UNIVERSITÉ DE LIMOGES

ÉCOLE DOCTORALE « Science, Technologie, Santé »

FACULTÉ DES SCIENCES ET TECHNIQUES

Thèse N° 42-2007

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline / Spécialité : Informatique

présentée et soutenue par

Pierre ROUSSEAU

le 12 octobre 2007

Simulation réaliste de pluie en temps-réel

Thèse dirigée par Djamchid Ghazanfarpour,

Co-encadrée par Vincent Jolivet

JURY

Rapporteurs :	
M. Kadi BOUATOUCH	Professeur à l'Université de Rennes 1
M. Mateu SBERT	Professeur à l'Universitat de Girona
Examinateurs :	
M. Roland BREMOND	Chargé de Recherche au LCPC
M. Djamchid GHAZANFARPOUR	Professeur à l'Université de Limoges
M. Dimitri PLEMENOS	Professeur émérite à l'Université de Limoges
M. Vincent JOLIVET	Maître de conférences à l'Université de Limoges

Remerciements

En premier lieu, je tiens à remercier messieurs Kadi Bouatouch et Mateu Sbert, ainsi que monsieur Roland Brémond pour avoir accepté d'examiner mes travaux malgré leur emploi du temps chargé.

Je remercie mon directeur de thèse, Djamchid Ghazanfarpour, pour la confiance qu'il m'a accordé au cours de ces trois années. Merci également à Vincent Jolivet pour avoir encadré ces travaux. Ce fût un plaisir et une chance de travailler avec toi.

Je remercie Dimitri Plemenos, qui de mes balbutiements informatiques à aujourd'hui, a contribué à me faire prendre le chemin de la recherche.

Un remerciement particulier va à Alexandre. Ces trois années à partager le même bureau resteront inoubliables. Je n'oublie pas tous les autres doctorants et amis que j'ai eu le bonheur de côtoyer pendant mes années jidéennes : Aurélien, Sylvain, Céline, Yann et tous les autres, merci pour tout. Je remercie également les trois autres piliers de Jidé, Hubert, Suzy et Laurent, pour leur amitié.

Je ne saurai assez témoigner de l'ampleur de mon affection envers Élodie, Julien, Joël, Mathieu, Nicolas, Jeff, et Olivier, ainsi que Sonia, Christelle et Mélanie pour leur amitié incomparable depuis tant d'années.

Je remercie chaleureusement messieurs Steve Streeting et Wladimir J. van der Laan pour leur grande gentillesse, et avec eux toute l'équipe du moteur Ogre3D, sans qui mes travaux n'auraient probablement pas atteint ces résultats.

Je terminerai ces remerciements par une pensée pour ma famille, toujours présente derrière moi, et en particulier pour mon grand-père, dont j'ai beaucoup appris, et ai encore beaucoup à apprendre.

Résumé

La simulation de phénomènes naturels occupe une place importante en synthèse d'images. Elle permet d'améliorer le réalisme des scènes représentées, en rappelant à l'observateur des éléments de son environnement quotidien. Parmi ces phénomènes, la pluie est l'un des plus courant dans la nature ; peu d'études y avaient pourtant été consacrées avant une époque récente. Dans cette thèse, nous étudions à la fois le rendu des gouttes d'eau, et leur animation. Nous proposons des modèles adaptés aux problématiques des jeux vidéos et des simulateurs de conduite, permettant une simulation visuellement réaliste de la pluie en temps-réel.

Nous présentons dans un premier temps les travaux existants en synthèse d'images, portant sur le rendu des gouttes d'eau et sur l'animation de particules sur carte graphique. Dans le second chapitre, nous présentons les propriétés physiques des gouttes d'eau, sur lesquelles nous avons basé nos travaux. Le troisième chapitre introduit le modèle de rendu des gouttes que nous avons proposé, permettant une simulation réaliste du phénomène de réfraction observable au travers des gouttes d'eau. Des extensions à ce modèle, permettant de simuler l'impression visuelle liée au phénomène de persistance rétinienne, et le changement d'apparence des gouttes de pluie en présence de sources de lumière, sont également présentées. Le quatrième chapitre introduit le modèle de simulation de la dynamique des gouttes que nous avons développé. Les particules, animées par une approche *GPGPU*, rebondissent en cas de collision avec un objet présent dans la scène observée. Grâce à une interface spécifique, l'utilisateur a la possibilité de modéliser intuitivement un vent pouvant modifier les trajectoires suivies par les particules. Les résultats que nous avons obtenus sont présentés dans le cinquième chapitre.

Mots-clés : synthèse d'images, rendu réaliste, rendu temps-réel, animation de particules, phénomènes naturels, pluie.

Abstract

Natural phenomena simulation is an important topic in computer graphics. It helps improving realism of displayed scenes, reminding the observer of elements from his daily environment. Amongst these phenomena, rain is one of the most frequent ; few studies had nevertheless been led upon it until recently. In this thesis, we study water-drops rendering, as well as their animation. We propose models enabling visually realistic real-time simulation of rainfalls, which are adapted to the needs of video-games and driving simulators.

We first describe the previous works in computer graphics dealing with water-drop rendering, and GPU-based particle animation. In the second chapter, we present the physical properties of raindrops, upon which we built our models. The third chapter introduces the raindrop rendering model we proposed for realistic simulation of the refraction phenomenon which can be observed in raindrops. Extensions to this model are then presented, to simulate the visual phenomenon of retinal persistance, and the change in raindrops' appearance due to illumination from light sources. The fourth chapter introduces the dynamics simulation model we developped. Particles are animated through a *GPGPU* approach, and bounce when they collide with objects in the scene. Using a dedicated interface, the user can intuitively design wind which alters particles' motion. The results we obtained are presented in the fifth chapter.

Keywords : computer graphics, realistic rendering, real-time rendering, particle animation, natural phenomena, rain.

Table des matières

Table des figures

Liste des tableaux

C'était une *bonne* tempête. Elle projetait son feu intérieur, elle s'exprimait avec passion, et les critiques le reconnurent : Pour peu qu'elle apprenne à mieux maîtriser son tonnerre, ce serait, d'ici quelques années, une tempête à suivre.

Trois sœurcières

Terry Pratchett

Introduction

Qu'est-ce que la pluie ?

Qu'est-ce que la pluie ? Derrière cette question en apparence triviale, se cache un phénomène plus complexe qu'il n'y paraît.

La définition littérale est la suivante : « *Eau qui tombe en gouttes des nuages sur la terre* » (d'après le dictionnaire *Robert*).

Bien évidemment, cette définition ne suffit pas à résumer ce qu'est la pluie. Elle peut prendre les formes les plus diverses dans ses manifestations, de la bruine bretonne à la mousson asiatique, et être pour l'Homme associée à une large gamme d'émotions. De la mélancolie romantique des pluies d'automne observées derrière une vitre, au soulagement en période de sécheresse, en passant par tous les présages auxquels la lie la sagesse populaire, elle occupe une place de premier ordre dans les préoccupations humaines.

La pluie implique de nombreuses manifestations physiques perceptibles. On évoquera en premier lieu les gouttes elles-mêmes, visibles sous la forme de traînées claires. Les impacts au sol génèrent des éclaboussures, ainsi qu'un son caractéristique dont la magnitude est source d'information sur l'importance de la précipitation. Au fur à mesure, des flaques se forment sur les surfaces horizontales, et amplifient l'importance des éclaboussures. On peut observer sur ces flaques des ondes qui se propagent en cercles concentriques, mettant en évidence les impacts des gouttes. Le degré d'hygrométrie de l'atmosphère évolue, les surfaces se mouillent et changent d'aspect, et c'est toute la perception de son environnement par l'observateur qui se trouve modifiée.

On peut s'interroger sur la façon dont l'Homme perçoit la pluie. Dans le cas

d'une pluie fine, l'information « il pleut » n'est pas uniquement visuelle. Ainsi est-il fréquent de sortir la main de sous son parapluie, pour chercher le contact d'une goutte. L'œil seul n'est pas toujours capable de déterminer s'il pleut ou non. Pour une pluie très fine, on cherchera davantage à voir les impacts dans les flaques au sol que les gouttes elles-mêmes. Outre l'information visuelle et tactile, l'information sonore revêt également une importance majeure dans la perception de la pluie.

Dans un jeu vidéo ou un simulateur de conduite, la perception tactile est très restreinte (dans le meilleur des cas, un dispositif de retour de force), la perception sonore est en général imprécise, et même la perception visuelle est limitée aux zones de l'espace que l'écran couvre. Pourtant, même une pluie légère doit rester perceptible.

L'industrie du cinéma est également sensible à ces problèmes de perception, et une technique largement répandue consiste à pulvériser un liquide coloré en blanc sur les acteurs dans les scènes censées se dérouler sous la pluie. Cela a pour effet de renforcer la partie visuelle de la perception de la pluie en rendant les gouttes plus visibles, pour pallier au manque naturel de réalisme des autres composantes de cette perception. L'illusion est en général satisfaisante, et ne choque pas le spectateur.

Pourquoi représenter la pluie ?

L'informatique graphique est une science jeune. Depuis ses premiers balbutiements jusqu'à aujourd'hui, seules quelques dizaines d'années se sont écoulées, mais le développement des techniques a été extrêmement rapide. Il y a une vingtaine d'année, on exhibait fièrement à la télévision des images très pixellisées, utilisant une palette de couleurs très réduite ; à l'époque, ces images étaient associées à une notion de modernité, alors que l'utilisation d'ordinateurs n'était encore que très peu répandue. Aujourd'hui, ces mêmes images paraissent dépassées voire archaïques ; on cherche au contraire à faire oublier la nature synthétique des images, et à les intégrer de la manière la plus transparente possible dans leur contexte.

