water droplets and their streams on a glass plate(for example,a glass pane), taking

into account interfacial dynamics. A discrete model, in which the surface is divided into equal meshes, is used to simulate the meadering behavior of a droplet. The system also considers the reflection and refraction of light through the glass and the droplet. An efficient rendering approach is employed where pixel colours are determined by the intersection of a ray and a cuboid onto which scene objects are projected. Gravity is not integrated as an external force, but rather simulated by a higher priority move to the position (i,j+1) on the grid.

4. P.Fournier,in[4], proposed a model for the flow and shape of liquid droplets. A neighbourhood graph is constructed for the traversal of the drops down the surface meshes. The aim is to simulate their visual contour and shape when influenced by internal and external forces. Collisions between droplets are considered, and by assuming that the event is instantaneous, the complexity is greatly reduced. Streaks left behind are represented by narrow rectangles as a droplet rolls over each triangular mesh. A point-mass model is used as a dynamic modeller for a droplet. The droplet structure consists of point-masses connected with springs, governed by the classic equations $F = ma$ and $F = (k(l_0 - l) - z\frac{dl}{dt})u$. By maintaining four defined constraints true, realistic droplet shapes are obtained and no further dynamics are considered.

5. M.Jonson and A.Hast [6] introduced a method to animate the flow of water droplets on a structured surface, and consists of texturing and bump mapping a flat surface. The bump normal is used to control the motion of the droplets. Their model includes trails produced by changing the background texture on the surface and runs in real-time. They apply the Gram Schmidt orthogonalization algorithm to describe the motion of the droplets.

6. G.Miller an d A.Pearce [12] use the idea behind flocking systems to model viscous fluid flow. A connected particle system represents the flocklike behaviors as described in [13], and making it dynamic allows for particle-particle interactions and interactions with the local environment. Soft collisions are intergrated in order for particles to flow over one another, by having factors such as repulsion and distance decrease gradually from a particle's centre. For rendering, the model uses smooth shaded spheres to produce an approximation of an isosurface which covers the particles, thus avoiding the expense of actually computing an isosurface.

7. Y.Yu, H.Jung and H.Cho,[15] achieved photorealistic looking droplets by using metaballs and their deformations under gravity and friction. They introduce a new vector field iso surface function controlling the basic scalar model with respect to the norm of gravity. For rendering the droplets, ray tracing techniques are used, which gives natural shadows.

2.3 Dynamics of water droplets

2.3.1 Interfacial tensions γ

Def:2.1

Following from its definition, γcan be quantified as the force acting normal to the interface per unit length (that is, force/unit length). This force tends to minimise the area of the surface, explaining why droplets are round.

2.3.2 Contact angle θ

Def:2.1

The contact angle gives a measure of the wetting of a solid by a liquid.[3]

It is determined by the tangent at the contact point where liquid and solid intersect.

2.3.3 Dynamic contact angles (DCA)

As a droplet flows, its contact line is in controlled motion and its dynamic contact angles (DCA) need to be considered. DCA consist of the advancing contact angle θa and the receding contact angle θr.

Note: there are 2 cases

(i) the advanced/receded contact angles, for example, a drop on an inclined plate.

(ii) the advancing/receding contact angles, for example, a moving drop on an inclined plate.

For the purpose of the model, we are concerned mainly with (ii) and its effects on a droplet

Contact angle hysteresis

When $\theta a \neq \theta r$, the system is said to exhibit contact angle hysteresis (H). H is calculated by subtracting the maximum advancing contact angle with the minimum receding contact angle. That is,

H=$\theta a - \theta r$

In the case of droplets flowing down a surface, the hysteresis arised mainly due to surface roughness, heterogenity of the surface and the impurities deposited onto it.

2.3.4 Contact line

The contact line refers to the boundary line around a droplet where all three phases(solid, liquid, gas) are in contact. A sliding droplet then refers to the movement of the contact line.

2.4 Flow dynamics

There are various factors that affect the sliding behavior of a droplet down a solid surface. From observations, a stream can meander precariously giving the impression of turbulent flow or calmly following a defined path. Briefly discussed below are some different flow patterns that might be exhibited during the droplet's journey. The project is concerned mainly with the patterns that may arise depending on elements in the surrounding environment, for example wind.

Reynold's Number, Re

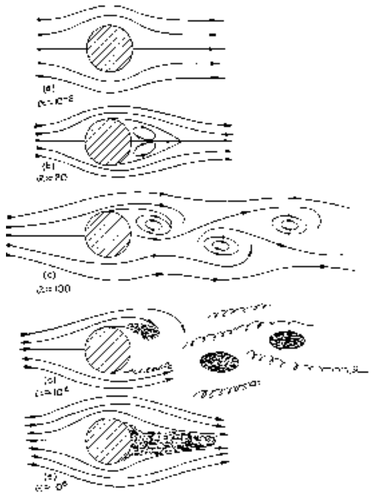
Def:2.1

The Re of a flow depends on the velocity, density and viscosity of the fluid. It can be calculated by

Re= $\nu D\rho/\mu$

where υ=mean velocity, D= characteristic length, ρ=density of fluid and =viscosity of fluid.

Critical Re:


a. The development of turbulence from laminar flow.

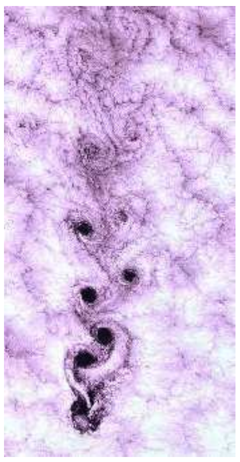

b. Von-karman vortex street

Figure 1: Reaching turbulent flow

The critical Re defines when laminar flow is disrupted and turbulence occurs.

Stable flow -> critical Re (point of instability) -> transition Re (point of transition) -> turbulent flow

This illustrates the gradual increase in turbulence with the increase in velocity. During this period, several flow patterns are observed, depending on the fluid's environment.

2.4.1 Laminar flow

Laminar flow is also referred to as streamline or viscous flow. In this case, layers of water flow over one another at different speeds, with no mixing among them. Instead, the fluid moves in definite paths. Note: the viscosity of the fluid plays a significant role in this case.

A small Re defines laminar flow, where velocity is constant at any point. (Re<1), as described in [**?**]

Figure(1a.) illustrates the various flow patterns that are exhibited during the transition to turbulent flow.

2.4.2 Turbulent flow

Turbulent flow is characterized by the irregular movement of particles of the fluid. The particles travel in irregular paths with no observable patterns and no definite layers.

Vortices start to develop when $Re > 4$. Von Karman street vortices become apparent around an $Re > 40$. The velocity at any point varies with time in a regular cyclic fashion, that is, they have a periodic nature, as shown below. In this case, the fluid elements possess both linear and angular momentum.

The shape of the droplet paths have a similar resemblence to the von karman vortex street pattern, depending on the surface properties. Figure 1b. gives an example of a von-karman vortex street turbulence.

When $Re > 400$, von karman street vortices disappear and turbulent flow occurs.

2.4.3 Thin film flows

Def:2.1

In this particular project, thin film flow is described as the flow of each droplet down the surface, under the action of gravity and friction. The most correct way to model this phenomenon is via the Navier-Stokes equations which require a lot of computation and presents a separate project on its own. In our case, what is of most concern is the effects of surface tensions and velocity on the film flow and its subsequent appearance.

Even when small, surface tension has a smoothing effect on the flow. As mentioned in [8], the film shape cannot be determined without first knowing the configuration of the various factors involved. The flow takes place in the direction of the longest dimension, the length L, as opposed to the thickness H, under the action of an external force such as gravity.

The film thickness H, depends on the viscosity and density of the liquid by [7] :

$\delta=[\frac{3\mu Q}{W\rho g sin\theta}]^{1/3}$

where μ=dynamic viscosity, Q=volumetric flowrate, W=work, ρ=density and θ=angle of plane inclination

Also, the film thickness is related to the average velocity:

$\delta= \left[\frac{3\mu\nu}{\rho g sin\theta}\right]^{1/2}$

note: this equation holds for a Reynold's number<500.

The flow velocity in the direction perpendicular to the surface is much smaller than the velocity of the main flow along the surface.

From observations, the film flow of water droplets viewed from behind a glass plate blurs and distorts the view beyond and will be considered in rendering.

2.5 Surface properties

The flow of droplets on a surface is influenced by the properties of the surface such as its roughness, impurities deposited by previous droplets. This section describes the characteristics that are most relevant and might influence aspects of the model.

2.5.1 Surface roughness

The surface roughness, kr, may be taken as a function of the size of the bumps on the surface relative to the size of the droplet [4],

kr=hb/hd if hb<hd

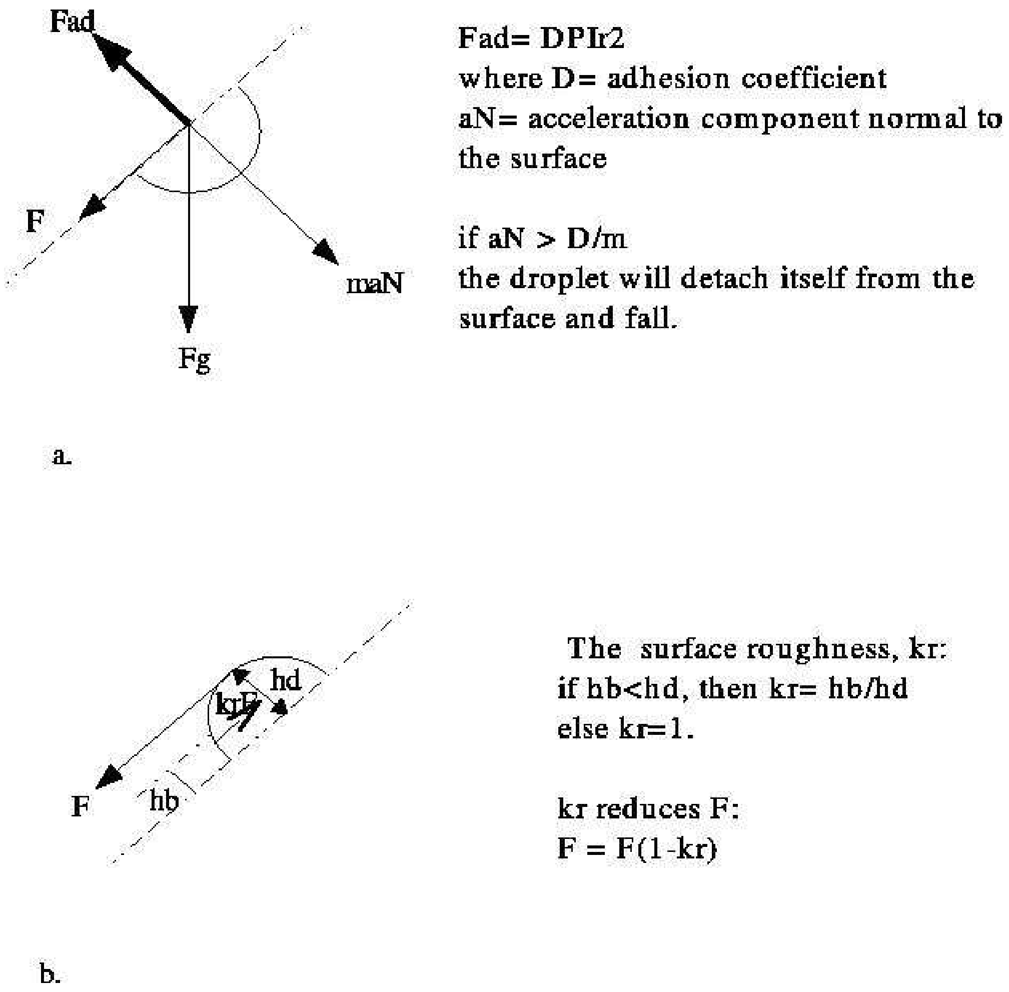
kr=1 otherwise

where hd is the height of the droplet and hb is the height of the bump.

In the case of a sliding droplet, we assume that the surface roughness will only reduce the tangential force on the droplet and is given by

$Ft \leftarrow Ft(1 - kr)$

Figure 2b. demonstrates the frictional force due to surface roughness on a droplet.



Fad= DPIr2
where D= adhesion coefficient
aN= acceleration component normal to the surface

if aN > D/m
the droplet will detach itself from the surface and fall.

The surface roughness, kr:
if hb<hd, then kr= hb/hd
else kr=1.

kr reduces F:
F = F(1-kr)

Figure 2: Surface roughness and adhesion to a surface

2.5.2 Affinity for water

The affinity for water factor is used to characterize the tiny impurities of solids present on the surface. The term affinity generally may be used to describe the hydrophobicity or hydrophilicity of a surface. It is dependent on the surface material and affects the flow of a droplet on the surface. Slick surfaces have low affinity that are almost constant over the surface, while dull rough surfaces have large jumps and higher affinities

2.5.3 Adhesion

A droplet flows on a surface because it adheres to it. Adhesion is known as the force F_{ad} along the surface normal[4]. It is a function of the contact area between the liquid and the solid surface, and may be intepreted by the adhesion coefficient D$\triangle$ between these two states. If F_{ad} is smaller than the acceleration component that is normal to the surface, then the droplet will be able to detach itself from the solid and fall. Refer to Figure 2a.

As described in the article in [5], adhesion of a downward moving droplet causes its leading edge to be pinned down due to impurities. The contact angle at this edge is therefore larger than that at the tail(please refer to contact angle hyteresis2.3.3).

The above explanation accounts mostly for the adhesion of water to the surface and would influence the flow of the droplet. Also to be considered, are the properties of water itself [**?**] . Water has a high surface tension, that is, a high ability to stick to itself, allowing it to stretch and deform under external influences. If free from a surface, a droplet would form a sphere and resists shape change.

2.5.4 Contact Area and weight

The larger the contact angle, the smaller the contact area. A rough surface has a larger surface area and also traps air. This would cause a droplet to sit partly on air, exhibiting poor wetting. As mentioned in [5], by taking fakirs as an example, he is normally able to rest on his bed of needles, but if the weight of the fakir is increased considerably, he would get impaled on the posts, therefore demonstrating that the weight of a droplet would also affect how much it would spread out. Hence, the contact area may be considered as a quantity to determine the wetting phenomenon. Please refer to 9.1.

2.6 Wetting

Wetting is the process when a liquid spreads on a solid substrate[3]. It can be considered as the amount of spread of the liquid over the surface. A measure of wetting would be the contact angle between the liquid and solid, where the smaller the contact angle is, the better the wetting. This phenomenon affects the shape of the droplet and is discussed more in section9 . In the case of a dynamic droplet, wetting also influences its sliding behavior, based on several factors.

Wetting is also responsible for the appearance of a droplet's stream, as explained in [1].

2.7 Numerical Solver

For every physical simulation, its numerical solver is set the task of intergrating the equations of motion set for the system. Out of the various methods for intergration, the numerical intergration is often chosen as it is the simplest to implement and performs well.

Ordinary Differential Equations (ODE's)

Intergrating Ordinary Differential Equations results in a numerical approximation of the derivative, $f'(x)$, to a function $f(x)$. The first order derivative of a variable is the simplesst ODE and often arise in *initial value problems, in which the* initial conditions of the system are known and time moves forward.

(from Mathsworld)

Numerical integration is the approximate computation of an integral using numerical techniques.

2.7.1 Euler Method

The Euler method, given in equation 1, is the simplest method employed to solve ODE's. It uses the initial value technique to feed on the information at the beginning of each time step so as to get to the end of the step.

$y_{n+1} = y_n + h.f'(x_n, y_n)$ eq.1

For example, Newton's equations of motion are solved by numerically finding the velocity,($f(x)$), over time from the acceleration*(f '(x)).* Consequently, knowing the velocity, it in turn becomes the derivative*(f '(x)), the distance,(*$f(x)$), may be numerically calculated.

2.8 Particle Systems

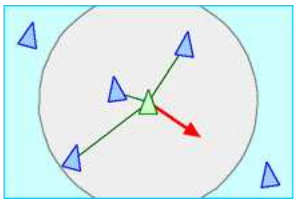
A particle system, as defined in the original paper by W.T.Reeves,[11], is a collection of many minute paticles that together represent a fuzzy object. Particle systems are popular in computer graphics and have been used to describe a vast number of phenomena, for example, galaxies, fire. They are easily programmed and techniques to describe their motion can be found in well documented literatures.

Such a system provided an effective way for representing objects with behaviors that evolve over time. Each particle carries a set of sttributes,for example, position, velocity and lifetime. The particle system determines initial values for each of these attributes, which then get updated throughout the course of their lifetime. For example, the particle's lifetime gets decremented on every frame until it is 0, which kills it off. The generation shape defines the region where new born particles are placed and can range from the simplest sphere to more complex shapes based on the laws of nature. Particles may be managed on a per-particle basis, which allows a greater precision to be modelled.

2.8.1 Flocking System

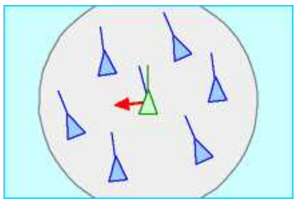
Craig Reynold's paper on Flocking[13] describes a flocking system as an elaboration of a particle system. More precisely, it can be regarded as an intelligent particle system. With an appropriate set of rules that defines the behaviors of an individual member, the latter is given a means to interact with its flockmates. Each behavior influences the motion of the particle, causing it to react accordingly. The paper describes 3 basic rules, each expressing a manner in which the flock member should behave individually according to their position and that of its flockmates:

Rule1: Collision avoidance



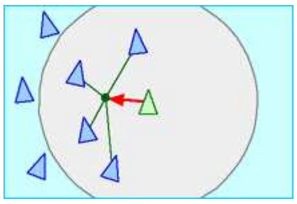
Collision avoidance ensures that boids in a flock allow some spacing between each other. It is usually represented as a repulsive force from the boid to its neighbours. Without this spacing , boids could overlap and collide, disrupting the appearance and meaning of a flock.In Reynold's paper, static collision avoidance is taken into account, which is based on the position of a boid's neighbours, ignoring velocity.

Rule2: Velocity matching



Velocity matching seeks to match the particle's velocity to that of its neighbours. It is a predictive version of collision avoidance, that is, if the particle manages to keep the same velocity (a vector quantity: magnitude and direction) as its neighbours, then it is logical that it would not collide with its neighbours.Therefore it tends to maintain the spacing established by collision avoidance.

Rule3: Flock centering



Flock centcering tries to keep the members of a flock in unity. It is represented by a vector that directs the particle towards the centre of the flock.

There can be two cases:

(i)Global

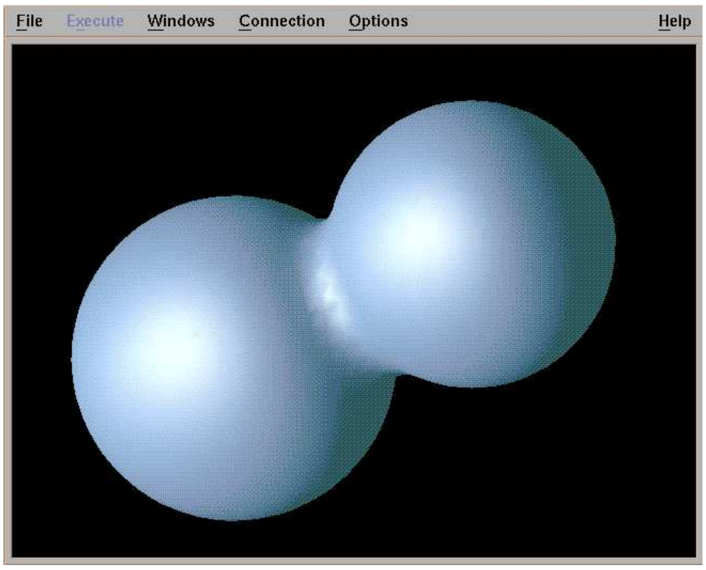
A global flock center can be defined as the average position of all the particles in the flock.

(ii) Localized flock centering takes only the positions of the particles' nearby flockmates into account. This aspect turns out to be essential for a proper flocking simulation. This has been approached in the model, and causes some nice splits when avoiding an obstacle.

By reversing the vectors, the rules can be changed so as to accomodate different types of behaviors.

2.9 Implicit surfaces

An oject described by an implicit surface may be thought of as a particle surrounded by a density field, where the density attributed to the particle decreases with distance from the particle location. This represents the influence of the particle. A surface is implied by taking an isosurface through this density field - the higher the isosurface value, the nearer it will be to the particle. The way in which these surface can combine is what makes them a powerful tool in computer graphics. By simply summing the influences of each implicit surface object(also known as a blobby) on a given point, very smooth blendings of the spherical influence fields can be obtained. An example is shown below:



2.10 Texture mapping

Texture mapping is the process of laying an image(the texture) onto an object in a scene. The image content may be interpreted as different types of information, for example normal values defining the normals of an object at a particluar point(normal map) or a factor that modifies the opacity at each point. Texture maps provide a powerful way of adding details and attaining various effects that are not easily obtained or require specific software. Generally, mapping a texture over an object consists of modifying the colour of the object at each pixel by the corresponding colour from the texture image. Texture maps are the basis of techniques such as depth maps, bump maps and normal maps and is a matter of how the image data gets translated for the purpose.

2.11 Refraction

Refraction is the bending of light as it passes between materials of different optical density. It is due to the change in speed and wavelength, λ, at the boundary between two materials. The more dense the material, the slower the speed of light in that material. This relationship is captured by the refractive index, n,(2.1) of the material:

eq.(1): In terms of speed, $n = \frac{c}{v}$, where c is the speed of light in vacuum and v is the speed of light in the material. As the density of the material increases, v decreases, and consequently, from the equation, $n$increases. Hence, $n > 1$ for all materials with $n = 1$ for vacuum.

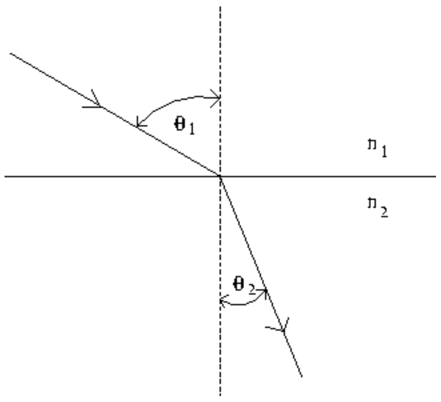


Figure 3: Snell's law

eq.(2): In terms of wavelength, $n = \frac{\lambda_0}{\lambda}$, where $\lambda_0$is the wavelength of the light in vacuum and $\lambda$is the wavelength of light in the medium. Consequently, light splits up into the spectrum colours depending of the distance travelled, to display a series of complex and intriguing patterns.

Snell's law relates the refractive indices, n,of two media to the directions of propagation in terms of the angles to the normal, and is given by

$\frac{n_1}{n_2} = \frac{sin\theta_2}{sin\theta_1}$

Figure 3 illustrates Snell's law as a ray of light is refracted.

2.12 RIB file format

The RIB(Renderman Interface Byte Stream) file format provides an interface between the modelling program and a particular renderer. Rib files are generated via calls to the C API and stored to disk, before being sent to the renderer. This means that the rib file can be observed and modified further if necessary. A rib file is stored as a text file and can therefore be created without complexity while providing powerful resources and techniques.

2.13 tx format

Before use as a texture, images need to be converted into texture coordinates (s,t). When using PRMan, textures should be converted into Pixar's proprietary file format, the .tx format. the command $txmake$ converts an image file into a texture file with a $.tx$ extension.

2.14 Shader variables

Shaders essentially contain two types of variables: uniform and varying variables. Uniform variables hold values that are constant over a surface, while varying variables contain values that vary from point to point over the surface. Variables declared in the parameter list of a shader are uniform by default, and variables declared in the body of a shader are assumed to be varying. Specifying variable as uniform and varying help in optimization and improve rendering speeds.

3 Project Overview

3.1 Aims

The aim of this project is to create a realistic animation of water droplets meadering down a surface, taking into account interfacial dynamics.

The flow of water droplets along a surface is a complex process, with a number of factors affecting this behavior. The project aims to identify the donimant parameters, for example, gravity, surface roughness and interfacial tensions, of such a dynamical system, and construct a texture map technique that depends on these parameters.

Note: At this early stage, the project will not be addressing GUI issues. Instead it seeds to identify the influences that will create the effect desired, that is, that of droplets running down a glass pane.

The dynamics involved in certain droplet behaviors, for example, merging and splitting, are computationally intensive. Attention needs to be paid to the internal structure of the water droplet, contact angle hysteresis2.3.3 for an accurate simulation, and forms a separate project altogether. Instead, the project will develop techniques that will simulate the effects of merging and splitting, according to surface properties. Hence, similar visual results may be achieved while avoiding the major computational load of these processes.

The project relates most to the model presented in [10] for the use of texture maps for distributing parameters across the surface. The flow is based on the experiments carried out in [1] which integrates the dynamics of droplets into the systems.

3.2 Tools and libraries

GraphicsLib Library

The GraphicsLib library provided by Jon Macey is used for point and vector manipulations, and contains the functions required by the system fo determining the motion of the particle system.

Libtiff Library

The libtiff library has been developed for reading and writing tiff images. It also includes a set of tools for the manipulation of tiff images, sufficient for what the system seeks to achieve with the images generated.

The Renderman Standard

The renderman standard is chosen for the complex system of shaders it offers and its high control over surface properties.

RendermanProServer 12.0 has been chosen as the renderer because it introduces some new features to the renderman shading language from which the shading model in the system benefits from. Allowing dynamic arrays to be passed in the shader arguement list has only recently been added to the renderman SL. The number of droplets in the list changes from frame to frame, and hence, using dynamics arrays qallows droplet information to be passed to the shader which can then define the appearance dynamically.

4 The model

The model for the droplet system can be split into three components:

1. A particle system to define the flow of water droplets,
2. A texture mapping technique to store updated values corresponding to each individual droplet,
3. And a rendering method for the appearance of the droplets.

Haivng these different components provides the ability to handle each part separately.

Sections 5-11 now describe each of these sections more in depth.

5 Droplet Particle System

The particle system, where each droplet is represented as a particle, describes the flow of the droplets. It uses Newton's equations of motion to determine a global(primary) flow over the surface, and a discrete model to trace the path of an individual droplet.

Particles get created at random positions over the surface via a random number generator. The initial values are defined before the simulation begins, based upon the environmental aspects, for example, the strength of the wind. Its set of attributes, listed in**??**, determines its state at the current simulation time and gets updated as the particle moves to its next position in time. The Euler Method, mentioned in 2.7.1, is used for the numerical solver. It feeds on the initial direction vectors set at the start of the process, and using the classics dynamics $F = ma$, a new velocity, v, and position, p, is calculated for the next time step. Motion is controlled by rules based on the dynamics and interactions of the droplets with the underlying surface, for example, the interfacial tensions between the liquid and solid, surface roughness and affinity for water. The flow in turn, controls the appearance of the particles, depending on factors such as amount of water it contains.

Image26 on the Color Plate show the droplet positions(blue) on the surface as applied by the shader using dynamic arrays.

The particle system offers a simple way of controlling the different aspects of the animation.

5.1 Droplet Flock

The particle system is elaborated into a flocking system in order to include particle-particle interactions, which is essential for a realistic animation of droplets. Interactions would define how they collide, merge or separate from each other, while taking into account surface properties. Rules are defined following a localized method, as described in C.Reynold's paper, [13], that is, droplets flowing down the one end of a glass plane under gravity are not likely to interact with those at the other end. This assumption improves the efficiency and speed of the algorithms.

According to [13], the 3 rules that describe a flocking system are collision avoidance, flock centering and velocity matching.

6 Flow and Dynamics

There are three main sources for the forces that drive the motion along the surface:

Gravitational force, a frictional force due to interfacial tensions, and a force resulting from discrete model used to define the meandering algorithm14 for the droplets.

6.1 Global flow

From the water sources, in this case, rainfall, water flows downwards under the influence of gravity. The flow depends on the surface that the droplets are interacting with, the quantity of liquid and its presence on which part. The flow may split into streams due to various factors such as surface impurities. The global flowof the droplet particles in the system is based on the external forces acting, that is, the motion is described by integrating Newton's equations of motion under gravitational influence.

6.2 Individual flow

Each droplet defines its own path by computing its position over the texture map each time. This way, the each position is taken into account and an individual path defined. This is achieved through the meandering behavior, elaborated in13.1.

Flow pattern

A global flow pattern arises from each droplet's individual pattern. This type of flow is most effective when described with vector fields, which has been left as future work, 23. The main idea is to be able to specify a unified direction towards which the droplets can flow, essentially behaving as a flock. This basis is used to apply a global wind effects13.4, that eventually get passed down to each droplet individually.

6.3 Internal and external forces

6.3.1 Internal forces

The internal force is modelled as a resistance force opposing the direction of motion of the droplet. It describes the interfacial tension forces [2.1] between the liquid droplet and the solid surface, and is affected by the affinity for water.

6.3.2 External forces

External forces depends on the surrounding environment, for example, the direction of incident water.

Currently, the model runs under the influence of gravity, causing droplets to flow downwards. Any other external forces such as wind may be added. 13.4

6.4 Path of water droplets

The path of a droplet describes its own pattern, depending on its deviation towards its next position in time due to surface roughness and impurities present.