L'avènement des périphériques Haute Définition va dans ce sens : faire en sorte que les images apparaissent aussi réelles que possible. Leur objectif est d'immerger l'observateur dans le monde représenté, pour qu'il ne fasse plus la distinction entre

son environnement réel et les images qu'il regarde. Dans le cas d'images filmées, l'immersion se fait avant tout par le matériel de diffusion. Un grand écran Haute Définition et un système sonore bien étudié, disposé tout autour de l'observateur, permettent généralement une bonne immersion. Lorsque les images sont générées par ordinateur, elles doivent apparaître le plus plausibles possible et donner l'impression d'avoir été filmées. L'objectif principal des images de synthèse intégrées dans des films, ainsi que dans de nombreux jeux (en particulier sur les consoles récentes) est désormais le *photo-réalisme*. Pour atteindre cet objectif, les images cherchent également à rappeler à l'observateur des éléments de son environnement, afin de sembler visuellement réalistes par leurs qualités techniques (ombres, définition des modèles, particules...) mais également par leur composition, c'est-à-dire par le type d'objets ou de phénomènes qui les composent.

C'est dans cette optique que s'inscrit la simulation de phénomènes naturels en synthèse d'images : rendre les images crédibles en intégrant des phénomènes présents dans la vie quotidienne de l'observateur. Ceci explique pourquoi la simulation de phénomènes naturels en synthèse d'images est devenue un domaine de recherche très actif depuis quelques années.

Dans les jeux vidéos

Si elle est inutile dans « Tetris », la pluie s'avère indispensable dans un jeu où le joueur doit se placer dans la peau d'un personnage virtuel évoluant en univers extérieur. Le jeu apparaîtra inévitablement moins crédible s'il fait un temps magnifique et constant tout au long de la vie virtuelle du personnage que le joueur incarne.

Jusqu'à une époque récente, la rapidité d'exécution d'une application tridimensionnelle temps-réel était généralement prioritaire par rapport au réalisme de l'application. Aujourd'hui, les possibilités techniques des cartes graphiques rendent ces deux points de plus en plus compatibles. Les applications temps-réel se cherchent de nouveaux objectifs, qu'elles peuvent satisfaire sans sacrifier leurs performances : un rendu photo-réaliste, le respect des lois de la physique, la gestion d'un grand nombre de phénomènes naturels. Du moment que les contraintes relatives au temps d'exécution sont satisfaites, toutes sortes d'effets peuvent être ajoutés aux applications

temps-réel.

Dans les simulateurs

L'intérêt de représenter la pluie en synthèse d'images n'est pas seulement ludique. Des applications comme les simulateurs de conduite n'ont pas les mêmes objectifs que les jeux vidéos. Le réalisme recherché se place plus sur un plan perceptif que visuel. La manière dont un environnement est perçu importe plus que son apparence. Ces applications s'intéressent davantage à l'impact de la pluie sur la perception sensorielle humaine, qu'aux gouttes elles-mêmes. Les questions posées ici sont de savoir quelle sera l'altération de la perception visuelle d'un conducteur (perte de profondeur de champ, reflets dérangeants sur les surfaces mouillées, baisse de luminosité, projections de particules, mouvement parasite des essuie-glaces dans le champ de vision...). Ces applications s'intéresseront également aux changements de comportement du véhicule (perte d'adhérence, aquaplanage, freinage...).

Cette nécessité de réalisme perceptif revêt un aspect particulièrement important dans le cadre de simulateurs dédiés à la sécurité routière. L'enjeu est alors d'estimer les impacts sur le comportement du conducteur engendrés par les perturbations météorologiques, et d'en tirer des enseignements permettant d'améliorer les processus de conception des véhicules et équipements routiers, dans le but de renforcer la sécurité des usagers de la route.

Autres phénomènes météorologiques en synthèse d'images

La simulation de phénomènes naturels, et en particulier météorologiques, est un domaine de recherche très actif en synthèse d'images.

L'utilisation de brouillard est courante en informatique graphique depuis de nombreuses années, y compris par le biais d'une accélération graphique complète. Venceslas Biri s'est intéressé [Bir03] à la représentation du brouillard à l'aide de milieux participants. Éric Dumont s'est attaché au cours de sa thèse [Dum02] à caractéri-

ser les propriétés micro-physiques et optiques du brouillard ; il a ensuite décrit les effets du brouillard sur la perception, en vue d'intégrer ses résultats dans un simulateur de conduite. Dorota Zdrojewska [Zdr04] a présenté une méthode pour générer du brouillard non homogène en utilisant un bruit de Perlin. Cette méthode utilise les possibilités actuelles des cartes graphiques pour générer des images de qualité en temps-réel. D'autre part, une implantation matériellement accélérée d'effets de brouillard homogène est disponible sur les cartes graphiques ; cette méthode empirique permet d'améliorer la rapidité d'affichage, en limitant la profondeur observable dans la scène.

L'ajout de neige dans une scène a également fait l'objet d'études. Les flocons de neige peuvent être rendus de manière satisfaisante dans le cas de scènes neigeuses par des matériaux simples, opaques et diffus. De ce fait, une chute de neige sera représentée réalistement par un système de particules simple [LZK+04]. Nishita *et al.* [NIDN97] ont proposé en 1997 un modèle de neige utilisant des *metaballs*, appliqué non pas à la neige en train de tomber, mais plutôt au sol, ou éjectée au passage d'un skieur. Certains travaux se placent dans un cadre plus général, et permettent indifféremment d'animer et de visualiser de la neige ou de la pluie. C'est le cas de [WW04] et [LZK+04], travaux sur lesquels nous reviendrons en section 1.2.

Enfin, de nombreux travaux ont été réalisés sur les nuages. Lorsque le point de vue de l'observateur est très éloigné (en particulier au sol), une simple texture (éventuellement animée) peut suffire, mais, par exemple, dans le cadre de simulateurs de vol, un modèle plus évolué se révèle vite nécessaire. Parmi ces travaux, on peut citer ceux de Mark Harris [HBSL03], qui, au cours de sa thèse [Har03], a développé plusieurs systèmes de rendu de nuage. Dans ces méthodes, des rectangles texturés (*imposteurs*, ou *billboards* dans la littérature anglo-saxonne) sont générés à partir de grilles de voxels englobant les nuages. La création de ces imposteurs est répartie sur plusieurs phases de rendu successives. Un critère d'erreur (éloignement par rapport au point de vue pour lequel a été généré l'imposteur) permet de savoir quand un imposteur doit être re-généré. Dans le cas où un objet (par exemple un avion) est inclus dans un nuage, deux imposteurs sont utilisés, le premier placé devant l'objet par rapport au point de vue de l'observateur, et le second derrière. Ceci permet d'éviter

que les imposteurs, en intersectant la géométrie de l'objet, ne deviennent visibles et provoquent des artefacts indésirables. La génération de ces imposteurs prend en compte la propagation de la lumière à travers le nuage, grâce à une technique désignée sous le terme « multiple forward scattering » par son auteur. D'autres travaux importants ont été conduits sur la synthèse réaliste de nuages, entre autre par Neyret *et al.* [BN04] [Ney00], ou encore par Dobashi *et al.* [DKY+00].

Cependant, et bien qu'étant certainement le phénomène météorologique le plus courant, la pluie manque encore souvent de réalisme dans les applications temps-réel, et seule un nombre restreint d'études y avaient été consacrées avant 2003.

Objectif de la thèse

Nos travaux ont été partiellement financés par le projet Européen *GameTools*, qui vise à développer des bibliothèques logicielles adaptées aux dernières innovations matérielles du monde du jeu vidéo. Les modèles proposés devaient être utilisables dans le contexte des jeux vidéos et des simulateurs de conduite.

L'objectif principal de cette thèse est de proposer de nouvelles méthodes pour simuler la pluie dans des applications temps-réel, en se basant sur les propriétés physiques des gouttes. Le cadre est donc double, regroupant à la fois des travaux au niveau du rendu de l'apparence des gouttes, et d'autres relatifs à la simulation de leur dynamique.

En nous appuyant sur les propriétés optiques des gouttes de pluie, nous avons cherché à établir un modèle représentant fidèlement les phénomènes observables à l'intérieur des gouttes, et en particulier la réfraction. Nous avons pris en compte la forme adoptée par les gouttes au cours de leur chute, ainsi que les déformations périodiques que suivent les gouttes sous la pression de l'air. Nous avons également étudié l'influence du phénomène de persistance rétinienne sur l'apparence des gouttes, ainsi que l'illumination des gouttes par des sources lumineuses.

Dans un deuxième temps, nous nous sommes attachés à définir un modèle du comportement dynamique des gouttes. Nous avons développé un système de particules animé sur la carte graphique par le biais d'une approche *GPGPU (General-Purpose*

Computation Using Graphics Hardware). Dans cette approche, les capacités de calcul des cartes graphiques sont déviées de leur usage naturel, la création d'images, pour devenir des processeurs de calcul parallèle puissants utilisables pour des applications autres, en particulier des simulations physiques. Notre système de particules utilise des textures dans lesquelles sont stockées les positions des particules, et qui sont animées en tenant compte d'un vent modifiable soufflant sur la scène, ainsi que des collisions entre les gouttes et les objets composant la scène.

Organisation du mémoire

Une fois arrivé au terme de cette introduction, la première partie de ce mémoire détaille les travaux existants en synthèse d'images sur la simulation de pluie. Les travaux ayant trait au rendu des gouttes sont présentés en premier, suivis des travaux évoquant leur dynamique, en particulier sur la carte graphique.

Le chapitre 2 présente les propriétés physiques des gouttes de pluie, tant sur un plan optique que dynamique.

Le chapitre 3 décrit la méthode de rendu des gouttes de pluie que nous avons élaborée, qui permet une simulation basée sur les lois de la physique de la réfraction des images à l'intérieur des gouttes. Deux extensions que nous avons apportées à cette méthode sont ensuite présentées. Ces extensions permettent de simuler l'illumination des gouttes d'eau par des sources lumineuses, ainsi que le changement d'apparence des gouttes lié au phénomène de persistance rétinienne.

Dans le chapitre 4, nous explicitons les modifications que nous avons apportées aux algorithmes de gestion de systèmes de particules évolués gérés sur cartes graphiques, que nous utilisons pour simuler la dynamique des gouttes. Nous présentons ensuite les travaux que nous avons réalisés dans l'optique de permettre l'influence sur les particules d'un vent physiquement plausible et aisément contrôlable par l'utilisateur via une interface dédiée. Nous détaillons enfin la méthode de détection des collisions que nous avons mise en œuvre pour permettre le rebond des gouttes sur les surfaces impactées.

Le chapitre 5 présente les résultats de nos travaux.

La pluie est quelque chose que les Britanniques font mieux que personne.

The bleeding heart
Marilyn French

Chapitre 1

Rendu et animation de pluie en synthèse d'images

Historiquement, les phénomènes météorologiques ont longtemps été délaissés en synthèse d'images. On peut considérer que la pluie a commencé à être étudiée en 1983 avec l'introduction du concept de systèmes de particules par Reeves [Ree83]. Les premiers articles significatifs sont apparus dans les années 1990. Les principaux sujets d'étude liés à la pluie ont trait au rendu de gouttes d'eau évoluant sur des surfaces, le plus souvent avec une base physique forte, et fréquemment visualisés à l'aide de techniques à base de lancer de rayons, sans se préoccuper des temps de calcul.

Dans les années 2000, des domaines connexes à la synthèse d'images se sont intéressés à la représentation des phénomènes météorologiques. La vision par ordinateur a ainsi étudié les propriétés de la pluie afin d'appliquer des traitements sur des séquences vidéos (ajout ou suppression de pluie).

Depuis 2004, des techniques utilisant les possibilités offertes par le matériel graphique ont été mises en œuvre pour simuler de manière réaliste l'apparence de la pluie en temps-réel. Parallèlement, les avancées matérielles ont également permis l'implantation de systèmes de particules directement sur les cartes graphiques, présentant des performances très supérieures à leurs homologues sur CPU.

La première partie de ce chapitre présente les travaux portant sur la simulation de gouttes d'eau évoluant sur des surfaces. Ces travaux portent sur l'étude des tra-

jectoires suivies par les gouttes, ainsi que leur visualisation. La section 1.2 présente les travaux portant sur le rendu de pluie, en particulier les méthodes issues de la vision par ordinateur, et les méthodes orientées temps-réel. La section 1.3 présente les techniques ayant trait à la simulation de la dynamique des gouttes, à savoir les méthodes d'animation de systèmes de particules gérées sur la carte graphique, et les techniques de simulation de vent et de traitement de collisions.

1.1 Gouttes d'eau évoluant sur des surfaces

Les premiers travaux ayant cherché à représenter la pluie l'ont abordée par le biais des gouttes évoluant sur des surfaces, comme par exemple un pare-brise de voiture. Ces méthodes définissent à la fois les trajectoires suivies par les gouttes au cours de leur évolution, ainsi que les modalités de leur visualisation. Cette section vise à présenter les travaux majeurs réalisés dans ce cadre.

Kaneda *et al.* (dans [KKY93], [KZYN96] et [KIY99]), ainsi que Sato *et al.* (dans [SDY02]) se sont intéressés à l'écoulement de gouttes d'eau sur des surfaces, et ont appliqué leurs travaux à des pare-brise de voiture.

Fig. 1.1 – **Gauche** : Résultats obtenus dans [KIY99] ; simulation d'un pare-brise avec essuie-glaces devant une scène filmée. **Droite** : Résultats obtenus dans [SDY02], utilisant une photographie en arrière plan.

Dans les travaux de Kaneda *et al.*, une surface est discrétisée en une grille régulière. Le déplacement d'une goutte se fait de case en case dans la grille suivant

plusieurs paramètres comme son poids, ou encore l'affinité pour l'eau des cases adjacentes. L'affinité décrit la répartition des impuretés et des rayures sur la surface, ainsi que l'état sec ou mouillé des cases. Elle permet d'établir une probabilité d'évolution dans chacune des directions possibles. [KIY99] ajoute à ce modèle la force du vent, pouvant faire remonter la goutte sur le pare-brise. La présence d'obstacles comme des essuie-glaces sur la trajectoire des gouttes d'eau est également prise en compte. Dans ce cas, l'obstacle entraîne les gouttes dans son mouvement. Pour effectuer le rendu des gouttes, six images de la scène sont capturées pour chacune des cases de la grille, par projection des objets de la scène sur un cube englobant la case. Ces images sont utilisées pour former une carte d'environnement, utilisée pour obtenir une image réaliste de la goutte d'eau pour une position donnée.

L'article de Sato *et al.* améliore la méthode précédente en limitant la carte d'environnement à une seule capture, plaquée sur les gouttes en faisant une interpolation des réfractions calculées aux quatre coins d'un imposteur. Les auteurs ajoutent à cette méthode la notion de profondeur de champ, permettant à l'utilisateur de voir distinctement l'arrière plan derrière les gouttes légèrement floues au premier plan. La figure 1.1 permet de comparer les résultats obtenus dans les travaux de Kaneda *et. al* et de Sato *et. al.*

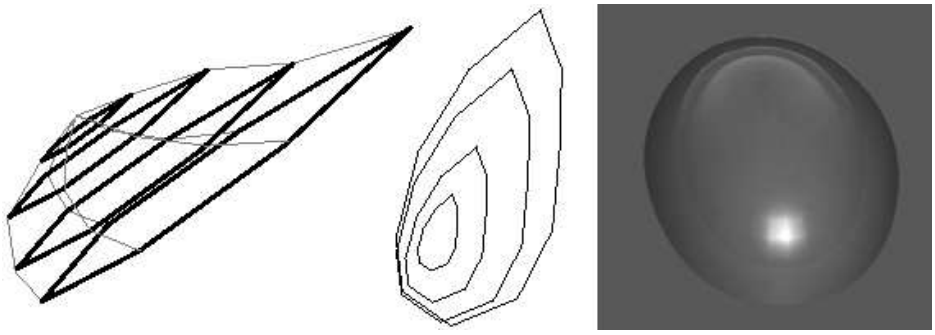

Fig. 1.2 – Résultats de [FHP98]. **Gauche** : Le maillage sous une forme d'équilibre. **Droite** : rendu lissé par une spline.

Fournier *et al.* dissocient dans [FHP98] les traitements relatifs aux déplacements de la goutte, de ceux ayant trait à sa forme. Le mouvement d'une goutte est affecté par plusieurs propriétés de la surface support, telles que la rugosité, les forces de

friction et d'adhérence, ainsi que par les collisions entre les gouttes. Chaque goutte se déplace sur une surface subdivisée en triangles, en avançant de triangle en triangle. Deux gouttes situées sur un même triangle peuvent entrer en collision et fusionner. Leurs masses sont alors additionnées, les moyennes de leurs positions et de leurs vélocités sont calculées, afin de définir les propriétés de la nouvelle goutte. La forme est définie par un modèle masse-ressort ; chaque sommet du maillage de la goutte est lié à ses plus proches voisins par un ressort, ainsi qu'au centre de la surface de contact entre la goutte et son support. L'ensemble des ressorts doit satisfaire quatre contraintes (préservation du volume, minimisation de la surface de contact avec l'air, maximisation de la surface de contact avec le support, déformation possible par des forces externes uniformes) afin d'arriver à une forme d'équilibre pour la goutte. Le maillage est ensuite lissé par une spline, dont les points de contrôle subissent des perturbations dépendant de la vitesse, afin de simuler les déformations à petite échelle que subit une goutte pendant son écoulement sur une surface. Une texture d'illumination est pré-calculée par un processus de Monte-Carlo, et utilisée pour effectuer le rendu des gouttes (la réfraction est ici ignorée). La figure 1.2 illustre le maillage utilisé par la méthode, et un exemple de rendu final.

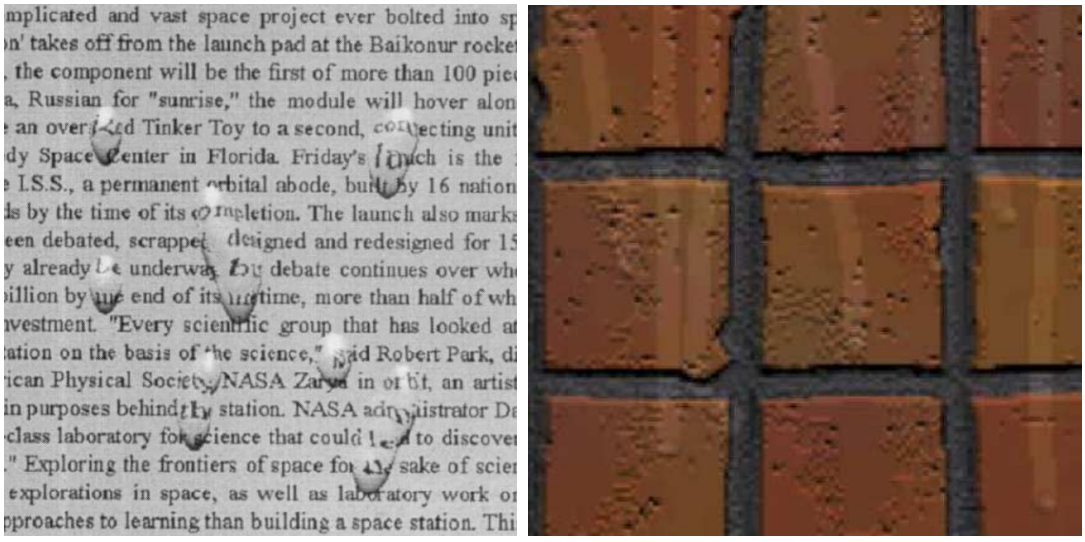

Fig. 1.3 – **Gauche** : gouttes sur une page de texte, dans [YJC99]. **Droite** : gouttes sur un mur de briques, dans [JH02].

Yu *et al.* proposent dans [YJC98] et [YJC99] des méthodes empiriques utilisant

des *metaballs* pour simuler des gouttes soumises à la gravité. Des particules sphériques sont déposées sur une surface, puis aplaties suivant un facteur dépendant de la gravité. Le modèle de *metaballs* permet aux gouttes de fusionner entre elles ; elles peuvent se déformer sous l'action d'impuretés déposées sur la surface, ou par interaction avec un utilisateur. Très peu de détails sont donnés sur le processus de rendu utilisé, si ce n'est qu'ils utilisent le modèle de Phong. Les résultats de cette méthode sont illustrés sur la partie gauche de la figure 1.3.