6.5 Kinetic impact on collision

Droplets in motion carry around kinetic energy and may collide with each other. The way their energy is distributed after the impact, depending on factors acting at the collision point,p_c, determines their subsequent behaviors. Largely due to the adhesion properties 2.5.3 that water exhibits with the surface, merging is the most likely action, where the energy gets added at p_c. Subsequently, the momentum, mv, increases in both directions, relative to p_c, preserving the conservation of momentum. That is,

if $m1v1$ =momentum of droplet1 = mass1*velocity1 and $m2v2$= momentum of droplet2 = mass2*velocity2,

then, after collision, their combined momentum would be $m1v1 + m2v2$.

By referring to Newton's 2nd law of motion, which states that the greater the mass of an object is, the harder it is to change its speed, how much a droplet will be affected by a collision also depends on the droplet's mass(size) at p_c, as can be seen from the above equation.

7 Shape of water droplets

A droplet is not perfectly symmetrical in real world as assumed in many models, for example, in [2]. Instead, it gets altered by forces such as gravity, interfacial forces, viscosity of the liqiud and the orientation of the surface. These factors are important for consideration in order to achieve a realistic animation.

7.1 Interfacial tensions, γ, and contact angle, θ

Ideally, the shape of a droplet and the magnitude of the contact angles with the surface are controlled by three interaction forces of interfacial tension γ at each participating phase(such as liquid/solid). These forces are related by the Young's equation:

$\gamma lv \cos\theta = \gamma sv - \gamma sl$

Note: The above equation holds true under ideal conditions. Effects from surface roughness and impurities create deflections from this ideal relationship.

If a horizontal surface is considered, a droplet(assumed to be in a stationary state) will satisfy the equation

$\gamma sl - \gamma s + \gamma l \cos\theta = 0.$

During its course along a surface, a water droplet exhibits different shapes depending on various factors such as its speed, the surface roughness and the contact angles at the interface.

For example, the contact angles in its head(θa)and its tail(θr) as the droplet rolls, change for each time step(refer to 2.3.3). The higher its speed, the higher the air resistance on the shape.

Figure 4 show the relationship of the interfacial tension at each participating phase between a droplet and its underlying surface.

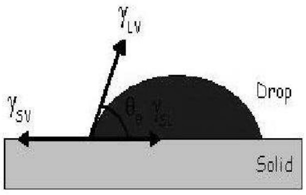
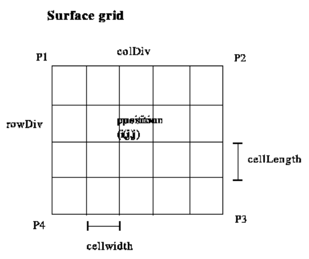
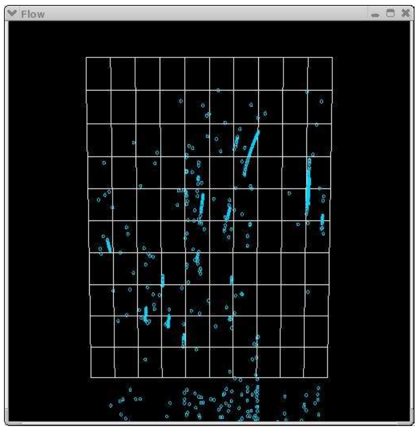


Figure 4: Interfacial tensions betewen a droplet and the underlying surface



a. A cell defined by a position (i,j)



b. A screen shot of the 10x10 surface grid

Figure 5: The surface grid

7.2 Energy

In the case of liquid droplets, the energy involved in the motion (in this case, energy dissipation occurs at the border line and centre, [1]) of each droplet is much higher than that involved in its shape deformations. Therefore, the flow can be generated separate of the shape, where as the shape must be simulated with motion inputs.

8 Surface Model

A discrete model is developed in order to intergrate certain rules that would control the flow of droplets over the surface. The surface geometry is describes as a simple flat 2-dimensional grid, made up of nxm cells. The position of each droplet placed on the surface grid may then be defined in terms of a position on the grid, that is, a cell(m,n). The algorithm for retrieving the position on the surface grid is given by the algorithm 3 The flow is simulated by moving each droplet to a next cell position, based on some defined rules, at each timestep.

The discrete model is set to a low resolution of 10x10, representing a coarse level of the system. The idea is to be able to have a low level from which a basic outline of the flow can be obtained, as shown in Figure 5.

The discrete method allows several surface factors to be distributed over the surface, for example, the affinity value may be spread across the grid in a ramp. The effect created would hence have droplets gradually sticking more to one side of the surface and more interesting variations would exhibit different behaviors.

Similarly, a dry surface may be simulated by setting every wetness value of each cell to 0. As the droplets begin to flow, the wetness values of the underlying cells start getting updated, which then affects the flow of following droplets.

A per-droplet meandering behavior describes how the flow is controlled over the grid. (please refer to 13.1)

9 Wetting

Water spreads over the surface when

$\gamma s > \gamma sl + \gamma l$

This effect relates to the quality of wetting. That is, good wetting will cause a drop to spread over a larger area of the surface, giving it a small contact angle . Increasing θthus exhibits poorer wetting, but a more definite shape for the

droplet.

Calculation of the spreading coefficient, S,2.1 also defines the wetting and is given by:

S=$\gamma sv - (\gamma lv + \gamma sl)$

if(S>0), then there is complete wetting.

if(S<0), then there is partial wetting.

By observation, wetting may be characterized by the time taken for the remaining water, R, left behind a flowing droplet to split into smaller droplets. Poor wetting will produce tiny droplets from R in a very short time (a short lifetime for the film). A good example of this would be the defined droplets of water on a waxed car. Better wetting will split R after a longer time (giving the film a longer lifetime). Experiments carried out by IM.Janosi and V.Horvarth[1] show figures of 1-2min for the lifetime of a film dueem. to good wetting.

9.1 Identifying parameters for wetness

The sticking of water droplets is difficult to model accurately since it involves a lot of unmeasured parameters, such as the contact angles 2.3.2 (note: methods for measuring contact angles have been developed and require special equipmnet(please refer to [3]). By taking into account the factors described in the previous section, parameters can be defined to simulate the wetting phenomenon.

If the shape of the droplet and its flow are separated into two components, then the contact angle could be used to link them together. That is,

Shape and Flow -> contact angle θ -> system

The spreading coefficient, S, would relate interfacial tensions and thus the surface roughness to the amount of wetting.

The surface's affinity for water contributes a lot to the amount of wetting exhibited. The impurities present define the roughness of the surface and hence adhesion. The system uses an affinity map to spread the affinity over the surface as this parameter gives a gradual and effective change in the behavior of the droplets as they flow.

9.2 Remaining mass, Mr

The mass of water that is left behind due to wetting is a percentage of the droplet's mass factored by the affinity for water of the surface. It is defined in [2]Section 2.1,

let $m'_{(i,j)}$= remaining mass,

$m_{(i,j)}$=mass of droplet,

m_{min}=minimum mass, m_{max}=maximum mass, $w2$=parameter for remaining water

Then,

$m'_{(i,j)}$=$m_{(i,j)}$ (if $m_{(i,j)}$< $f_{(i,j)}m_{min}$),

$m'_{(i,j)}$= $w2f_{(i,j)}(m_{(i,j)}$- $f_{(i,j)}m_{min})$+ $f_{(i,j)}m_{min}$ (if $f_{(i,j)}m_{min}$≤$m_{(i,j)}$ ≤$f_{(i,j)}m_{min}$+ $\frac{m_{max}-m_{min}}{w2}$),

$m'_{(i,j)}$= $f_{(i,j)}m_{max}$ (if $m_{(i,j)}$> $f_{(i,j)}m_{min}$+ $\frac{m_{max}-m_{min}}{w2}$)

note:in surface->updateWetting() wetting is directly related to remaininM. That is, wetting[]=remainingM*2.

10 The Texture Maps

The flow of droplets over a surface is influenced by a large number of factors, for example, evaporation, surface impurities and adhesion. Managing these parameters as they get changed dynamically during a simulation may become complex, especially if they are used as input for other actions. Texture mapping provides a simple, flexible way of storing and retrieving such parameter values and produced accurate results if managed properly.

At the moment, 2 texture maps are attached to the flat surface, distributing surface property values across it. The maps used are 256x256 greyscale tiff image files and reasons for this are discussed in 16.3.

The system applies texture mapping for the two main reasons that are given in sections 11.1 and 11.2:

Texture resolution

The maps provide a greater level of accuracy in the individual motion of the droplets. The higher the resolution, the more precise the path of the droplet will be.

That is, for example, a droplet might move over several cells in one time step, but only two positions will be know, the position at the current time and next time. Therefore, a higher resolution would yield a more accurate approximation of its path.

Throughout the simulation, the values get updated based on a set of defined rules that control the motion.

Distribution over texture map

The maps provide a simple and effective way of distributing the values across the surface. This will allow the animator to describe their own map. For instance, a ramp from black to white may describe low affinity to high affinity.

10.1 Affinity Map

The affinity map is used to spread affinity values2.5 over the surface at a finer level of detail. It is loaded at initialisation into an affinity buffer and remains unchanged throughout the simulation. The values are retrieved from the map in order to calculate the remaining mass, m',9.2 that a droplet leaves behind. Different maps result in different simulations, see 16.3.1.

The affinity for water droplets, 2.5.2 depends on the underlying surface. A low affinity value defines a slick surface while a high affinity relates to dull rough surfaces. This is taken into account during the implementation,16.3 when the values from the map get loaded.

10.2 Dmass Map

Unlike the affinity map, the Dmass map is constantly being updated at each frame. This is because the map value at a point gets modified according to a droplet's varying mass over the surface at that point, if there is a droplet present there at the time.This map then gets sampled by the shader to determine an amount by which light will be refracted. The algorithm, described in14, looks for the positions on the texture that have been visited by each droplet, *d*, and marks them by factoring the texture value at that position by the mass of *d*.

11 Shader Model

A surface shader that is controlled by the parameters of the underlying surface and those of the flow of droplets over that surface is presented. The idea is to obtain a physically correct simulation of a droplet's behavior over time. For this to be the case, the dynamic relationship between the surface and droplets covering it needs to be modelled. The particle system,5, will drive the droplets over the surface. The (point)positions carry with them values that would then be passed as parameters to the shader at each time, allowing properties such as the index of refraction to be altered according to each droplet and the surface area that it covers.

Two methods have been designed that consider the distortion occuring underneath the droplet, that is, on the surface that it covers and are describes in section 17.

The positions and points to be passed to the shader are varying and therefore should be passed to the shader as varying values. REnderman allows varying parameters to be passed in its arguement list. By using these updated values, a droplet's irregular shape, its tendency to merge and the effect of gravity on them, have been achieved.

12 Pipeline

The system consists of several different components that get linked together in the pipeline. It is essential to ensure that each aspect would perform correctly as intended before any elaboration. The first step, therefore, was to set up a basic model of the main parts so as to test their compatibility and connections with each other.

Three major components were identified: A particle system, a texturemap and the rendering process.

1. On creation, the particle system is the first element in the pipeline. An affinity map is fed into the program and is used to define the motion of the particles over the surface. These updated positions control the appearance of the droplets for every frame.

2. The appearance is acheived via a surface shader that simulates the effect of droplets flowing on a surface. The program generates the series of rib files that apply the shader to geometry.

3. The program generates an updated texture map for every frame that gets called by the shader. Before the rib files are sent off to render, these updated texture maps must first be converted to the tx format (required by the PRMan for texture mapping).

4. Once the texture maps are ready, the rib files are sent to the renderer and rendered off as tiff images.

When a working model was reached, more conponents could then be added to the system in order to suit for different instances.

5. Trails for the droplets are created by processing the rendered tiffs. Hence, a tiff library that contains operations for handling tiff images is included to handle tiff read/write processes.

This is explained more clearly on the pipeline flowchart in Figure 6.

12.1 Shell Scripts

In order to connect certain elements in the pipeline, the system relies on some shells scripts to take care of file formats and batch rendering. The scripts that have been defined for use are:

txmake.sh

This script converts a set of tiff images into corresponding textures that Renderman can use, that is, images in the tx format. Essentially, it searches for all the tiff files in a directory and changes each of their '.tiff' extension to '.tx'. This script must be run before any rendering is carried out, so that the .tx textures are located when a renderman shader is applied.

Render.sh

This script loops through all the rib files in a directory and renders them out. That is,

```
for every rib file, file.000f.rib, in the directory
{
render file.000f.rib
}
```

where f is the framenumber

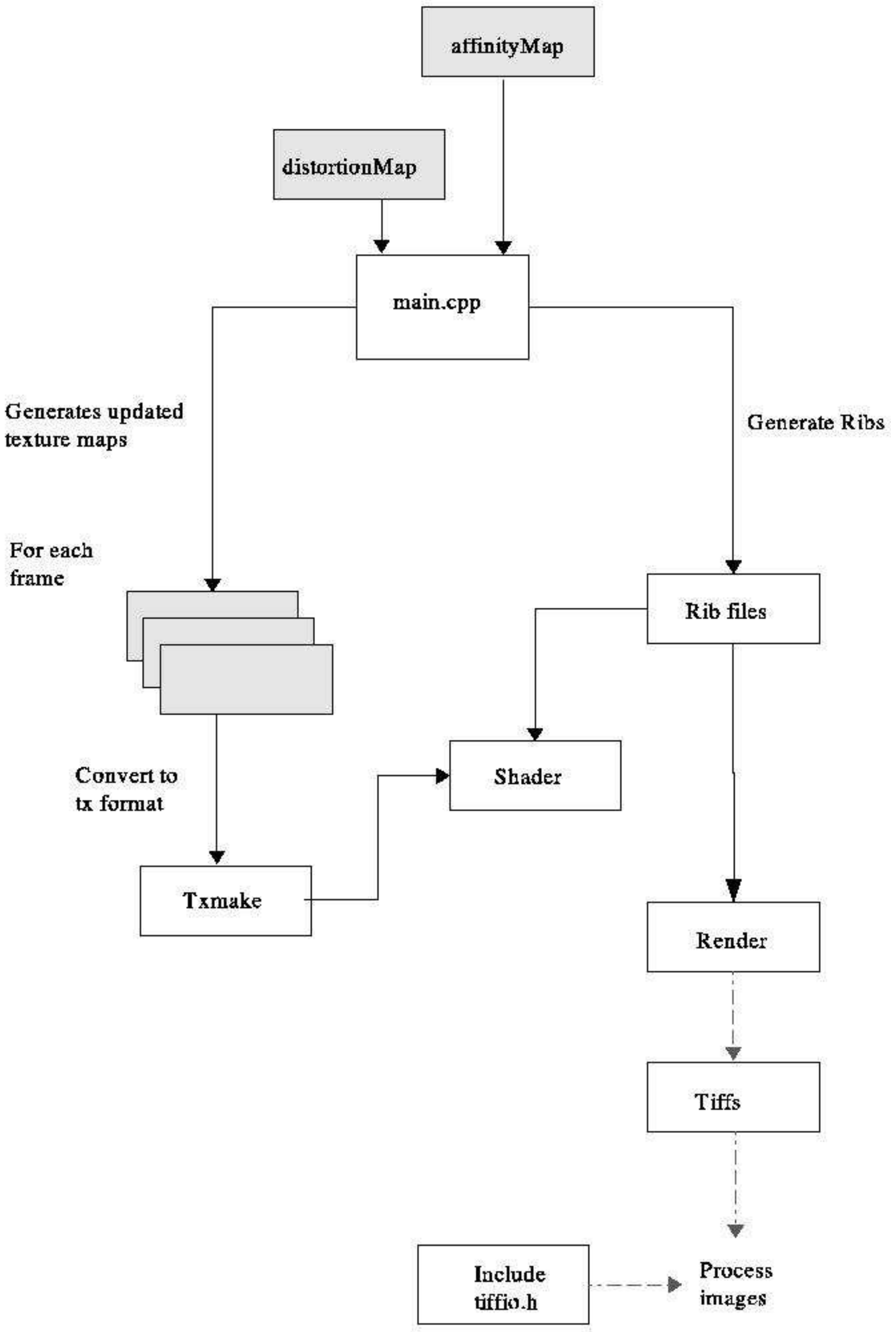


Figure 6: System flowchart

setDir.sh

This script has been written in order to help a user set up the correct paths and directories before running the program. So far, the focus in the development of the system has been to obtain a correct animation based on surface parameters and the dynamics involved. Not much attention has been paid to creating a user friendly environment. Hence, it might get confusing when trying to obtain a correct simulation if a new user attempts to obtain a correct simulation. setDir.sh sets up the initial directories that have been defined in the source code and avoids the user to have to set it up.