Dans [JH02], les auteurs s'intéressent à la simulation de gouttes évoluant sur des surfaces irrégulières. Ils proposent une méthode empirique utilisant des cartes de bosses (*bump-maps*) pour déterminer le chemin emprunté par les gouttes. Ces cartes sont appliquées sur les surfaces d'évolution des gouttes, et les normales tirées de ces cartes sont utilisées pour déterminer la direction de pente la plus forte. Le rendu adopté est une extension de celui proposé dans [KKY93], auquel est ajouté un effet de flou. Les résultats de cette méthode sont illustrés sur la partie droite de la figure 1.3.

Fig. 1.4 – Résultats obtenus dans [WMT05].

Plus récemment, Wang *et al.* [WMT05] introduisent une méthode d'animation de gouttes d'eau évoluant sur des surfaces quelconques. La méthode s'intéresse en

particulier à l'angle formé au point d'intersection entre une goutte et une surface, principalement défini par la tension de surface de la goutte. Le comportement des fluides à petite échelle est modélisé par l'emploi de ce que les auteurs appellent une *surface virtuelle*, définie par l'angle de contact entre la surface et la goutte, et qui est substituée à l'interface entre les deux milieux pour calculer la tension de surface. Le caractère hydrophile ou hydrophobe des surfaces est pris en compte par cette méthode, et permet la simulation de phénomènes de capillarité. Les gouttes sont représentées par des surfaces implicites, ce qui permet de simuler le regroupement et la séparation des gouttes. Celles-ci sont ensuite rendues par lancer de rayons. Les résultats s'appuient sur des bases physiques solides, et sont visuellement réalistes (comme l'illustre la figure 1.4), mais les temps de calcul sont très importants du fait de la complexité de la simulation physique (plusieurs heures de calcul par image).

1.2 Rendu de pluie en synthèse d'images

Depuis 2003, les travaux étudiant la pluie en imagerie numérique se sont étendus à sa perception au cours de sa chute. Des travaux ont été menés parallèlement dans le domaine de la vision par ordinateur, afin d'effectuer des traitements sur des vidéos pluvieuses, et en rendu temps-réel, afin d'obtenir une visualisation réaliste de pluie simulée.

1.2.1 Approches issues de la vision par ordinateur

La vision par ordinateur, domaine connexe à la synthèse d'images, s'est intéressé aux phénomènes visuels liés à la pluie. Starik et Werman, ainsi que Garg et Nayar, ont étudié les propriétés des gouttes d'eau, afin de pouvoir ajouter ou retirer de la pluie à des vidéos.

Starik et Werman [SW03] s'appliquent à réaliser des post-traitements sur des vidéos. Ils proposent une méthode permettant d'identifier dans une séquence vidéo les altérations des images caractéristiques d'une pluie fine. Ils peuvent ensuite retirer ces altérations dans une séquence filmée sous la pluie, donnant ainsi l'impression qu'elle a été filmée par temps sec. A contrario, ils peuvent appliquer un traitement

Fig. 1.5 – Résultats présentés dans [SW03]. **Gauche** : pluie réelle. **Droite** : pluie simulée.

à une séquence vidéo afin de donner l'impression qu'elle a été enregistrée par temps pluvieux. Ils utilisent pour cela un masque définissant les emplacements des gouttes dans l'espace image. Ce masque est établi en fonction de paramètres fournis par l'utilisateur, tels que la taille, la vitesse et la densité des gouttes, la direction du vent et la profondeur de la scène. Il est utilisé dans une approche empirique, pour augmenter la luminosité et flouter les zones de la scène censées êtres couvertes par une goutte. Aucune cohérence d'une image à l'autre n'est assurée, leur expérience ayant mis en évidence qu'un observateur ne perçoit pas le mouvement individuel des gouttes dans une séquence vidéo. Leur méthode permet également de corriger la luminosité globale de la scène, qui doit être moindre pour une scène pluvieuse que pour une scène ensoleillée. La figure 1.5 compare une image d'une séquence vidéo filmée sous la pluie, et une image obtenue en ajoutant à une vidéo filmée par temps sec de la pluie simulée par la méthode de [SW03].

Garg et Nayar ont également décrit des techniques de traitement de vidéos pluvieuses. Pour valider l'approche proposée, [GN03] établit un modèle théorique précis pour comprendre l'influence de la pluie dans une vidéo. Ce modèle analyse les contributions de la réfraction et des réflexions internes et externes sur l'apparence de la goutte. Des données simulées à l'aide de ce modèle sont comparées avec des données expérimentales, mesurées à l'aide d'une sphère en acrylique réfractant un damier placé en arrière plan ; le pourcentage d'erreur entre les données simulées et les don-

Fig. 1.6 – Une goutte simulée à partir d'une photographie omnidirectionnelle dans [GN03].

nées mesurées est inférieur à 1%. Les auteurs décrivent une méthode de rendu de gouttes utilisant une capture d'une scène réalisée par une caméra omnidirectionnelle, dont les résultats sont illustrés sur la figure 1.6.

Ces travaux sont étendus dans [GN04] à la détection et la suppression de pluie présente dans des vidéos (filmées par des caméras fixes). Les auteurs isolent les variations temporelles d'intensité de chaque pixel dans une séquence d'images, afin de détecter les gouttes présentes sur chaque image de la séquence. Les cas de faux positifs (liés à un mouvement autre que celui des gouttes), sont éliminés en s'assurant que la modification d'intensité détectée correspond à celle créée par une goutte de pluie, décrite par le modèle développé dans [GN03]. Un exemple de suppression de pluie dans une image est présenté sur la figure 1.7

Dans [GN05], le modèle d'apparence d'une goutte décrit dans [GN03] est utilisé pour étudier le comportement d'une caméra face à une scène pluvieuse. Les auteurs prouvent expérimentalement qu'il est possible en réglant les paramètres de la caméra (ouverture, longueur de la focale, temps de pose), de rendre la pluie plus ou moins

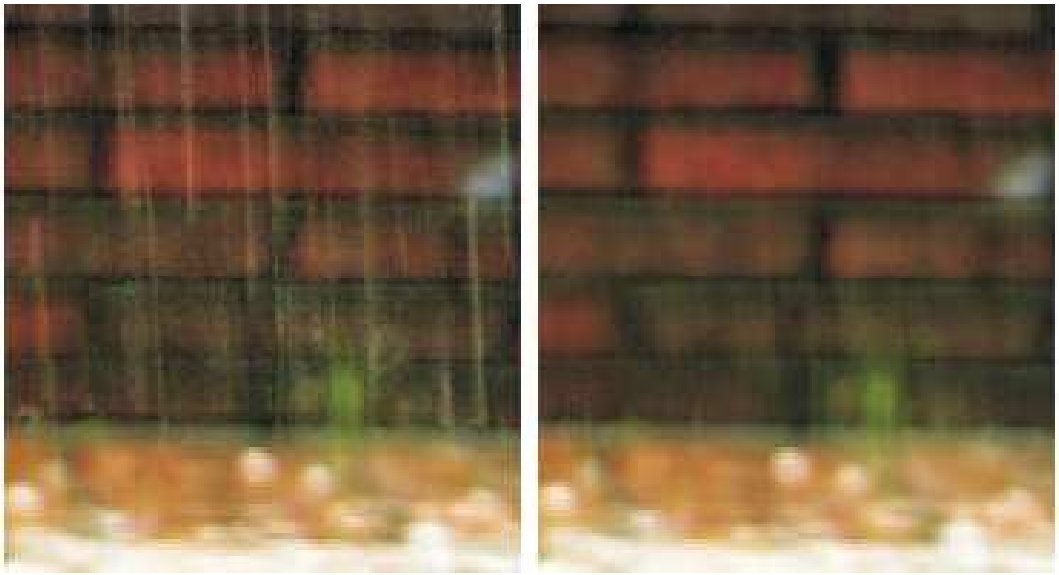

Fig. 1.7 – La pluie visible dans l'image de droite est supprimée dans l'image de gauche par la méthode présentée dans [GN04]

visible dans une vidéo. Une modification de la profondeur de champ est privilégiée pour estomper l'effet de la pluie dans le cas de scènes en mouvement, tandis qu'un allongement des temps de pose est utilisé pour les plans fixes.

Enfin, Garg et Nayar ont prolongé leurs travaux dans [GN06] par une étude de l'interaction des gouttes d'eau avec la lumière. Ils ont comparé des données réelles et des modèles analytiques tenant compte des oscillations des gouttes au cours de leur chute pour former une base de données d'images de gouttes étirées par un phénomène de persistance rétinienne, et soumises à l'illumination de sources lumineuses. Les images sont classées suivant l'angle d'observation et l'angle d'incidence de la lumière. Ils utilisent ensuite cette base de données pour ajouter de la pluie à des vidéos, auxquelles a été appliqué un pré-traitement manuel permettant d'obtenir une carte de profondeur des scènes représentées.

Les travaux présentés par Garg et Nayar dans [GN03], [GN04], [GN05], et [GN06] sont repris dans [GN07], qui présente une synthèse des méthodes proposées.

Ces travaux sont complétés dans [GKN07] par une étude empirique des éclaboussures générées par l'impact de gouttes d'eau sur une surface. À l'aide d'une caméra, des impacts réels sont observés sur diverses surfaces, classifiées selon des types gé-

nériques (rugueuse, lisse, flexible, autre), et dont l'inclinaison est prise en compte. Cela permet aux auteurs d'établir un ensemble de paramètres permettant de décrire l'éclaboussure générée. Une simulation d'éclaboussures est proposée en utilisant ces paramètres. Le rendu est réalisé en appliquant une méthode de lancer de rayons à une goutte projetée par les éclaboussures, puis en appliquant les résultats de ce lancer de rayons aux autres gouttes projetées.