13 Behaviors

13.1 Meandering

The meandering behavior is a per droplet action that describes its individual flow over the surface. The core of the implementation for this behavior comes from the experiments carried out by I.M Janosi and V.K Horvath, [1], on the dynamics of water droplets on a window pane.

The process is applied to the discrete model of the surface, described in 8, which is currently set to a 10x10 grid.

Meandering is dependent on the underlying surface properties, which may vary across the surface. If the droplets are in motion, then the property values at each point on the surface need to be taken into account. By storing the value of each property in a texture map, their values at any position on the surface may be retrieved. Therefore, for more defined behaviours, a high resolution map is required. At the moment, a 256x256 map is used from which values are retrieved from, based on the position of the droplet. These values are then used in the meandering algorithm, defined in **??**.

The algorithm considers only 3 positions at a time, and a decision is made on which one to move towards.

The next position of the droplet is based on the amout of water present ahead of its course. There are 3 cases:

1. If water exists at only 1 of the 3 positions
2. If water exists in more than 1 position.
3. if no water exists at any of the 3 positions.

In the third case, if there is no water present at any of the 3 positions ahead, then the droplet moves according to a decision factor, d, related to the affinity for water. It is given by the equation

$d_{(i+k)} = w_1(\varphi, k) f_{i+k,j+1}$

where k=(-1,0,1)

$d_{(i+k)}$ =the tendency for the droplet to move in the direction $(i + k, j + 1)$,

$w_1(\varphi, k)$ =parameter depending on an angle of the plane's inclination , and with a range $0 \leq w_1 \leq 1$

13.2 Merging

When droplets are within a certain vicinity to one another, they merge together to form a single heavier drop. Based on strength, each of their bonds break, get deformed and connected. The process of merging water drops is closely related to implicit surfaces,2.9, as applied in the paper by Y.J.Yu, H.Y.Jung and H.Cho[15], in which droplets are under the influence of gravity. Unlike their model, the approach taken by this system does not deal with implicit surfaces, but rather applies the principles of it to the method. The algorithm listed in 14 describes how the dynamics of merging droplets is handled.

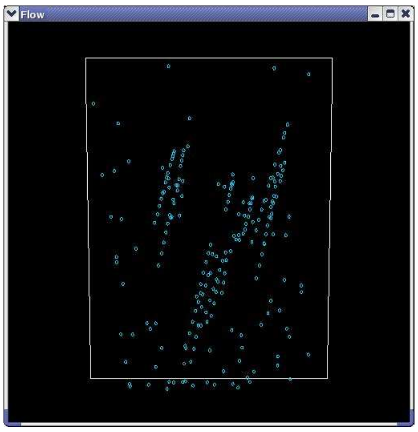


Figure 7: droplets splitting

13.3 Splitting

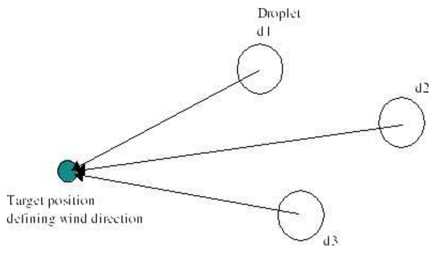
A droplet in motion tends to split up into smaller masses, in reaction to various factors such as interfacial tensions between the liquid and solid, its internal state and the surface roughness. These masses would then follow their own path, or the path of a nearby droplet with the greatest field of influence. Adhesion to the surface2.5.3 greatly contributes towards this behavior. Placed under the action of external forces, a droplet tends to move towards the resulting vector, F, of these forces. Depending on the magnitude of F, a high adhesion factor would act as a counterpart to this force and cause the droplet to split. If continuous, splitting would then result in a trail formation.

An accurate physical representation of splitting would require a tremendous amount of complex processing on its internal and external structure that are beyond the scope of this project, for example, the movement of liquid within the droplet, the capillarity of its surface and the interaction points with the surface. Instead, an effect of splitting is intergrated in the system by relating the process to the radius of a droplet. An outline of the algorithm is given in**??**. It takes the maxRadius attribute for a droplet**??** as the criteria against which the droplet is checked at each time step in order to determine whether the droplet should be split or not. The higher the maxRadius, the less likely the droplet is to split.

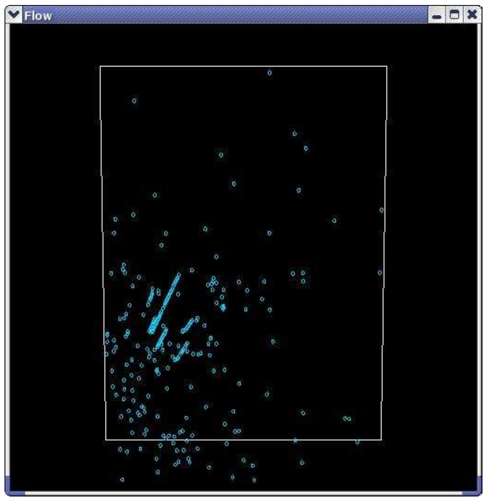
The screenshot in Figure 7 demonstrates splitting which results into a trail being created.

13.4 Wind Effect

External forces such as gravity and wind influence the flow of droplets. Whereas gravity constantly acts downwards, a wind vector varies, according to several enviromental aspects, for example, light breeze or any source that suggests a movement of air. The model simulates wind by applying a vector direction to each particle. The droplets are treated as one flock, for which a target point, pt, can be specified. By calculating the resulting vector from each droplet's position to pt, the wind direction is defined and applied to the droplet. Hence, the strength of the wind can be specified by the magnitude of this vector. Refer to Figure 8a. and 8b.13.4 for an example of this effect. The animation found at 22.6 uses a vector of (25.o, -11.0, 0.0) to create the gust of wind across the glass surface. A screenshot of it is shown in Figure 8c.



a. Droplets moving towards target wind position



b. Screenshot of the particle system with wind

c. A rendered shot of wind effect.

Figure 8: Adding wind effect

13.5 Trail Generation

A droplet, d, flowing down a particular surface generates a trail. This behavior is strongly related to what is called the wetting phenomenon9, which depends on the underlying surface properties. The trail outlines the path of d, and is composed of the mass of water, m' left behind by d. If the surface is exhibiting good wetting, then m'appears as a film of liquid rather than a droplet shape. On the other hand, if $m'$is affected by poor wetting, then the trail beads up onto the surface, tracing the course of the droplet as a string of little beads.

The trail behavior in the system is the result of the splitting action13.3. When a droplet reaches a certain size, it splits into 2, and an amount gets left behind, subsequently forming a section of its trail. The trails created in this case depend on the initial values defined at the beginning of the simulation.

A second level for trail generation is added by directly modifying the color values of the images that come out of the rendering process. The implementation of this method involves handling tiff images, discussed in more detail in 16.6.

The algorithm that produces the streams consists of 2 parts:

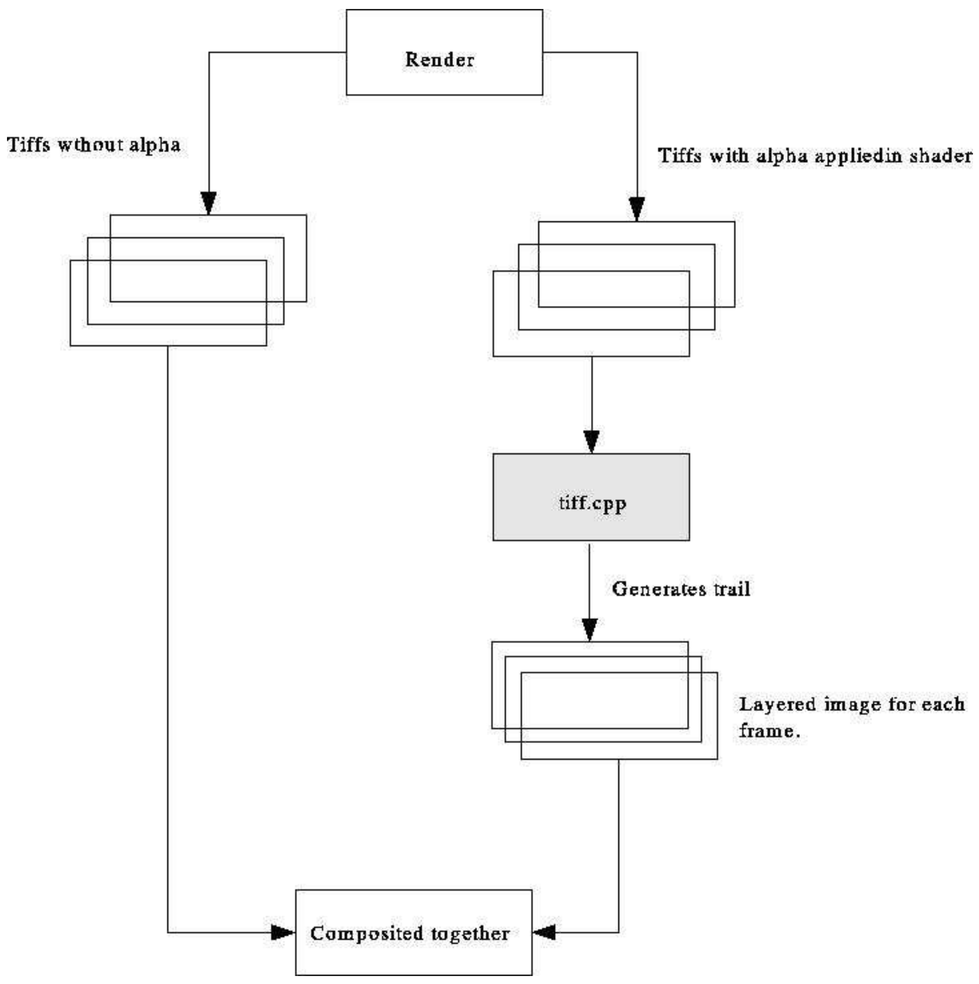


Figure 9: Trail generation process

(i) Loading in and layering a set of images

and (ii) writing out an image composed of a set of images, for every frame.

By thinking of the process backwards, the desired output of the trail process would consist of a sequence of images that is able to visually trace the route taken by a droplet.

If all the rendered images were compressed together into one, the result would be an image in which the droplets are shown in all of their positions for each frame, which essentially forms the trail. Hence, in order to animate this effect, each written image in the sequence would be composed of a set of layered images from the previous frames. That is,

```
for every frame f in the sequence,
Load in frames 1 to f.
Apply a blending function such that frames 1-f can be merged together
Write out the merged image
```

A flowchart illustrating the method is shown in Figure 9.

The algorithm for loading in each set of frames and writing out the merged frames is describe in 14

Note: The trail generation process is caried out on the $showA.000f.tiff$ rendered images. That is, the images that have been rendered with alpha=0 17.3.1 are used, since only trails to the droplets need to be generated.

13.5.1 The blending function

The images loaded are RGB tiff images. A blending function, based on the methods used for blending in OpenGL, is defined in order to merge the frames together. Since the images do not contain an alpha channel, a value, ranging from 0-1, is approximated for the alpha. The blending function takes a small percentage of the color being loaded in and adds it to a percentage of the desired image(the destination image).

```
    //specify an alpha value
    float alpha=0.07;
    //given p=R,G,B
    //factor out each pixel by alpha
    tempP=p*alpha;
    //factor out each pixel of the destination image
    //by (1-alpha) and add the loaded color
    destP= destp*(1-alpha)+ tempP;
```

The written images appear quite dark and subtle, given the low alpha factor, but for the purpose of what is being aimed for, the algorithm has been designed to work with low values. For instance, at the start of a simulation, droplets are spread randomly over the surface, with a very small mass. These masses get increased by a random number each time until they exceed the static critical mass. If the rendered images at the start are layered one on top of the other and corresponding pixel values get blended together, there is a risk that the final color of certain pixels reach white, The low alpha value helps in preventing the value to get too high.

Algorithm 1 Merge droplets

```
For every droplet, $d$, in the droplets list, compare with every other droplet, $d_N$.
If the distance between $d_N$ and $d$ <= the radius of the neighbourhood of $d$,
then $d_N$ is considered a neighbour to $d$, and a collision check is performed between the two droplets.
If the distance between $d_N$ and $d$ <= the radius of $d$, then a collision has occured and $d_N$ and $d$ are merged into one droplet. The mass of $d_N$ is added to the mass of $d$. The new radius is determied from the mass by calling updateRadius.
The velocity of the merged droplet = $\frac{(mass_{dN} * v_{dN} + mass_d * v_d)}{(mass_{dN} + mass_d)}$, using the law of conservation of momentum.
```

14 Algorithms

Merging droplets

The merging behavior considers only the neighbours of the droplet being tested. If the distance between them is less that a certain defined threshold value, the two droplet will merge together. Currently, The threshold is set to be the droplet's radius and represents a field of strength that emanates from its centre. A higher threshold would create a stronger field, simulating a higher need for making bonds with other neighbouring droplets.

Processing images

Note: The system assumes that the tiffs being loaded in are RGB tiff images.

The algorithm for layering multiple images approximates an alpha value and factors each pixel loaded in by the alpha.

Trail generation

A separate program has been designed to handle the trail generation (Tiff.cpp), due to the fact that the trails are processed after the rendering stage.

Methods for loading and decoding an image are the same to loading in the texture maps. The difference comes in the processing of the data read in and therefore, will be the main focus. The algorithm allows the starting frame, the number of frames to generate and the number of previous frames for blend, to be specified.

Algorithm 2 Meander : a per-droplet action

1. Get the current cell of the droplet on the surface grid: $cell(i, j)$
2. Retrieve the wetness value at (i, j) :

```
currentWetVal = wetting[col+(colDiv*row)];
```

where col+(colDiv*row) gives the index in the array corresponding to the current position.

3. Get the current cell, $cell_t$, of the droplet on the texture map: $cell_t(i_t, j_t)$

4. Need to consider the cells on the edges. If a droplet has reach the end of the surface as it is flowing, then call arriveAtEdge().
5. Find its remaining mass, m': findRemainingM($cell_t$,$cell$) 9.2
6. Update corresponding wetness value based on m', so as to be ready for the next visit to this cell.
7. Get the wetness values, w_1,w_2, w_3, at the 3 next possible positions, i.e at $p(i-1, j+1)$,$p(i, j+1)$,$p(i+1, j+1)$ respectively, on the grid.
8. Compare these wetness values and move to where there is more water present:

[CASE A] If there is water present at 1 place, move there.
if(w_1|| w_2|| w_3!=0) then there is water present at one of these positions. Determine which position.
If it is w_1, then its next position, p' =$(i-1, j+1)$.
step 1: Update its mass: $mass_D = (mass_D + m'(p')) - m'(i, j)$.
step 2: Then move to p':

```
applyForce((p'-p));
```

where applyforce(f) moves the droplet towards the directional vector f.
else if it is w_2, then its next position,$p' = (i, j+1)$.
Repeat step 1 and step 2.
else if it is w_3, then its next position,$p' = (i+1, j+1)$.
Repeat step 1 and step 2.
[CASE B] If water exists in more that 1 position.
if there is water at $(w_1\&w_2)$or$(w_2\&w_3)$or$(w_1\&w_3)$or$(w_1\&w_2\&w_3)$
then move to the cell with the highest priority, that is, $p(i, j+1)$.
Repeat step 1 and step 2.
if no water exists at the highest priority, then move in the direction with the largest mass,A, where $A = m'$at the next position.
Largest mass, A_1, at $(i-1, j+1) = m'(i-1, j+1)$
and largest mass, A_3, at $(i+1, j+1) = m'(i+1, j+1)$
if $A_1 > A_3$, then move to $p' = (i-1, j+1)$. Repeat step 1 and step 2.
else if $A_3 > A_1$, move to $p' = (i+1, j+1)$. Repeat step 1 and step 2.
[CASE C] If no water exists at any of the 3 positions.
if $(w_1\&w_2\&w_3) = 0$, there is no water at any of the next 3 positions.
then move according to decision equation, :
decision factor, d_1, at $(i-1, j+1) = x_1 * affinity$ at (i_t-1, j_t+1) on the Affinity Map.
where $x_1$is a parameter depending on an angle of incl. for the surface at that cell.
decision factor, d_2, at $(i, j+1) = x_1 * affinity$ at (i_t, j_t+1) on the Affinity Map.
decision factor, d_3, at $(i-1, j+1) = x_1 * affinity$ at (i_t+1, j_t+1) on the Affinity Map.
Compare d_1, d_2, d_3 and move towards the largest.
if$(d_1 > d_2)\&(d_1 > d_3)$,then move to $p' = (i-1, j+1)$. Repeat step 1 and step 2.
else if$(d_2 > d_1)\&(d_2 > d_3)$,then move to $p' = (i, j+1)$. Repeat step 1 and step 2.
else if$(d_3 > d_2)\&(d_3 > d_1)$,then move to $p' = (i+1, j+1)$.Repeat step 1 and step 2.

Algorithm 3 Determine the position of a droplet on the surface grid and on the texture map

```
1. Get the row on the surface grid onto which the droplet lies: divisionsR=10/Length * (P1.y-Pos.y), row = floor(divisionsR)
2. Get the corresponding column: divisionsC= 10/Width *(Pos.x-P1.x), column= floor(divisionsC)
3. Retrieve the position (row, column) in the array: position= gridArray[ col+ (surface->colDiv*row) ]
```

Algorithm 4 Split a droplet

```
For every droplet, $D$, in the list of droplets, check to see whether the radius of $D$ has reached its maxRadius yet.
if $(radius_D >= maxRadius_D)$, then split the droplet into 2.
Define an offset position $offp$,
Create and add a new droplet to the list at position $offp$,with a radius $radius_D/2$ and a mass $mass_D/2$.
```

Algorithm 5 Load in the affinity map

```
At initialisation, allocate memory for the affinity buffer.
Open the tiff image. Using libtiff defined procedures, load the image.
for every pixel in the map,
{
get the red channel value, $R$.
Make sure that it is not 0 so as not to divide by 0 later on. If $R = 0$,then $R = 1$.
Modulate $R$ with an affinity factor, $aff$, to reduce it to a more useable value.
Add $aff/R$ to the affMap list.
}
```

Algorithm 6 Load in an image for layering, based on a blending function.

```
1.Open the tiff image for reading.
2.libtiff defined function used to read in the image: TIFFRGBAImageBegin()
3.Allocate memory for buffer into which the image will be read in. Each pixel is allocated 32 bits, i.e 8 bits per sample.
4. Start the process: TIFFRGBAImageBegin().
4. Read and decode the image using TIFFRGBAImageGet().
5. Define a suitable alpha factor.
6. for every pixel in the image, process pixel data:
Get the R,G,B values using defined libtiff functions: R=TIFFGetR(); G=TIFFGetG(); B=TIFFGetB().
7. Apply alpha factor to R,G and B: $tempR = R * Alpha, tempG = G * Alpha, tempB = B * Alpha$
8. Make sure that the factored R,G,B don't go beyond 255.
if $(tempR \geq 255)$then $\{tempR = 255\}$. Repeat for G and B.
9. Apply (1-alpha factor)to the image being layered (destination image) and add factored R,G,B to it.
$buffer[s * 3] = buffer[s * 3] * (1 - AlphaFactor) + tempR$. Repeat for G and B.
10. Make sure that the updated layered image doesn't go beyond 255.
11. Release resources: TIFFRGBAImageEND().
```

Algorithm 7 Updating Dmass texture map

```
Create an array TexArray that will determine whether a droplet is present at a point or not:
TexArray[$texturesize$]=0.0;
On each frame, $f$,
for every droplet, $d$,
1. Get its position on the texture: tpos= $d.texPos$.
2. 'Mark' this position $tpos$so as to indicate that it has been visited by a droplet: value at $tpos$ marked according to $mass_d$
texArray[$tpos$]= $1.0/mass_d$
3. Load the texture map for current frame $f$. Store loaded information in an LBuffer for processing.
4. Process the data:
if ($texArray[tpos]$ ==0),then this position has not been visited by $d$, and therefore remains unaffected: texArray[$tpos$] = 0.0;
else is($texArray[tpos]! = 0$),then $tpos$has been visited by $d$,
therefore, Process value at $tpos$:
TexArray[$tpos$]= $1.0/mass_d$
```

15 Design

15.1 Classes

15.1.1 Droplet

The droplet class defines a single droplet. It is represented by a particle, which consists of the classic particle attributes shown in Table(). A droplet will have a set of individual actions (per-droplet actions) that characterize its flow over the surface. Each droplet will move according to its underlying surface parameters, and therefore, will need to be linked to the surface($* - 1$ relationship). The latter may be considered as the environment that the droplet gets affected by.

15.1.2 Behaviours

Droplets that get created during a simulation form part of the same particle system and share a single surface. The Behaviors class is set to control the global actions of the droplets and manages the droplet-droplet interactions. Its parameters are listed in Table(). it may be identified as the 'flock', acting as an interface between external influences and its droplets.

Its method $forceOnDrop(Vectorf)$ applies external forces6.3.2 on the droplets at eact timestep. Thus, Behaviors is also linked to the surface. This way, a value, X, may be passed from a higher level(user end) to Behaviors which then passes it across to each droplet. More flexibility is obtained without any change to the structure, for example, X may be modified per droplet, yielding different behaviours for each droplet. For example, for every droplet, assign a velocity v,decrement v by 0.1, $v = v-0.1$.Assign v to the next droplet. Effectively, the droplets would end up moving in a trail like formation.

15.1.3 Surface

The surface class defines the underlying surface that the droplets are flowing on. Its attributes describe the type of material it is made up of, its conduct with a certain type of liquid and so on. Modelled as a 2D grid, it determines the position of each droplet according to its cells and the values stored at these cells, such as, the wetness at that position.

Algorithm 8 Trail generation algorithm

```
//define a function to read and process an image for blending
void LoadTiffRGBA($*filename$)
{
open the tiff $filename$
//procedures for loading and decoding an RGB tiff image
//now process the image
a. Define a blending factor, $alpha$.
for every pixel in the buffered image, $imgBuffer$,
Retrieve its R,G,B values using defined libtiff functions: $R$=TIFFGetR(); $G$=TIFFGetG(); $B$=TIFFGetB().
b. Apply $alpha$ to R,G and B: $tempR = R * alpha$, $tempG = G * alpha$, $tempB = B * alpha$
c. Make sure that the factored R,G,B don't go beyond 255.
if ($tempR \geq 255$)then {$tempR = 255$}. Repeat for G and B.
d. Apply ($1 - alpha$) to the image being layered (destination image) and add $tempR$, $tempG$, $tempB$ to it.
$imgBuffer[s*3] = imgBuffer[s*3] * (1 - alpha) + tempR$.Repeat for G and B.
e. Make sure that the updated layered image doesn't go beyond 255.
f. Release resources
g. Close the image.
}
//define a function to write out the blended image
void writeTiffFile(*newtiff)
{
a. Allocate memory for the image to be written: $writeImg$ =new char[$height * width * 3$]
b. Open image $newtiff$.
c. //procedures for writing to a tiff file
d. Fill the image buffer
for every pixel, $p$,
$writeImg[p] = imgBuffer[p]$
e. Close image.
f. Free buffer.
g. Set the buffer for images that get loaded back to 0: $imgBuffer[p] = 0$
}
//Now handle the frames that need to be loaded for each frame
//continued further below. Please refer to Trail generation algorithm continued.
```

Algorithm 9 Trail generation algorithm continued

```
void main()
{
1. Specify the number of frames, $framenum$, to process.
2. Specify the starting frame, $startFrame$.
3. Specify the number of previous frames, $loadNum$, that should be blended together.
4.Loop through the number of frames to process:
for(int f=$startFrame$; f < $startFrame$+$framenum$; f++)
{
if($f \leq loadNum$), then there won't be $loadNum$ amount of previous frames to load, therefore
{
5. load the previous frames $1 - f$.
for(int s=1; s<$f + 1$;s++)
*note: taking into account frame padding*
$loadName$ =$loadName$.f.tiff;
loadTiffRGBA($loadName$);
}
else if($f > loadNum$)
{
then there will be $loadNum$previous frames, therefore
6. load $loadNum$previous frame for $f$
for(int q=1; q<$loadNum + 1$; q++)
*taking into account frame padding*
$loadName$ =$loadName.(f - loadNum) + q$.tiff;
loadTiffRGBA($loadName$);
}
7. Write out the blended image $blendedImg.f.tiff$
writeTiffFile($blendedImg.f.tiff$)
}
8. Free memory:
delete [] $imgBuffer$;
}
```

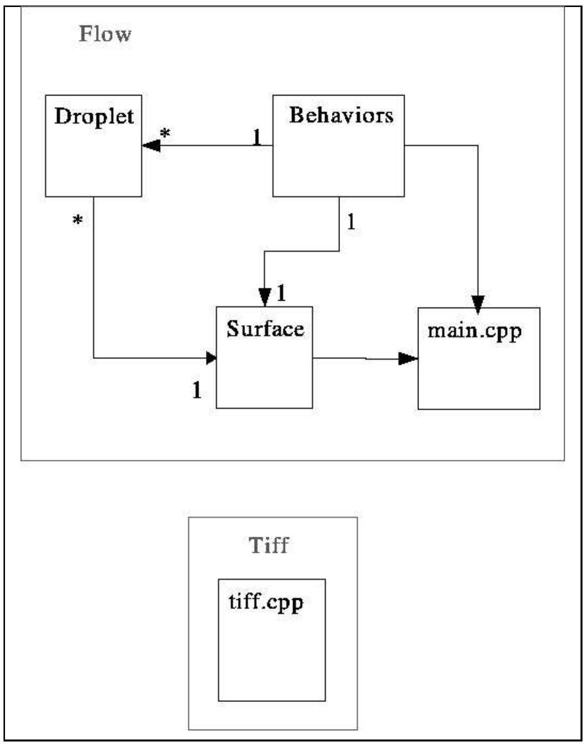


Figure 10: Class Diagram

The surface provides the interface for a low level of detail, enough so as to determine the motion of the droplets over the surface. A set of texture maps then get applied to it, bringing in details to the individual flows. Therefore, the class will contain methods to read, manipulate and write images. It is also important to note that as well as dealing with each droplet at each timestep, the surface also has to consider each of its cells and texture cells as they get updated each time. Thus, care must be taken so as not to introduce too many loops that might cause slow downs.

15.1.4 Main

Main sets the system running. It contains the numerical solver, determining the length of a timestep. A major part of the program deals with the generation of rib files that are required by the renderer. These ribs assign shaders to the surfaces of objects involved in the scene. The parameters of the shader will get updated at each timestep by methods defined here. It provided the connection between the particle system and the appearance of the droplets over the surface.

Figure 10 gives an overview of the class diagram of the system during its design stage.

16 Implementation

16.1 Parameters and Attributes

The main parameters and attributes of each class are listed in tables 1,2 and 3.

Note: These are not full parameter lists. They are the key parameters that contribute towards the flow and appearance of a dropelt.

16.1.1 Droplet Attributes

Type	Attribute	Function
GraphicsLib::Point3	Position	Position of droplet
GraphicsLib::Vector	Vel	Velocity of droplet
float	Dmass	mass of droplet
float	Radius	radius of droplet
float	maxRadius	max radius of a sphere
GraphicsLib::Vector	Force	external forces acting on droplet
GraphicsLib::Poin3	nextLPos	droplet's next position on the surface grid
int	Row, Col, Lindex	droplet's current row, column and index into surface array
int	texrow, texcol, texPos	droplet's current row, column and index into the texture array
bool	moving	droplet is either static or dynamic
Surface	*Surface	drplet's underlying surface.

Table 1: droplet attributes

16.1.2 Relationship between droplet mass and radius

If we assume that a droplet in its simplest form is a sphere, then it's volume is given by

$V = \frac{4\pi R^3}{3}$ (eq.1)

A droplet's mass is equivalent to its volume, that is, $m = V$. Hence, eq.1 relates its mass directly to its radius.