1.2.2 Méthodes temps-réel

À l'heure actuelle, de nombreux jeux vidéos voient leur action se dérouler partiellement sous la pluie. À la demande des joueurs, les phénomènes météorologiques ont même été récemment ajoutés à l'un des jeux les plus en vogue du moment, *World Of Warcraft*. En général, la pluie est représentée dans les jeux vidéos à l'aide de systèmes de particules, tombant verticalement devant l'observateur ; chacune de ces particules est un trait vertical semi-transparent de couleur blanche. Celui-ci peut être un segment, géométriquement composé de deux points, ou il peut être approximé par une texture plaquée sur un rectangle étiré verticalement (couramment appelé *imposteur*, ou *billboard*). Cette deuxième technique présente l'avantage de permettre de faire varier la taille des gouttes en fonction de la profondeur par rapport à l'utilisateur, contrairement à l'utilisation de segments, qui ont une largeur fixe quelle que soit leur position dans la scène.

Langer et al. [LZK+04] présentent une méthode de synthèse spectrale basée image pour rendre de la pluie ou de la neige en temps interactif. Ils proposent une méthode mêlant l'utilisation d'un système de particules et une technique d'analyse spectrale permettant de créer dynamiquement une texture d'environnement pluvieux ou neigeux qui est composée avec la scène. Une transformée de Fourier inverse est appliquée à la distribution tridimensionnelle des particules, afin d'obtenir l'opacité suscitée par la neige ou la pluie en tout point de l'image. Cette opacité est utilisée pour générer la texture d'environnement, en affectant une couleur blanche aux zones opaques. Cette méthode permet d'avoir quelques particules (flocons ou gouttes) très identifiables, et une impression générale satisfaisante en temps interactif, pour des temps de calcul inférieurs à ceux nécessaires à l'animation d'un très gros système de particules. Les

Fig. 1.8 – Pluie intégré à un tableau, dans [LZK+04]

résultats obtenus en appliquant cette technique à un tableau sont présentés sur la figure 1.8.

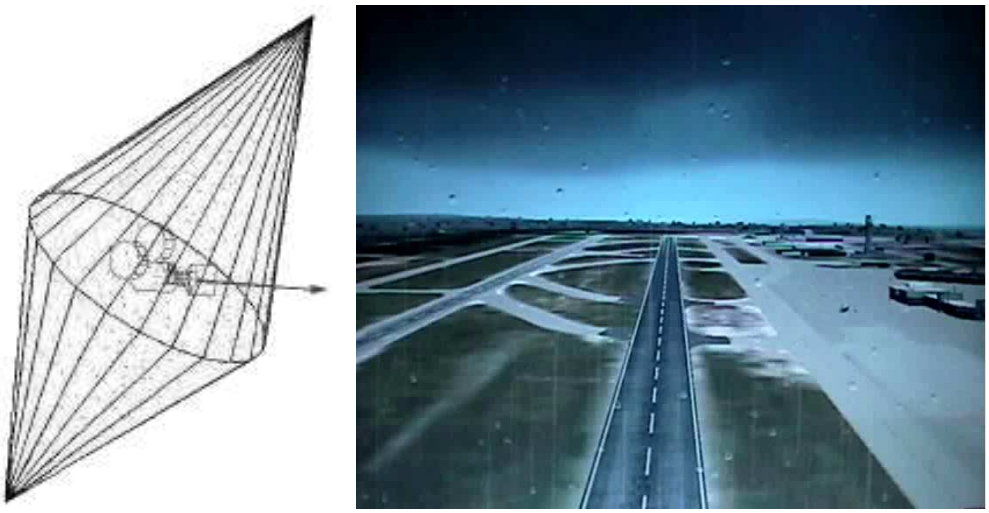

Fig. 1.9 – **Gauche** : les cônes employés par [WW04]. **Droite** : les résultats obtenus.

Un travail intéressant a été réalisé par Wang et Wade [WW04] pour *Microsoft*

Flight Simulator 2004. La méthode utilise deux cônes texturés, placés autour de l'observateur, comme illustré par l'image de gauche de la figure 1.9. L'orientation des cônes est déterminée en fonction de la position et de la vitesse de déplacement de la caméra. Les textures défilent sur ces cônes pour simuler la gravité. Plusieurs textures sont utilisées simultanément, avec des vitesses de défilement différentes, pour donner un effet de parallaxe. Il suffit d'imprimer une légère rotation à ces cônes pour donner l'impression que la pluie est soumise au vent. Cette technique est présentée comme étant plus adaptée que les systèmes de particules dans le cas de précipitations (pluie ou neige) importantes, et a un coût en temps de calcul très limité. L'inconvénient majeur de cette technique repose sur le fait qu'elle ne permet pas de gérer les interactions entre les particules et leur environnement, que ce soit des interactions dynamiques (rebonds et autres forces appliquées sur les particules) ou optiques (réfraction et réflexion de la scène visible dans les gouttes). Les textures utilisées doivent ici être créées par des artistes, qui devront donc dessiner des textures pour toutes les conditions présentes dans le jeu, en tenant compte de la densité des précipitations souhaitées et du niveau de luminosité de la scène dans laquelle sera utilisée cette texture. La partie droite de la figure 1.9 présente un exemple de visualisation obtenue par cette technique.

D'autres travaux, proposés par Yang *et al.* [YZZ04], introduisent une méthode simple de distorsion d'une image de la scène observée, visant à donner l'impression de gouttes sur un pare-brise. Une capture de la scène est plaquée sur un maillage disposé devant l'observateur, déformé autour des positions des gouttes pour produire une impression semblable à celle obtenue avec une loupe. Cette méthode utilise un algorithme phénoménologique au coût en temps de calcul très faible, non relié aux propriétés physiques de gouttes d'eau. Les résultats générés manquent nettement de réalisme, en particulier à cause du fait que les gouttes affichées se contentent de déformer la scène vue au travers d'elles, au lieu de la retourner comme le font les gouttes réelles. La figure 1.10 illustre les résultats obtenus par cette méthode.

La représentation de la pluie en temps-réel intéresse désormais les constructeurs de cartes graphiques. ATI et Nvidia ont intégré des programmes comprenant une simulation de pluie à leurs kits de développement logiciels afin d'illustrer les possi-

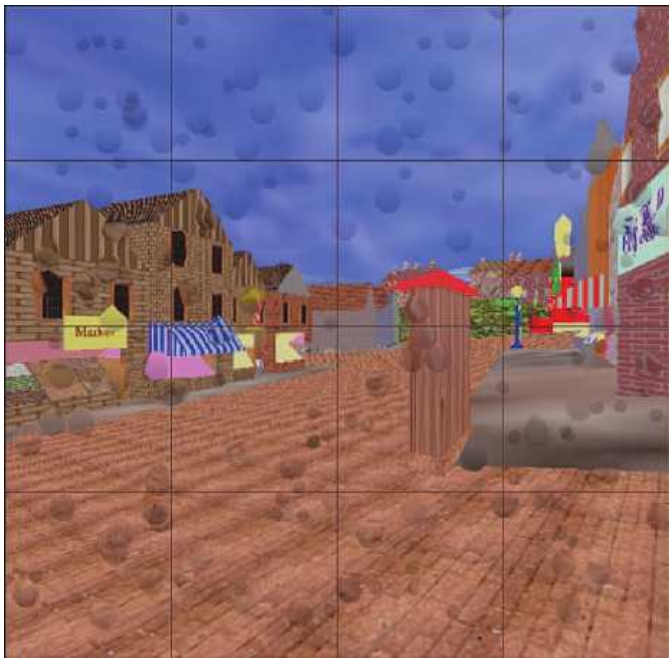

Fig. 1.10 – Résultats obtenus dans [YZZ04]

bilités de leur matériel. La figure 1.11 permet de comparer les résultats obtenus par les deux constructeurs.

ATI a été le premier à publier une démo intitulée *Toyshop Demo* mettant en scène une ville de nuit soumise à un orage. Cette démo regorge d'effets divers, détaillés dans [Tat06a], [Tat06b] et [TI06]. Parmi ces effets, celui qui nous intéresse le plus est évidemment la simulation de pluie. Les particules ne sont pas ici animées indépendemment, mais la technique utilisée repose plutôt sur un système de post-traitement : la pluie est ajoutée une fois que le reste de la scène a été rendu. Divers plans de gouttes sont utilisés pour donner un effet de parallaxe. Le fait que l'animation se déroule la nuit, et donc par faible luminosité, permet au rendu de s'affranchir d'une simulation de la réfraction et de la réflexion de la scène dans les gouttes. Les textures utilisées ont donc une teinte constante et une intensité modulée par l'intensité lumineuse globale de la scène. Ceci permet une interaction réaliste avec les éclairs qui illuminent brièvement et fortement la scène de temps à autres. Outre cette simulation de la pluie en chute libre, la démo contient également une simulation du ruissellement de gouttes, par exemple à partir de l'embouchure d'une