Calculating radii values from its mass for each frame would mean performing a cube root operation which is expensive. Thus, in order to avoid this computational cost, the radius is defined as a factor of the mass:

```
    Radius= DMass*0.1;
```

16.1.3 Surface attributes

Type	Attribute	Function
float	w2	parameter for remaining water
float	w1	parameter for dry surface, relating to the plane inclination
float	aff	controls affinity for water
float	up	controls the amount of splitting
float	roughness	defines surface roughness with a range 0-1
float	roughnessAttr	controls the amount of wetting exhibitted.
vector<float>	AffMap	buffer array to store affinity map
vector<float>	remainingM	array for remaining mass values

Table 2: Surface attributes

Type	Attribute	Function
vector<Droplet>	droplets	list of droplets
float	Dnum	number of droplets in the list
GraphicsLib::Vector	Gravity	gravitational force
GraphicsLib::Vector	Friction	frictional force due to surface roughness

Table 3: Behaviours attributes

Angle of inclination, φ, of the surface plane

The angle of inclination φ relates to the static critical mass, M_{sc}. In the method presented by Kaneda,[2], the speed of a droplet in motion depends on the angle φ, rather than external forces such as gravity:

v= v_0+ $a_{(i+k,j+1)}(\varphi)t$

where v_0=speed when the droplet is put on the glass plate or after it has collided with another droplet

and $a_{(i+k,j+1)}(\varphi)$= acceleration of the droplet when the surface angle of inclination is φ.

In this system, though, the droplets are subject to forces of gravity and friction, which affects the speed of each droplet.

16.1.4 Behaviors attributes

Friction, Fr, due to interfacial tensions and surface roughness

A frictional force, Fr, is included as the internal resistance force on the droplets, to counteract the gravitational force applied but with a smaller magnitude. In order to keep computation simple, these forces are kept constant, and hence the shape of the droplets remain unchanged. If not, the shape generation itself would require a simulation on its own so as to take account of the dynamic parameters of the droplet shape model.(please refer to)

Defined Parameters

angle of inc. φ	$w_1(\varphi, k)$	RoughnessAtt
90 degrees	when $k = -1$	0.25
M_{sc}	1.95	Affinity
2.5	when $k = 0$	AffMap.tif (rgb)
M_{min}	2.0	
0.2	when $k = 1$	
M_{max}	1.95	

Table 4: Initial surface parameters

16.2 Critical values

16.2.1 Self-organised criticality

Based on the purpose of the experiments performed by I.M.Janosi and V.K.Horvath, [1], the idea of self-organized criticality rests upon the assumption that the coverage of the surface has a critical value. If this amount is continuously increased, the system will organise itself in such a way that its average coverage will be the critical value by unloading excess amounts through streams.

16.2.2 Static critical mass, M_{sc}

Following from the idea presented above, a critical value, static critical mass, M_{sc}, is defined to control the motion of the droplets and its value depends on interfacial tensions and the slope of the surface. A random amount of water gets added to each cell on the grid at each timestep. If this amount exceeds the critical value M_{sc}, then the droplet starts to flow (it moves to another cell position). If not, the droplets remain stationary at that cell.

Water gets added to stationary and moving droplets. Once a droplet is in motion, this continuous addition gets counterparted by the reduction in mass caused by the amount left behind (RemainingM) due to wetting.

This effectively creates a fair balance in the system.

16.2.3 Dynamic critical mass, M_{dc}

This value controls the deceleration of a droplet until it comes to rest and is less than the the static critical mass. Factors such as friction would cause a droplet to slow down. But in a system where mass is continously added to simulate rain, for example, streams will keep on forming and therefore the overall effect of this event is negligible.

16.3 Initial state

Depending on the nature of the problem. initial values are predefined, as illustrated in Figure 11a. and 11b. show different amounts of water droplets distributed across the top of the surface to simulate rain flowing down a surface.

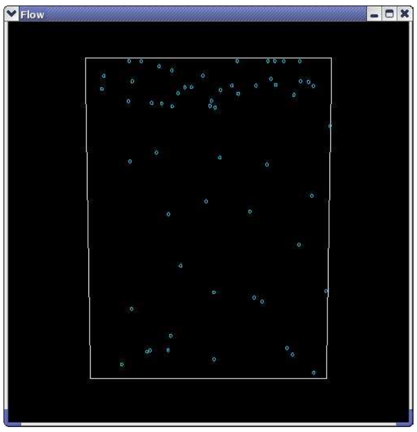
The initial surface values currently defined are given in table(initial values).

16.3.1 The affinity map

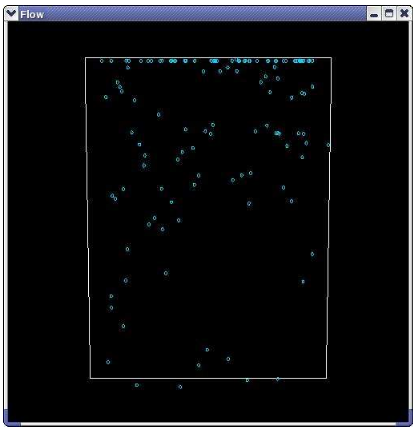
An affinity map with a 256*256 resolution is applied to the surface at initialisation for a finer distribution of affinity values. A greyscale image is chosen for the map, from which only the value from 1 channel is loaded into a buffer (since $R = G = B$). This approach saves on memory that is allocated to the buffer, that is, instead of allocating a space of 256*256*3 for each RGB channel, only 256*256 needs to be designated. The algorithm for loading in the affinity map is described in .

The texture values need to be modulated to a more suitable range before is involved in calculating the remaining mass, m'.

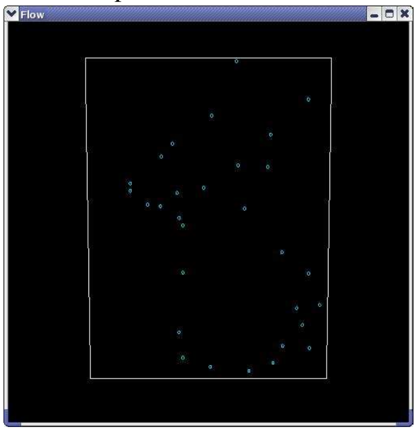
The map contains RGB pixels with values in the range of 0-255. These need to be modulated to more suitable values before being used in the calculations for m',see 9.2 . An aff factor is introduced and applied to the map value.



a. 10 droplets distributed at random



b. 100 droplets distributed across the top



c. droplets distributed at random

Figure 11: Simulating different scenarios at initialisation

```
    affinity = $aff$/texture value;
```

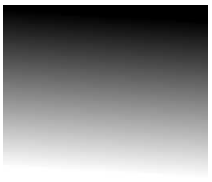
Dividing aff by the texture value yields appropriate values, that is, the lower the affinity, the slicker the surface. Consider a pixel value of 15 and aff of 5.0. Then affinity = $5.0/15 = 1/3$.

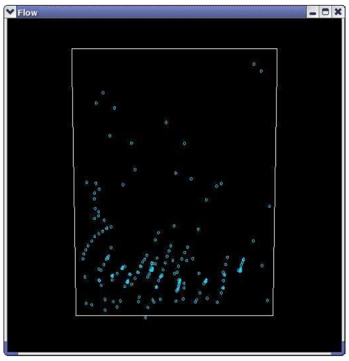
And a pixel value of 45 and aff 0f 5.0: Affnity= $5.0/45 = 1/9$.

The darker pixel value of 15 gives a higher affinity. Therefore, if a gradient was used on the map, then black -> white would mean high -> low affinity values.

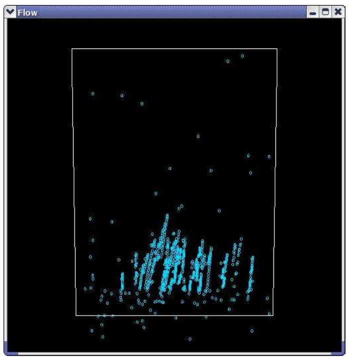
aff provides a more subtle way of controlling the affinity over the surface. By referring to a greyscale map, the higher the affinity, the darker the texture colour.

Results of tests carried out can be seen in Figure 12 .

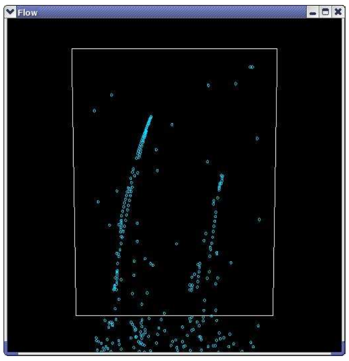

a. Affinity map used



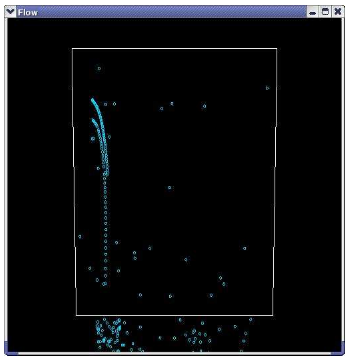
b.when $aff = 5.0$



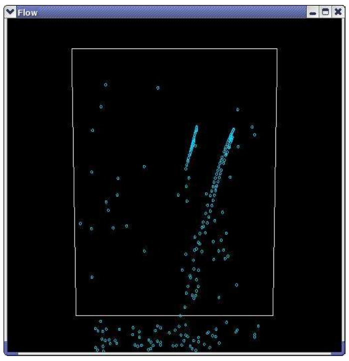
c. when $aff = 20.0$



Flow pattern with $aff = 5.0$: a.



b.



c.

0

correnponding affinity maps: for a.

A diagonal ramp for b.

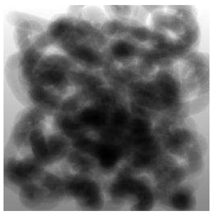

A noisy map for c.

Figure 12: Demonstrating flow influenced by affinity over the surface.

16.4 Rib generation

Several methods to generate rib files have been defined, each describing a different scene. The ribs are created by printing out a series of lines appearing in the specific format that a rib file is set in.

Depending on the scene that is being rendered, one or a conbination of these method can be enabled and the corresponding rib files created to the specified folder. The method generateRib(int n) implements a switch statement where n specifies which type of scene the rib file contains, that is, switch(1) enables generateRibRefract(), which describes a glass plane with the refraction.sl shader applied to it. Similarly, switch(2) generates rib files that use the distortDrop shader on a flat plane,. Switch(3) allows the droplets to be rendered as spheres.

The rib files are generated twice, that is, one with $alpha = 1$ and the other with $alpha = 0$ because both instances are required later on during compositing. The second instance is needed for the trail generation process. The two sets of rib files need to match and therefore

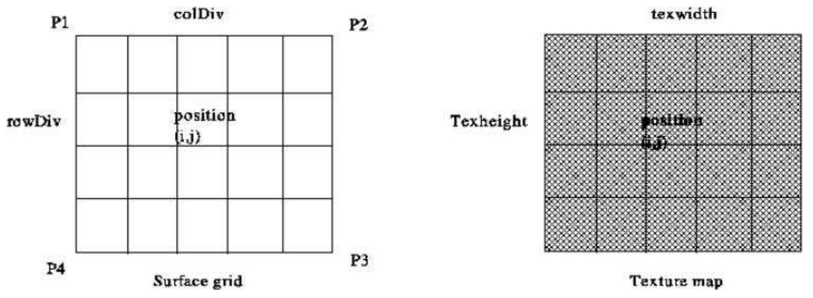


Figure 13: Position on the surface and on the texture map.

have to be generated at the same time, that is, on the same run of the simulation. Otherwise, two different flow patterns will be exhibited.

16.5 Determine the position on the grid

A droplets' s position, p_d, is a 3D point in world coordinate space. Since the surface is given as a 2D grid, $p_d$needs to be interpreted in 2D when placed on the surface. The procedures getRow() and getCol() defined in class Droplet finds its position in terms of rows and columns, and interpretes this results as an index into the position array for the surface (Lattice, please refer to **??**).

The algorithms used to determine the position of a droplet on the surface grid are listed in 3. The same technique is applied to retrieving the position on the surface grid and on the texture maps attached. The only change is the resolution, that is, for the surface, the dimensions rowDiv and colDiv are used, that is, 10x10, and if the texture position is required, the dimensions texwidth x texheight, (256x256), are used. Figure 13 shows a position on the surface grid and its corresponding location on the texture map.

Note: The positions are calculated from the point P1 on the surface plane, and in the case of the texture, from the upper left corner.

16.6 Tiff manipulation

Currently, the system is designed to handle tiff images. Before any processing can be done to an image, it must be read into a buffer that has been allocated enough memory. The library libtiff is used to to manipulate the images.

briefly go through loading and writing alg.

16.7 Integrating droplets into live footage

16.8 Interaction

By making the flow generation interactive, the user can have a level of control and produced the desired images. This may be required to produce effects such as making the streams follow a target point in order to simulate the side window inside a moving car.

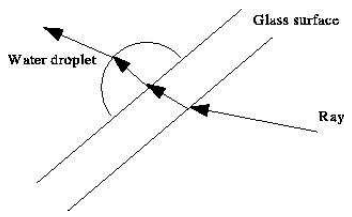


Figure 14: Light refracting into a glass surface, into a water droplet and out into air.

17 Rendering

The interaction of light with a water droplet is a complex process involving reflection and refraction. Ray tracing techniques are continuously being developed for their realistic results, for example, in [15], droplet geometry is built and each ray traced to calculate the underlying pixel color. The only major drawback from such methods, especially for real-time purposes, is the intensive computational and rendering process involved in order to attain decent outcomes.

17.1 Approach

The key factors that have led to the technique chosen for the model are the quantity, scale and shape of droplets present on the surface.

As a large number of droplets pours down a surface, depending on the surface properties, the streams join up to form a sheet of water across certain areas of the surface, especially for the case of smooth surfaces such as glass panes. A scene viewed from behind would look wet, almost as if underwater, and an accurate modelling of such a view would manage complex rendering processes involving raytracing. One aim of the project is to preserve the realistic look of droplets flowing down a surface using inexpensive and rapid rendering methods. It has been observed that the particular behavior mentioned above creates an impression, which implies that having an actual geometry for each droplet is not necessary. Instead, the technique presented applies a distortion process to approximate the droplets and their visual effects on glass, taking into account the refraction of light, its optical behavior in water and the bonds that define their shape.

17.2 The distortion model

When you look through a glass pane covered with scattered droplets, the areas that are covered by the droplets appear blurred and distorted. This is due to the refraction of light happening at the boundary of the two media involved, that is, between air and water. In this case, that is, when a glass surface is involved, the refraction of light as it passes through the glass needs to be considered as well, depending on where the viewer is(whether the front or behind). For example, if a scene is being viewed from behind a a glass surface, the light will pass into the glass, through the droplet of water, and back out into the air, as illustrated in Figure 14.

So as to allow for experimentations, two techniques have been developed:

1. The first method is closely related to the model presented in [14] and involves a partial distortion method that moves pixel coordinates around on a 2D texture image such that it mimicks refraction.

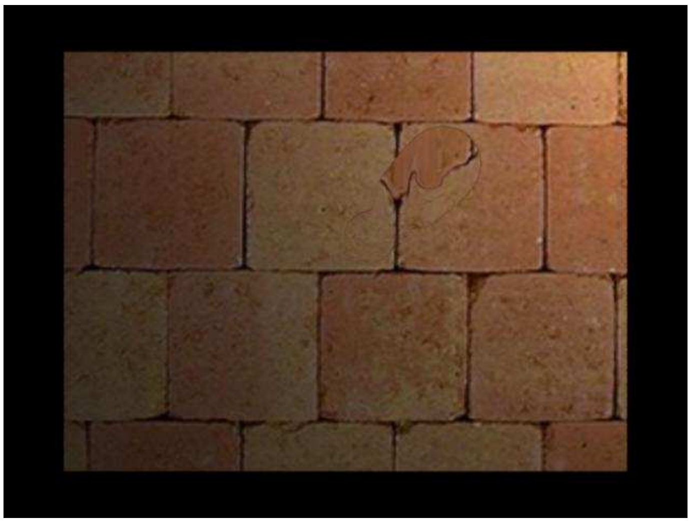


Figure 15: Test render illustrating shader with merging effect

2. The second method applies ray tracing to a sheet of glass inserted in the scene.

While both produce realistic results, their efficiency lies in the nature of the problem. For example, a scene involving a large amount of tiny droplets is better described by (2), while a scene that involves close ups of a small amount of droplets interacting with each other looked more defined by using (1).

17.3 Merging effect

Although a surface shader is defined for each method, both are based on the same principle. The desired effect is to model the droplets flowing over a surface, while interacting with each other the way they would in the real world. Previous works have illustrated realistic fluid effects achieved through the use of implicit surfaces. For example, in their paper[15] , Y.Yu, H.Jung and H.Cho employ metaballs to model merging droplets. While very realistic images are obtained, their method not only involves specifying an isosurface but also involved ray tracing, which are both time consuming and intensive processes. Instead, the approach taken in this model applies the principles of implicit surfaces[**?**] to the shaders such that they behave as though an isosurface were present. An example on a test render illustrating the merging effect achieved of 2 droplets is shown in Figure 15.

The position of each droplet represents the centre point of a blobby. The field that emanates from each blobby would represent the region that a droplet covers while the strength of the field would be represented by the radius of the droplet. By considering these 3 factors, the shape of a droplet is obtained.

Below, a template of the shaders is given, outlining the basic common parts that mostly handle the droplet shape. Sections 16.3 and 16.4 then picks out the majors aspects of each one.

The shaders take in an array of points, defining each droplet's positions for each frame. Droplets also get created or killed off at each time, therefore the number of droplets varies for each frame. Due to this, the shader makes use dynamic arrays to define the size of the position and radius arrays that are needed in order to model the isosurface effect. That is,

```
varying float rad[] = {};
varying point pnt[] = {};
```

The arrays are declared as varying since the positions and radius are constantly getting updated over the surface .

Note: although the number of droplets is changing for every frame, its value for each frame is constant throughout the shader, and therefore gets specified as a uniform float.

The strength of the field decreases gradually with distance from the centre. This effect is obtained by using the smoothstep function over the area covered by the droplet.

By looking straight down onto the flat surface, the area appears as an irregular disk.

```
//for each droplet
//calculate the distance of P from droplet pos
dist=distance(P,Pos);
//distort the radius of each droplet to get its irregular shape
radius= radius+ noise(pc)*1.5;
```

The smoothstep function is defined as $smoothstep(floatmin,max,val)$, returning 0 if $val < min$,1 if $val > max$, and performs a hermite interpolation between 0 and 1. Using it on $dist$ defines the disk, that is, 0 is returned if P is found inside the shape, and 1 if P is outside. Reversing the function, that is, $1 - smoothstep(floatmin,max,val)$,reverses the return values to 1 for $val < min$ and to 0 for $val > max$. Now 1 is returned if P is inside, if $(P < 0)$, and 0 if P is ouside the shape $(P > 1)$.

By adding up the returned value for each droplet, they are combined into 1 area (shape) over the surface for P. This method effectively draws merging droplets.

```
//get the combined values returned for each droplet
field += 1 - smoothstep(0, radius, dist);
```

The isosurface joins points sharing an equal value. By defining a variable, eq, to represent this equal value, which in this case, will be the distance from the centre position, the added gathered $field$ value can be checked. By defining eq as , it can be used in the smoothstep function to determine the merged area of close droplets. If $field > eq$, then we are inside the area, which means that we are in an area covered by a droplet, and thus should appear distorted. This is achieved by applying a distortion factor to the color, that is

```
if($field > eq$)
//define the distortion factor by an fBm function.
//give it a slightly brighter colour to simulate the highlight from water
float fbm= fBm(pshad, filterwidthp(p), 0.1,0.2,0.1)*0.1;
//distort the color this factor
```

Only distorting the area created a very hard edge and unrealistic look, as can be seen in the figure (). By softening the edge with a smoothstep function, the colour is smoothed from the edge to the centre of the droplet.

```
float smooth = 1-smoothstep(iso-0.4,iso+0.4,field);
surfcolor= mix( droplet+0.05, droplet+0.09, smooth);
```

If P is not found inside the droplet area, then it is coloured with its corresponding texture colour.

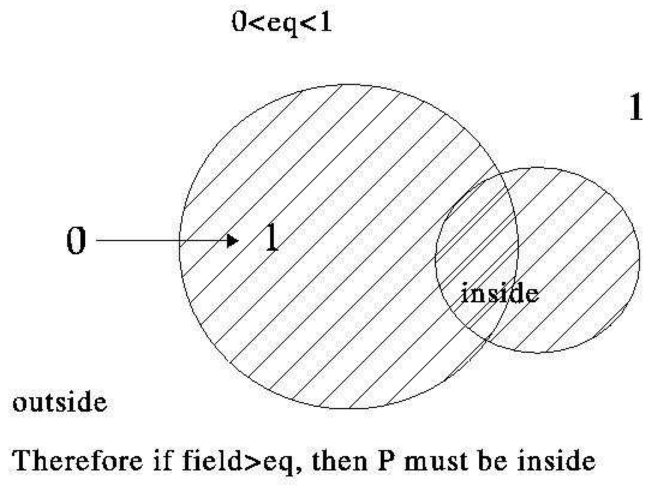


Figure 16: Smoothstep function computing merging effect

```
    else surfcolor = surface color;
```

Figure 16 explains how the smoothstep computes the merging effect more clearly.

Reversing $smoothstep(0, rad, dist)$to $1 - smoothstep(0, rad, dist)$causes 0 to be returned if $eq > rad$, that is, outside the droplet area, illustrated below, where inside is marked by the shaded area. $eq < 0$

17.3.1 Alpha parameter

An alpha parameter is defined for each shader that allows only the droplets to be visible. This is required in the trail generation process13.5.

If alpha=1, the surface is completely opaque and $Oi = surfOpacity$;

If alpha=0, then the area not covered by a droplet is made transparent. It allows more control over the droplets during the compositing stage.

17.4 The distortion Shader

The shader DistortDrop describes method(1) presented in the section above. The colour that is gets modified by the distortion factor is obtained from the texture applied over the surface the droplet are positioned. Also, the area covered by a droplet is given a specular term to simulate the way in which a layer of water reflects light with a higher specularity than the underlying surface. That is, following from above,

color droplet in area covered by droplet is given by:

```
color droplet=color texture(TexName ,s+ fbm*field/1.85,t+ fbm*field/1.85);
```

By tweaking with the fBm function, a convincing distortion has been obtained. Having more or less distortion matches the optical behaviors and thickness of the water, that is, there is more distortion where water is thicker and less on thinner water.

Image26 and Image27 on the Color Plate illustrate the droplets with different values of eq (where eq=iso on the ColorPlate).

17.5 Refraction

17.6 The refraction shader

The refraction shader achieves a more 'wet' appearance. Instead of just distorting the texture, it calculates the refracted ray through the surface. By using an index of refraction, ior,2.1 of 1.5, a glass surface is simulated. In the case of droplets on a glass surface, refraction would occur through the droplet as well, before getting refracted through the glass surface, or, if viewing from behind, then other way round, as mentioned in 17.2 and in Figure 14.

The droplets are simulated by considering a droplet to be the 'thick' layer of water that light would bend through. Thicker droplets give a more severe distortion, and hence, the refraction shader is defined such that if shaded point P is in the area covered by a droplet, then the light gets bent more towards the normal. The shader computes a smaller refractive index for this area, which produces more distortion. Figure shows some test carried out with different values of ior. As can be seen, a stronger ior defines an area around the droplet's position.

```
//define a smaller ior value if inside region of droplet
if(field>eq)
float iorS=ior/4;
//now calculate the refracted ray using renderman's ray tracing features.
vector refractRay= refract(i,nf,iorS);
```

Refract() RSL function takes the $incident$vector, $normal$vector and $ior$and calculates the transmitted ray using Snell's law, stated in 2.11.

The edges tended to be sharp where a large colour change was involved. Hence, a smoothstep has been added to take away the harshness from the edge into the droplet.

```
//smoothstep between stron and week ior
float ref= smoothstep(iorS, ior, totalfield);
//modify iorS by ref
vector refractRay= refract(i,nf,iorS*ref);
```

Each droplet has its own mass, wetness, thickness properties, displaying different optical effects , thus, the next step was to add in an individual characteristic for each. Thier mass, indicates their density, where more dense would contain more water than less dense. A texture mapping method was used, where the texture map stores the droplets' masses. These get updated at every step due to the wetting phenomenon.

a. No smoothstep between the strong and weak ior, where strong ior = ior factored by 1/2

b. No smoothstep strong ior=ior factored by1/4

c.No smoothstep strong ior=ior factored by 1/8

e. Added smoothstep

f. Added smoothstep

g. Added smoothstep

Figure 17: Varying the index of refraction

```
//store 1.0/Dmass into the texture buffer that gets written to a texture map
for each frame in update(dt) function
TexArray[pos]= 1.0/behaviors->droplets[d].DMass;
```

The texture map defined by the user is then loaded and its values modulated with TexArray[pixels], and get written out as the texture map to use for the next frame. The process is explained more clearly in17.7.

The shader retrieves the values stored by point sampling the map.

```
float distort= float texture(bendMap, s,t,s,t,s,t,s,t,"samples",1);
//now modulate the ior by this factor
float iorS=ior/4 *distort;
```

convert to tx format

One thing that needed to be considered here is the tx file format that Renderman uses for texture mapping. Since the values are updated for every frame, the texture map gets updated, therefore, each map needs to be converted to the .tx format before being applied in the shader for rendering. Therefore each map was weritten out and converted to .tx by running a shell script before rendering the rib files, see section (Shell scripts) for more detail.

Figure 17 presents some results obtained by varying the ior with and without the smoothstep function.

See Images31-36 on the Color Plate for a coloured version.

17.7 Applying the texture map

18 User control

Users are given control via a texture map that assigns affinity values across the surface the droplets are on.

The user has the capability of controlling the droplet flow by defining a texture map for the affinity surface parameter. The values, read in as chars, range from 0-255, which gets turned into a more suitable range when loading the texture map. That is,

```
AffMap.push_back(aff/(float)R);
```

where aff is parameter for affinity, acting as an alternate control to using a texturemap. It is given a default value of 5.0. Therefore, as an example,

If R=150, then the walue that gets loaded into $AffMap[]$is $5.0/150 = 0.03$.

If R=12, then the walue that gets loaded into $AffMap[]$is $5.0/12 = 0.416$.

That is, a darker colour gives a higher affinity, which will cause less splitting to occur. Where as a lighter shade computes a lower affinity which generates more splitting actions.

The user can hence control the flow as desired up to a certain degree. As a next step, the system will be integrated into a more interactive environment, where user interface issues will be paid more attention(please refer to 23).

19 Computational cost

The naive use of for loops in the mergeDroplet algorithm,14 creates some cost in processing.

At each timestep, for every droplet, d, in the list, a check is being carried out with every other droplet so as to determine its local neighbourhood. Therefore, the process is of the order $O(N^2)$. This is an issue that needs to be taken into account before the system is exposed to more processes that require a fair amount of computation. Reducing the problem to $O(N)$would improve the performance of this process. The mergeDroplet method handles the dynamics involved as 2 droplets collide by referring to the law of conservation of momentum and thus, a collision check has to be made. An alternate to this method could be to create an array on the surface (in the class $Surface$) that holds a list of droplets for each grid cell. Each cell can then be looked up, using a more efficient searching algorithm, if a droplet lies on it at that time. This would be on the low resolution grid, that is, the 10x10 grid, in order to save memory and the number of look ups that have to be made.

20 Technical problems and solutions

1. Limit of (dynamics) arrays in Renderman SL.
2. The system shows a considerable slow down when the surface grid is increased to a higher resolution, for example, 100x100 grid.
3. Decreasing the value of MaxRadius attribute to about 0.1 causes the system to slow down, as more streams, hence more droplets, get created.
4. The tiff images that are being handled in the program are continuously being loaded and closed at each frame, which can sometimes cause the system to run slower. During the early stages of generating the streams, a very important aspect, that of memory allocation, was being neglected. This caused the program to stop running after a given time, yielding the error:

```
    TIFFWriteBufferSetup: /usr/tmp/writtenTest.0072.tiff:
    No space for output buffer.
    Aborted.
```

The problem was resolved by deleting the image buffer each time, which freed memory and thus allowing for the next round. Precautions are being taken to ensure that sufficient memory is available, for example, an image being loaded within a loop is closed before the next one gets loaded. The buffer each image gets written to is reset to 0 before the next one gets written to it.

5. As each image is being loaded to build the streams, there comes a point where all that gets written out is a blank image in which all RGB pixel values are 0. It is believed that the problem might be due to a memory leak happening somewhere within the program, affecting the subsequent pixels that get written. Each step in the implementation has been repeatedly stepped through but the source of the matter remains unclear. A solution is reached that greatly reduces the problem, although not entirely resolve it, deals with the observation that only the previous 20 frames or so need to be considered and is described in 21

6. The wetness values were originally being initialised to 0.0. When these values were used to control the texture map, the program faulted after a short while, due to the fact that the calculations for generating the updated texture map divides the value of 1.0 by the updating parameter, that is, the wetness value and therefore attempting to divide by 0.0. The problem has been resolved by initializing the wetness value to a small negligible value of 0.001.

7. system slowed because of too much splitting. With black(dark) texture map, higher aff, more halting , causing more mass to be accumated and increased in radius-> resulting in more splitting.

8. When specifying the filename for the affinityMap to be loaded, there appears to be a problem when the name goes beyond a certain amount of characters. The source for the cause is still unknown. As a temporary solution, the affinity maps are being saved to a higher level directory, in order to keep the number of characters down.