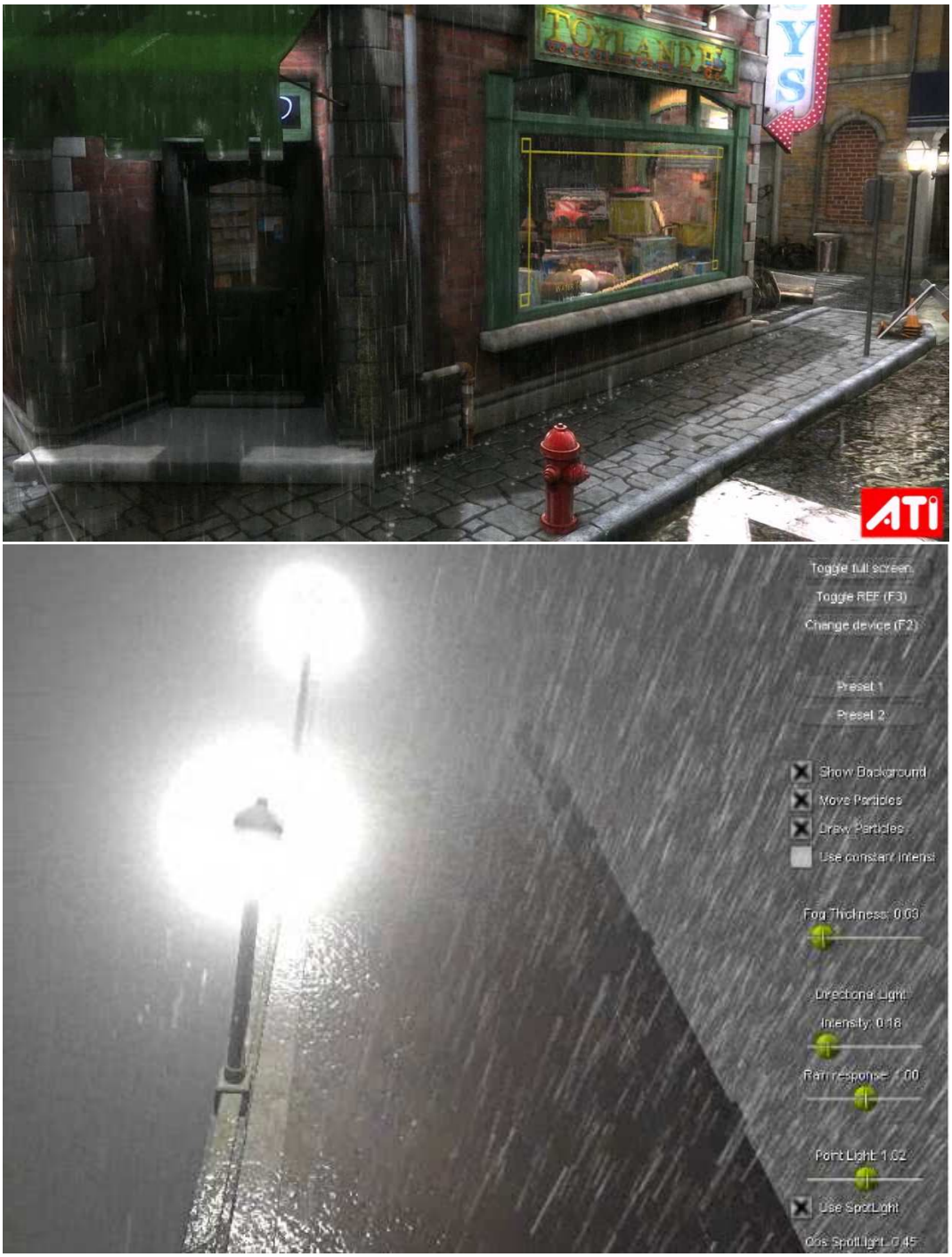

Fig. 1.11 – Haut : Demo Toyshop par ATI. Bas : Demo DirectX10 Nvidia

gouttière. Les gouttes sont ici simulées comme des particules indépendantes, animées sur la carte graphique par la méthode décrite dans [KSW04]. Enfin, cette démo intègre la simulation des éclaboussures provoquées par la collision des gouttes avec une

surface mouillée. Les positions des collisions sont aléatoires et non corrélées avec les positions des plans de parallaxe contenant les gouttes en chute libre. Le rendu des éclaboussures est réalisé à partir d'une séquence filmée d'une goutte tombant dans du lait (afin d'obtenir un contact plus visqueux et visible). Cette séquence est déformée pour éviter les trop fortes similarités entre les diverses éclaboussures visibles, puis est déroulée à chaque position d'une éclaboussure.

Nvidia propose aussi une démo de pluie avec son kit de développement logiciel pour DirectX 10, pour illustrer les possibilités de ses modèles Geforce 8. Les détails de cette démo sont donnés dans [Tar07]. Ils ont pour cette démo utilisé la base de données d'images créée par [GN06], dans une version allégée. Certains paramètres ont été éliminés, et la base a été restreinte à 300 textures. Pour le rendu d'une particule, les 4 textures les plus proches, suivant l'angle d'observation et d'illumination, sont sélectionnées et interpolées pour obtenir le résultat final. Comme la démo publiée par ATI, cette méthode utilise une scène nocturne et ne prend donc pas en compte la réfraction et réflexion dans l'apparence des gouttes, seule l'interaction avec la lumière est simulée.

1.3 Dynamique des gouttes de pluie

Nos travaux ont pour objectif de proposer une simulation réaliste de la pluie, en étudiant à la fois la visualisation des gouttes de pluie, et leur comportement dynamique. Afin d'obtenir une animation réaliste, nous nous sommes intéressés aux travaux antérieurs menés dans le domaine des systèmes de particules, et en particulier au méthodes apparues depuis 2003, permettant la simulation d'un système de particules sur la carte graphique de l'ordinateur.

Nous souhaitions également que les particules présentes dans notre simulation puissent être entraînées par le vent balayant la scène, et réagir à des collisions avec les objets présents dans la scène. Nous avons donc également étudié les méthodes antérieures permettant de simuler le vent, et de gérer les collisions entre particules et objets.

1.3.1 Systèmes de particules sur les cartes graphiques

Les systèmes de particules ont été initialement introduits par Reeves en 1983 [Ree83]. De nombreux articles ont depuis proposé des améliorations de cette technique fondamentale. Nous nous intéresseront ici en particulier aux méthodes implantées sur la carte graphique.

Les premiers systèmes de particules gérés par la carte graphique sont apparus en 2003. Le constructeur de cartes graphiques Nvidia s'est intéressé à ces phénomènes, et a proposé dans son kit de développement logiciel une application de démonstration présentant un système de particules géré par un *vertex shader*. Il est toutefois important de noter que ce système de particules est *sans état* ; les positions des particules sont calculées par le biais d'une fonction mathématique dépendante du temps, sans transfert de données d'une image à l'autre.

Le concept des applications GPGPU (*General-Purpose Computation Using Graphics Hardware*) est ensuite apparu. Les applications développées suivant ces principes utilisent les cartes graphiques comme des processeurs massivement parallèles et ultra-puissants, mais dévoyés de leur prédestination d'origine. Il s'agit ici de faire des calculs à vocation généralement non graphique, en utilisant les textures comme des tableaux de données. Les applications les plus courantes relèvent de la physique, même si des domaines aussi divers que l'économie, les bases de données ou la synthèse de sons ont également recours à cette approche. Owens *et al.* ont réalisé un état de l'art très complet des possibilités offertes par cette approche dans [OLG+05], actualisé dans [OLG+07]. La création de systèmes de particules gérés par la carte graphique est un bon exemple en matière de programmation GPGPU.

En 2004, [KLRS04] et [KSW04] ont introduit simultanément l'idée de systèmes de particules dynamiques accélérés par les cartes graphiques. L'idée directrice de ces techniques est d'enregistrer les positions et les vélocités des particules dans des textures, et de les mettre à jour en utilisant des *pixel shaders*. Ces deux méthodes, très proches tant d'un point de vue chronologique que technique, sont illustrés sur la figure 1.12

Dans [KLRS04], les auteurs utilisent les *Superbuffers* OpenGL pour transformer

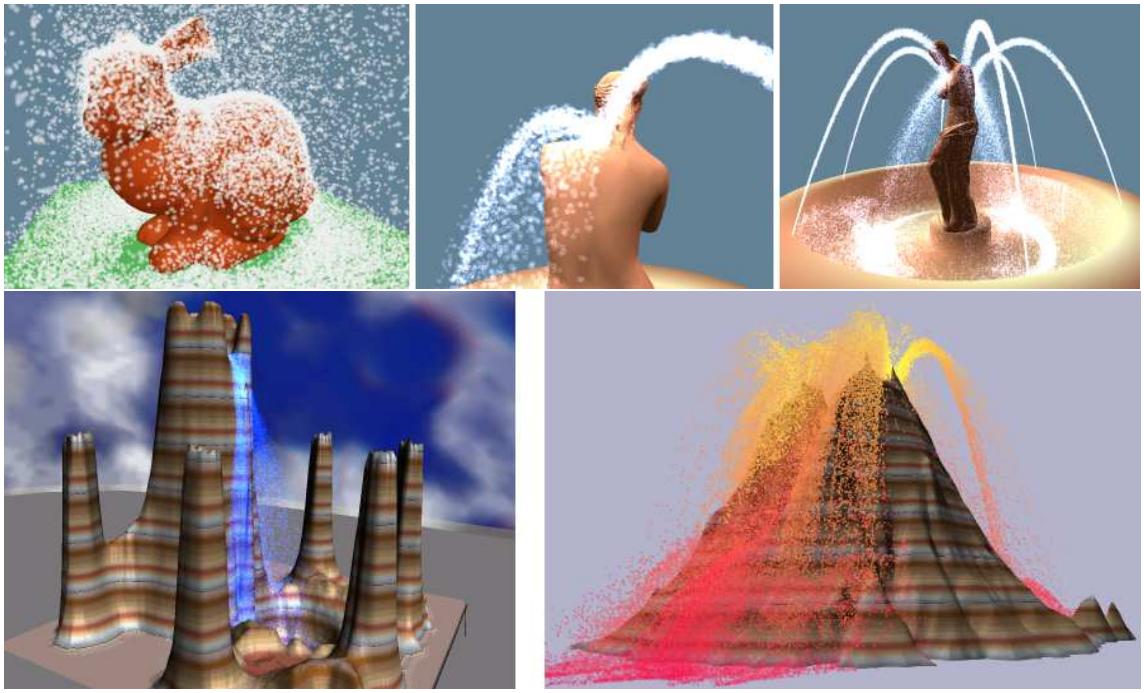

Fig. 1.12 – Haut : Résultats obtenus par [KLRS04]. Bas : Résultats obtenus par [KSW04]

les coordonnées des particules en géométrie. Ces tampons permettent de considérer les mêmes données comme étant alternativement une texture ou un tampon de sommets (*vertex-buffer* dans la littérature anglo-saxonne). Les auteurs procèdent au tri des particules suivant leur profondeur (indispensable pour un rendu semi-transparent) par l'algorithme de tri par fusion pair-impair de Batcher [Bat68]. Un ensemble de cartes de profondeurs est utilisé pour représenter les frontières externes des objets présents dans la scène, dans une approche basée-image similaire à celle introduite dans [KJ01]. Ces cartes de profondeur sont utilisées pour détecter les collisions entre les particules et les objets de la scène ; les collisions entre particules sont négligées. La méthode permet de rendre un million de particules en 10 images par seconde.