21 Optimization

Stream generation algorithm

Initially, the algorithm was set to load every previous frame up to the current frame, for each frame. That method caused the system to run considerably slower and a speed up was required if the program were to handle a large number of images. Examining the written images more closely revealed that the one for the current frame, f, seems to get the most affected by its 20 or so previous frames. That is, frames $< f - 20$ are negligible. Hence by making the assumption that only the 20 previous frames matter for each frame, the loop is cut down by a considerable amount to only 20 frames, enhancing the speed and the efficiency of the algorithm.

The alpha value alters the RGB values that are being loaded and has to be set appropriately to suit that particular image. It uses a simple additive blending function and care must be taken so as to ensure that the values don't go beyond 1 and create artifacts. A more defined blending function should reduce the risk for these artifacts occuring.

21.1 Killing particles

The algorithm includes a method AddDroplet() that adds a droplet to the system on every frame. This helps in obtaining the desired effect of raindrops appearing on the glass surface. This builds up the list of droplets managed by the class Behaviors gradually and is not an immediate thread to loss in performance. The splitting process, however, is invoked depending on the surface property at that particular point on the texture map and therefore unpredictable. When the process occurs, particles get added in one go, which causes the system to slow down. The solution has been to make sure that the particles are given a lifetime when they get created. The algorithm removes particles based on their position in relation to the surface. That is, when the particles have arrived at the edge of the surface, a counter is started that decrements the particle's lifetime by one each time. When the lifetime reaches 0, the particle is removed from the list,allowing a balanced droplet list to be maintained.

```
for every droplet s in the list,
if(droplets[s].Life<=0)
{ droplets.erase(droplets.begin()+s); }
```

21.2 Adding features using texture maps

The system can be extended to include more features such as staining, 23. This effect is a precise process involving pigment colour and therefore more visually noticeable. By using a texture map to input parameters that control this behaviour, as was done with the affinity map, for example, the amount of exposure to sunlight, the effect can be achieved. The required amount of detail may be entered by defining a desired size for the texture. This method avoids the surface being divided up into smaller cells for a high resolution surface, and saves on processing power. This way, more aspects can be included while providing control to the user.

22 Animated Sequences

Several animated sequences have been produced, illustrating how various effects have been achieved and used in scenes. This section discussed how some of the results have been obtained. Snapshots may also be viewed on the Color Plate.

22.1 Tropical Rainforest

File: forest2.mp2

This short sequence gives the view of a rain forest from behind a glass pane on a rainy day. The scene is composed of a backplate of the forest image and a flat plane surface onto which the refraction shader17.6 has been attached. A simple diagonal gradient from dark grey to white has been used as the affinity map. An *alphablend* of 0.07 has been used to generate the trails left. When the trails were composited in, they were given a slightly higher gamma value, so as to give the warm humid atmosphere of a rainforest.

22.2 The driveway

File: InFootage.mp2

The driveway is a little sequence that demonstrates how the system has been integrated into live footage. The scene has been shot outside and shows an 'ordinary' neighbourhood. Droplets flowing down a glass pane are then added so as to produce the effect of looking out of a window from inside a building while it is raining. Each frame in the sequence has been used as texture to a background plane. A slightly lighter gradient to the one used in Section22.1 has been applied in order to create thin wet films of water flow down the gass. The footage has been obtained from Kader.

22.3 Suspicious Droplets

File: ColouredDroplets.mp2

The refraction shader has been modified such that the final colour at the point being shaded, P, found inside a droplet region is given an overall colour. This produces effective results depicting instances of blood or ink. The animation illustrates red coloured droplets splatted onto a glass placed in the rainforest scene of Section22.1. Affmap5.tif has been used as the affinity map.

22.4 London, Italy, New York

Files: Backplate1.mp2, Backplate2.mp2, Italy.mp2

Various sequences have been produced so as to give an example of using the droplets to compliment a backplate in a scene. The droplets in backplate2, *New York,* have been created using the affinity map affmap4.tif whereas those in *the Italy* and *London* sequences use affmap5.tif. The trails generated in *Italy* have been made slighly darker by giving them a brightness value below 1.0, in order to cut the radiance from the large sky portion. Meanwhile, the subtleness of each result makes it a good component to a backplate.

22.5 Fishtank

File: Fishtank.mp2

This animation produced the effect of droplets running down the side of a glass fish tank. As a test, the image of an underwater scene has been used for the texture to the background plane. Although the image is static, the display obtained is quite effective, and work is in progress so as to obtain an animated sequence of an actual aquarium and integrate the droplets onto the glass pane.

22.6 Wind

File: WithWind.mp2

In order to create the wind effect, the target boolean in the system has been set to true, with a vector of (25.0, -11.0, 0.0) specified. The droplets flow across the surface. Improvements can be made on the shape of the trails generated,that is, streak them in the direction of the wind vector.

23 Further work

23.1 Staining and splashback

Staining has been addressed in the paper presented by J.Dorsey, H.K.Pedersen and P.Hanrahan [10] to simulate how water picks up colour from one point and deposits it at another point further along the flow. This phenomenon is an expected effect when cases involving dirty, absorbent, or weathering surfaces are being considered and hence, has become a goal for the future of this project . It can be achieved by developing another texture map that gets attached to the surface and updated by the colour gathered from the region on the surface where a droplet previously visited during its journey. By using this approach, different materials may be simulated, for example, copper, watercolored paper, and rusty metals.

Another effect described in [10] is splashback, which occurs where a wall meets the ground. At the moment, the system addresses the issue of what happens when the droplets have reached the end of the surface by simply removing the external force caused by the surface properties, and keeping gravity activated. Splashback presents the logical behavior of droplets on meeting the ground, where interaction with dirt particles are taken into account and would add a valuable extension to the system.

23.2 Curves and more

So far, only flat planar surfaces have been considered. It would be interesting to see droplets meandering over a curved irregular surface. Hence, as a next step, the system will be extended to include curves. One way in which this can be achieved is by constraining the droplets to the surface and calculating the slope of the surface at each droplet's position, at each timestep.

23.3 Affinity map

Affinity is dependent on the surface material. A list of materials may be produced with predefined values and texture maps simulating various types of materials and instances, which would make it simpler for the animator.

For example, in the case of glass, it has been observed that when surface aff attribute**??** is >5.0, and anAffmap with grey-white values is used, the effect of flow on a glass surface has been achieved.

23.4 Vector fields

To make more interesting pattern for the flow, vector fields have been considered. This is a common technique that is used to generate flow. By assigning a vector to each point n the texture map and specifying a certain magnitude and direction, smooth flows may be obtained. Software, for example, Maya, describes its fluid simulation by allowing the vector fields to be displayed for each cell. This helps in bringing control to the flow. Thus the flow may vary from normal to completly chaotic, depending on what is being modelled. Taking a it a step further, turbulent flow

23.5 User control

So far, the system presents no GUI for the user. This is because work was mostly being done on the dynamics of the droplets and their behaviors . As a next step, GUI may be implemented that will allow the user to alter parameters and enter texture images via a more user friendly interface.

23.6 Directory system

At the moment, the pathnames to the files to be loaded or generated are being specified directly into the code. This is quite weak and not a very efficient way of managing directories and risks running into problems in the future. Hence, a stronger directory system needs to set up, especially at this early stage, so as to maintain a strong backbone if the system is integrate onto different platforms.

24 Evaluation and Conclusion

A system that simulates the flow of water droplets over a flat surface has been described. The flow is generated from the underlying surface properties and the appearance of the droplets is achieved via a surface shader. The surface paramaters that have played a role are mainly:

a. The affinity for water

b. The surface roughness

c. The wetting phenomenon that the surface exhibits.

The affinity values across the surface are assigned via a texture map and hence an wide variety of complex flow patterns may be produced, while maintaining a realistic appeal.

The droplet attributes that have contributed are:

a. Its position

b. Its mass

These have been used to update texture map values for use by the shader model.

The system is simple, performs well and contains a good pipeline. One thing to remember though is the flow of the pipeline. The processes should follow a specific order so as to achieve the desired results.

As mentionned in23.5, the system may be embedded into an interactive environment, giving the user control over the flow, which is important if the system is to be used for artistic purposes.

The merging and splitting algorithms may be improved by finding an alternate to using for loops.

The shader model provides a rapid and efficient way of producing the effects of merging water droplets, and also works well with refraction.

A disadvantage to the system is that a suitable value for $alphablend$ has to be provided. Although a value of 0.05 gives the required results, a more accurate one relies on trial and error.

In conclusion, the system has been able to deliver the desired effects. A close up look at a few droplets on a glass pane reveals more work that can be included to the model for improvements and extending research in this area of computer graphics.

25 Acknowledgements

Thanks to Jon Macey, Ian Stephenson and John Vince for their valuable help and assistance. Also thanks to Bruce Steel, Pete Riley and Anders Hast for their useful suggestions on the development of the system. Thanks to Jim for his great help, and Kader for providing some live footage sequence.

26 AppendixA

User Manual

The system requires the correct directories to be set up in order to handle files that get manipulated during the processes.

A shell script has been written that creates the directories needed.

1. Place the script in the directory where the program will be run.

2. Change the rights to the script: $chmod775setDir.sh$

3. Run the script.

Five new directories should have been created:

Droplet - where the generated rib files will be placed for render.

DropletA - where the generated rib files in which the 'alpha' parameter in shader is 0 will be placed for render.

Stream - where the images output from 'Tiff program' are placed.

TextureMap - where the updated texture maps for each frame get written to.

AffMap - where the textures that have will be used for the affinity map will be placed.

The files provided must be put into the correct directory.

4. Place the files refraction.sl, refraction.slo, distortDrop.sl, distortdrop.slo, filterwidth.h, noises.h, patterns.h, Render.sh, textureName.tx in both directories Droplet and DropletA.

5. Place the texture map to be updated in the directory Texturemap. This can be a user defined image but must be and RGB image with the name '$filename$.0001.tif'.

6. Place any affinity maps created in the directory AffMap.

The next step is to make sure that the path names are set up correctly where necessary.

7. In main.cpp, include in the string variables 'droplet' and 'dropletA', the full path to the directory from which $setDir.sh$ has been run.

for example, if the script has been run from the home directory, then add the full pathname such that

string $droplet$= "/masters/sjogoo/Droplet/droplet"

string $dropletA$= "/masters/sjogoo/DropletA/dropletA"

Note: At this stage of development, the pathname for the affinity map has been hard coded into Surface.cpp in its constructor. Thus, please make sure that the full pathname to the chosen affinity map is specified in the Surface constructor method.

This should have the pathnames set up where necessary.

8. Type Flow to run the program.

In the directory Droplet, the rib files $droplet.000f.rib$, where f=framenumber, should have been generated for the number of frames run.

In the directory DropletA, the rib files $dropletA.000f.rib$, should have been generated. The rendered images from these rib files are used for the trail generation.

Rendering:

9. Go to the Droplet directory and render the rib files by running Render.sh. This script essentially finds all the rib files in the folder and renders them off. For each rib file, a tiff image named $show.000f.tiff$ should have been created for the corresponding rib file $droplet.000f.rib$ for frame f.

10. Next the the rib files in DropletA need to be rendered. Following the same step as in 9. above, the rendered images $showA.000f.tiff$ should have been created for the corresponding rib file $dropletA.000f.rib$.

11. Tiff.cpp is used to generate the trails. Go to the directory where tiff.cpp is being stored (note: it should not be placed in Droplet or DropletA directory). The trail can be created by loading in the tiff images generated, $showA.000f.tiff$. Once again the pathname to $showA.000f.tiff$ needs to be specified for every *LoadTiffName*< < present, that is,

LoadTiffName<<$showA.000f.tiff$

The pathname to the directory where the written images will be stored in should also be specified for every *newtiff*< < present

for example *newtiff*< < *"pathname/writehere.00f.tiff";*

When the path names have been set up properly, run the program:

12. Type Tiff to run the program.

This should start writing out the layered tiff images. The message '*written pathname/writehere.000f.tiff'* should be printed out for every tiff image written.

The default values:

framenum=150; The program performs well up to about 190-200 for framenum, but 150 is the safer option.

startframe=1; This specifies the frame number on which to start generating the trails. For example, if a sequence only requires trails from frames 10-100, startframe can be set to 10.

LoadNnum=10; That is, the 10 previous frames are loaded for each frame. values of up to 20, 25 may be entered, but produce similar results as to when using 10. A value of 10 instead of 20 will also cut down on the computation process, ans therefore, has been set as the default value.

References

[1] I.M.Janosi and V.K.Horvath. Dynamics of water droplets on a window pane. Physical Review A(General Physics), 40(9):5232-5237, November, 1989.

[2] K.Kaneda, T.Kagawa and H.Yamashita. Animation of water droplets on a glass plate. Proc.Computer Animation'93, pp.177-189, (1993).

[3] Ksvltd glossary at http://www.ksvltd.com/content/index/applicationnotes

[4] P.Fournier, A.Habibi, P.Poulin. Simulating the flow of liquid droplets. Proceedings of Graphics Interface 98, pp.133-142, 1998.

[5] D.Quere. Fakir Droplets, Surface Chemistry, Vol1, September 2002, www.nature.com/naturematerials.

[6] M.Jonsson, A.Hast. Animation of water droplet flow on structured surfaces.

[7] C.Schaschke. Fluid Mechanics, Worked examples for engineers, Institure of chemical engineers, p79-81, 1998.

[8] S.B.G O'Brien, L.W.Schwartz. Theory and modelling of thin film flows, Encyclopedia of surface and colloid science, 2002.

[9] An introduction to computational fluid dynamics, p54- characteristics of simple turbulent flows.

[10] J.Dorsey, H.K.Pedersen, P.Hanrahan. Flow and changes in appearance, SIGGRAPH'96, pp.411-420, 1996.

[11] W.T.Reeves. Particle-Systems- A technique for modelling a class of fuzzy objects, Vol2,No2, April 1983.

[12] G.Miller and A.Pearce. Globular dynamics: a connected particle system for animating viscous fluids. Comp. and Graphics Vol.13,No.3,pp305-309, 1989.

[13] C.W.Reynolds. Flocks, herds and schools: a distributed behavioral model. Computer Graphics, Vol.21, No.4, July 1987, ACM.

[14] Y.Yang, C.Zhu, H.Zhang. Real-time simulation: water droplets on glass windows, IEEE

[15] Y.J.Yu, H.Y.Jung, H.G.Cho. A new rendering technique using metaball in the gravitational field. Proc. of WSCG '98, pp.432-439. Febuary, 1998. Pizen, Czech.

[16] Energy misdefined in physics, Section 5. Collision analysis. http://nov55.com/coll.html

[17] http://www.extension.umn.edu/distribution/youthdevelopment/components/0328-02.html, University of Minesota, our world of water, physical properties of water, section 2.

[18] Rick Parent, Computer animation algorithms and techniques,Morgan Kaufmann Publishers,2002.

[[27] key-27Blobby Effects: http://fundza.com/rman_shaders/blobbies/blobbies.html

[19] N.Foster and D.Metaxas. Realistic animation of liquids. In Proceedings of Graphics Interface 1996, p 204-212. August 1996.