Dans [KSW04], les auteurs utilisent également les *Superbuffers* OpenGL pour générer la géométrie des particules. La technique présentée gère la naissance et la disparition d'un nombre limité de particules. Les collisions entre les particules sont rendues possibles par un tri des particules via une extension de la méthode de tri

bitonique adaptée à la carte graphique. La scène dans laquelle évoluent les particules est stockée dans une texture 3D, qui est ensuite utilisée pour détecter les collisions entre les particules et leur environnement (on notera que cette technique interdit implicitement toute évolution temporelle de la géométrie de la scène). Le système permet d'animer un million de particules simples à 7 images par seconde (lorsque tri et détection des collisions sont activés).

Les systèmes de particules gérés sur la carte graphique sont également utilisés pour la visualisation de grands ensembles de données. La technique proposée dans [KSW04] a été étendue dans [KKKW05] pour permettre l'advection des particules par un champ de vélocité statique tridimensionnel, pour simuler l'influence du vent (non-animé) sur les particules. Le vent est stocké dans une texture tridimensionnelle. Divers modes de rendu des particules sont proposés, tels que des lignes ou des rubans. [KKW05] présente des applications de cette technique à la visualisation d'images du cerveau humain obtenues par Imagerie à Résonance Magnétique. [BSK+07] utilise ce système pour visualiser les données issues d'un simulateur de tremblements de terre.

[Tar07] décrit les détails de l'application de démonstration technique comportant de la pluie, fournie dans le kit de développement logiciel pour DirectX 10 du constructeur de cartes graphiques Nvidia. Cette application utilise les *geometry shader*s introduits dans le *Shader Model 4.0* de DirectX 10 pour créer la géométrie des particules à la volée à partir d'une texture contenant les positions des particules.

1.3.2 Simulation du vent

Pendant la chute d'une goutte, sa trajectoire peut être affectée en présence de vent soufflant dans la scène. L'un des objectifs de nos travaux est de pouvoir gérer ce phénomène, aussi nous sommes nous intéressés aux méthodes existantes de simulation du vent.

Navier-Stokes

L'animation de vent est ce que l'on pourrait qualifier de cas d'école pour les méthodes de simulation de la mécanique des fluides. Elle peut être menée à bien de manière satisfaisante par des méthodes utilisant les équations de Navier-Stokes.

Ces équations sont des équations aux dérivées partielles non-linéaires, qui décrivent le mouvement des fluides dans l'approximation des milieux continus. Elles gouvernent par exemple les mouvements de l'air de l'atmosphère, les courants océaniques, l'écoulement de l'eau dans un tuyau, et de nombreux autres phénomènes d'écoulement de fluides. Elles sont nommées d'après deux physiciens du XIXe siècle, Claude Navier et George Stokes.

Ces équations décrivent le comportement des fluides avec une grande précision, et leur résolution est complexe. En synthèse d'images, elles sont le plus fréquemment employées dans des applications à forte base physique, utilisant des simulations très coûteuses en temps de calcul. Mark Harris a proposé une méthode de résolution de ces équations en temps-réel, utilisant la carte graphique, dans [Har04]. Les exemples présentés sont toutefois limités au cas bidimensionnel, et à des grilles de résolution restreinte.

De très nombreux articles présentant ces équations ainsi que leurs applications à la synthèse d'images sont présents dans la littérature scientifique. Du fait de leur coût en temps de calcul, ces méthodes ne sont pas en adéquation avec l'orientation que nous souhaitions donner à nos travaux ; nous n'avons donc pas cherché à en faire une analyse exhaustive. Une synthèse de ces méthodes, et de leurs dérivés visant à simuler toutes sortes de fluides, est présentée dans [BFMF06].

Treillis Boltzmann

Récemment, des techniques alternatives ont émergé, basées sur la méthode dite « treillis Boltzmann » (*Lattice Boltzmann Method* ou LBM dans la littérature anglo-saxonne).

La méthode treillis Boltzmann est une méthode numérique de simulation d'écoulement de fluides visqueux. Elle approxime l'équation continue de Boltzmann (du

nom de Ludwig Boltzmann, physicien autrichien du XIXe siècle), en discrétisant l'espace physique par un treillis, et l'espace des vélocités par un ensemble de vecteurs de vélocité microscopique. Dans la méthode treillis Boltzmann, l'espace physique est discrétisé en un ensemble de nœuds uniformément répartis dans l'espace, et le treillis (*lattice* dans la littérature anglo-saxonne) représente les espaces pleins et vides. Un ensemble discret de vélocités microscopiques est défini en chaque nœud pour la propagation des molécules du fluide. Les mouvements microscopiques des particules sont modélisés en utilisant des populations moléculaires appelées « fonctions de distribution », qui définissent la densité et la vélocité en chaque nœud du treillis. Les règles d'interaction spécifiques entre les particules sont définies afin de satisfaire les équations de Navier-Stokes.

À partir d'un noeud, les fonctions de distribution définissent la proportion de molécules qui se déplacera vers chacun des voisins. Les treillis sont nommés suivant l'espace dans lequel ils sont définis, et le nombre de nœuds vers lesquels les particules peuvent se déplacer. Ainsi, un treillis D3Q19 couvrira un espace tridimensionnel, et les molécules pourront se déplacer dans 19 directions :

- 1 direction nulle (pas de déplacement).
- 6 voisins immédiats, liés à la cellule courante par une face.
- 12 voisins secondaires, liés à la cellule courante par une seule arrête.

Les modèles les plus couramment utilisés sont les treillis D2Q9 (bidimensionnel), D3Q15 et D3Q19 (tridimensionnels). Ces deux derniers treillis sont illustrés sur la figure 1.13. Une présentation plus complète de cette méthode peut être trouvée dans [CDE94].

Appliquée à la synthèse d'images, la méthode treillis Boltzmann présente l'avantage d'être une méthode locale, permettant une parallélisation du calcul. Ceci permet une transposition simple vers une version matériellement accélérée par la carte graphique, comme le démontrent les auteurs de [LWK03] et [WLMK04]. Elle permet également une bonne gestion du comportement des fluides en présence d'obstacles aux contours complexes, comme expliqué dans [LFWK05].

Toutefois, cette méthode présente deux inconvénients majeurs. L'air présente un nombre de Mach (rapport de la vitesse locale d'un fluide sur la vitesse du son dans ce

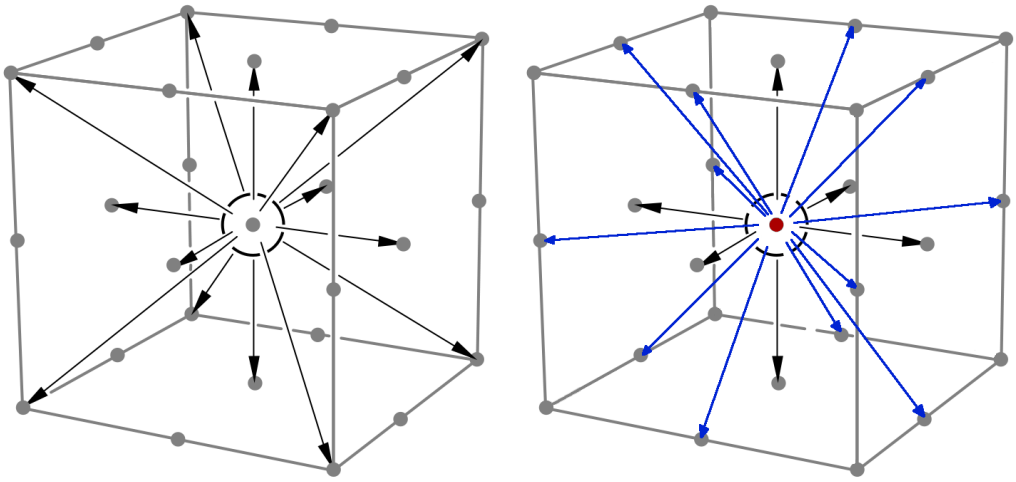

Fig. 1.13 – Treillis D3Q15 (gauche) et D3Q19 (droite) de la méthode treillis Boltzmann

même fluide) élevé. Ceci implique une instabilité numérique dans le cas de la méthode treillis Boltzmann. Dans [WZF+03] et [WZF+04], les auteurs proposent une méthode pour contourner ce problème, en utilisant un modèle de grille sous-jacente au treillis, prenant en compte les transferts d'énergie entre les différents niveaux d'échelle. Cette méthode engendre cependant une augmentation du temps de calcul de la simulation. Le second inconvénient de la méthode treillis Boltzmann réside dans sa capacité à s'adapter aux grandes échelles. Les techniques présentées dans [LWK03], [WZF+03] et [WLMK04] ne peuvent maintenir une vitesse d'exécution conforme aux contraintes temps-réel pour des treillis de résolutions supérieures à 64x64x64. L'animation de vent avec une dimension des cellules de la grille acceptable dans le contexte d'un espace extérieur ouvert comme celui qui nous intéresse, implique une très haute résolution de la grille, inatteignable en temps-réel en l'état actuel des capacités du matériel graphique pour la méthode treillis Boltzmann.

Méthodes d'animation procédurales

Ces considérations nous ont amené à étudier avec attention les méthodes de modélisation procédurale de vent à base de primitives simples, comme celles présentées dans [WH91], [PC01] et [GPR+03]. Des primitives simples sont ici combinées pour

créer des vents complexes dans l'espace.

Dans [WH91], les primitives sont choisies de manière à être des solutions des équations de Navier-Stokes. Ces équations présentent la propriété que les combinaisons de solutions sont elles-mêmes des solutions. En combinant des primitives simples, on peut donc obtenir des systèmes de vent complexes, qui restent solutions des équations de Navier-Stokes. Les primitives proposées sont au nombre de quatre : vent *uniforme* (homogène en force et direction dans l'espace couvert), *source* (le vent émane d'un point), *puits* (le vent se rassemble en un point) et *vortex* (le vent tourne autour d'un axe). Ces primitives sont utilisées dans un simulateur, permettant d'obtenir une animation de feuilles convaincante.

Dans [PC01], les auteurs utilisent des considérations d'ordre phénoménologique pour obtenir le réalisme visuel recherché. Les primitives utilisées sont ici sans base physique, mais offrent des résultats visuellement convaincants. Des masques planaires sont définis pour délimiter les différentes zones impactés par les différents types de vent (tourbillons, sources comparables à celles de [WH91], brise douce, vent aléatoire). Ces primitives sont ensuite utilisées pour animer une représentation multi-échelle de brins d'herbe. Ces travaux ont ensuite été étendus dans [GPR+03]. Ce deuxième article ajoute une nouvelle primitive, permettant d'écraser l'herbe sur le passage d'un objet. Il offre également la possibilité d'appliquer les masques de vent à des arbres animés, en plus des brins d'herbe gérés par [PC01].

Ces méthodes autorisent des échelles supérieures à celles obtenues par les méthodes plus conventionnelles de la mécanique des fluides, issues des équations de Navier-Stokes ou de la méthode treillis Boltzmann, du fait de la faible complexité des calculs mis en œuvre. Elles permettent donc de préserver les temps d'exécution à un niveau acceptable pour les hautes échelles.

Dernièrement, [Skj07] propose une méthode utilisant la carte graphique pour l'animation d'arbres par le vent. Ce vent est modélisé dans une approche empirique sous la forme d'une somme de sinus, dont la fréquence et la phase sont contrôlables. Cette somme est calculée dans un *pixel shader*, et utilisée pour définir la position et l'orientation des branches d'arbres.

1.3.3 Détection des collisions

Dans le cadre de notre système d'animation des particules, un objectif de nos travaux est de permettre la détection et la réponse aux collisions entre les particules et les objets présents dans la scène. Nos travaux se placent dans le cas particulier de la simulation de pluie, pour lequel l'emploi d'un modèle général de traitement des collisions n'est pas justifié.

D'innombrables articles évoquant la détection et la réaction aux collisions sont présents dans la littérature scientifique. Il est hors du périmètre du présent mémoire de les couvrir, aussi nous renvoyons le lecteur à [LG98] et [JTT01] pour un état de l'art complet de ces méthodes. Dans notre approche, nous avons affaire à un cas très particulier de collisions, dans lequel les collisions entre les particules peuvent être négligées sans compromettre l'aspect visuel de la simulation, et où toutes les particules partagent à quelques degrés près la même direction de chute.

Ces considérations nous ont amené à porter notre attention sur [TC00]. Dans cet article, les auteurs utilisent une technique de détection de collisions basée-image, étendant celle initialement présentée dans [MOK95]. Dans le contexte de l'animation de foule, cet article introduit l'idée d'utiliser une carte de hauteur pour déterminer si des particules (des personnes dans ce cas) peuvent se déplacer dans une direction donnée sans risquer de pénétrer un mur ou une autre particule.

1.4 Conclusion

Dans cet état de l'art des techniques de rendu et d'animation de pluie en synthèse d'images, nous avons pu constater que 2003 et 2004 se sont avérées être des années charnières. De nombreux articles fondateurs ont été publiés au cours de ces deux années, tant du point de vue du rendu que du point de vue de l'animation.

Auparavant, les travaux les plus intéressants avaient pour objet l'étude de gouttes d'eau évoluant sur des surfaces ; la pluie en chute libre était presque toujours représentée par le biais de systèmes de particules basiques, utilisant des textures dessinées par des artistes. Ces techniques induisent généralement un manque de réalisme, particulièrement lorsque les gouttes sont observées d'un point de vue proche.

L'évolution du matériel graphique a ouvert de nouvelles perspectives depuis le début des années 2000. On a ainsi pu voir apparaître des résultats visuellement intéressants, et fonctionnant pour la première fois en temps-réel. Parallèlement, l'émergence des concepts de programmation GPGPU a permis l'avènement des systèmes de particules implantés sur le matériel graphique, et utilisant des textures pour enregistrer les coordonnées des particules.

Dans ce contexte d'activité récente mais intensive, nos travaux ont visé à unifier les méthodes de rendu et d'animation pour offrir une approche globale de la simulation de pluie utilisant le matériel graphique, tout en proposant des avancées dans les deux domaines. Notre étude se place dans le cadre des jeux vidéos et des simulateurs de conduite ; il importe d'avoir un modèle de rendu capable d'afficher des gouttes visuellement réalistes, y compris lorsque celles-ci sont observées depuis un point de vue proche. L'animation doit également se montrer rapide et paramétrable afin d'adopter un comportement dynamique réaliste.

Dans le domaine du rendu, nous avons pu constater que la plupart des travaux antérieurs traitent la réfraction par des techniques qui ne correspondent pas aux objectifs visés par nos travaux. Elle est selon le cas, ignorée, approximée par des méthodes empiriques, ou calculée par lancer de rayons au prix de temps de calculs inabordables en temps-réel.

Dans le domaine de l'animation, les méthodes antérieures sont conçues pour être génériques, et travaillent généralement à une échelle réduite, insuffisante dans le cas de la simulation de pluie. De plus, l'interaction entre les particules et le vent, en particulier dans l'optique d'un vent dynamique et contrôlable par l'utilisateur, est un domaine que peu d'études ont traité par le passé.

Quand la pluie étalant ses immenses traînées
D'une vaste prison imite les barreaux

Les fleurs du mal
Charles Baudelaire

Chapitre 2

Propriétés physiques des gouttes d'eau

Si l'étude de la pluie en synthèse d'image est un domaine d'intérêt récent, les physiciens s'y sont intéressés depuis des décennies, et s'y intéressent encore aujourd'hui. Certains des derniers travaux du Prix Nobel de Physique Pierre-Gilles de Gennes portaient ainsi sur ce sujet ([GBWQ02]).

Les propriétés physiques des gouttes, tant d'un point de vue optique que dynamique, sont décrites par de nombreuses équations. Ce chapitre vise à les introduire, afin d'expliciter les bases théoriques sur lesquelles se sont appuyés nos travaux.

Nous décrirons dans un premier temps les équations régissant la forme adoptée par les gouttes au cours de leur chute, et la vitesse de cette chute. Nous étudierons ensuite les propriétés optiques des gouttes d'eau, et en particulier les phénomènes de réflexion et de réfraction dans l'approximation de l'optique géométrique.

2.1 Forme et dynamique des gouttes

2.1.1 Forme

Une idée reçue très courante veut que les gouttes de pluie ressemblent à des larmes, rondes en bas et effilées en pointe en haut. Les différentes études réalisées sur le sujet nous montrent que ceci est parfaitement faux. Oliver Ross [Ros00] dresse

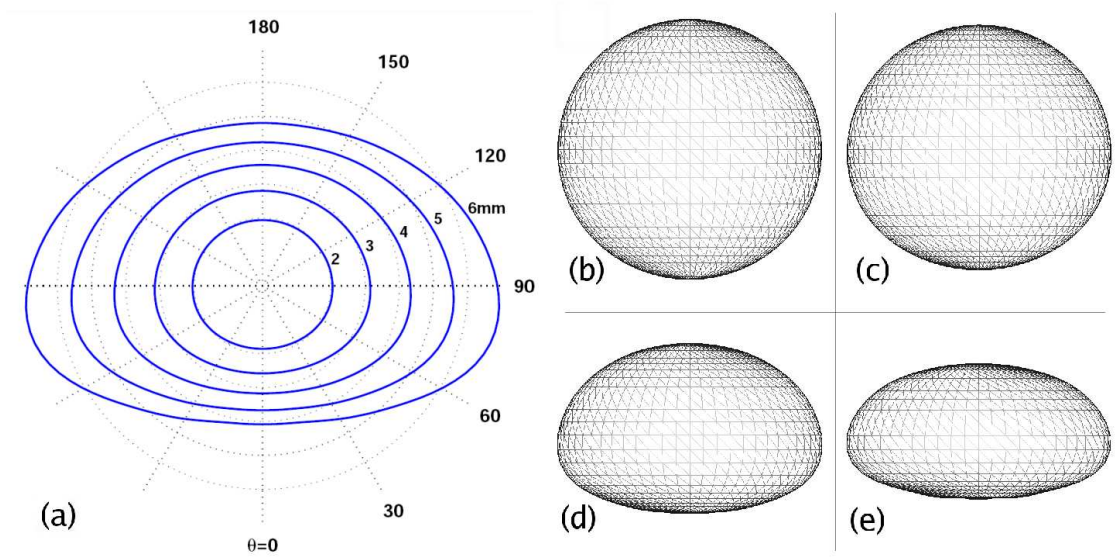

Fig. 2.1 – (a) : Profils de gouttes de Rayon R = 1, 1.5, 2, 2.5, et 3 mm [Ros00]. (b), (c), (d), (e) : formes de gouttes de rayons 0.5 mm, 1.0 mm, 3.0 mm et 4.5 mm

une liste des articles de référence dans ce domaine. Pour des gouttes de faible diamètre (inférieur à 2 mm), les formes sont presque parfaitement sphériques. Quand le diamètre augmente, les gouttes se déforment, elles s'allongent sur le plan horizontal, en s'écrasant sur le dessous ; elles tendent en fait à ressembler à des ellipses à la base aplatie, comme le montre la figure 2.1. On trouve parfois des références à un hamburger pour décrire cette forme.

Des équations ont été établies pour décrire cette forme. Green ([Gre75]) a proposé un modèle simplifié ; Beard et Chuang ([BC87], [CB90]) ont quant à eux présenté un modèle plus complexe mais très précis, introduisant une distorsion cosinusoïdale d'ordre 10 sur une sphère. Cette forme est le résultat de l'action de différentes forces antagonistes. La tension de surface cherche à minimiser la surface de contact entre l'eau et l'air, et tend à rendre la goutte sphérique ; la pression aérodynamique de l'air pendant la chute tend, quant à elle, à étaler la goutte et lui donner une forme ellipsoïdale.

Ces études montrent qu'il n'est pas indispensable de prendre en compte les déplacements et tensions électriques internes à la goutte, qui ont une influence négligeable sur sa forme. L'équation de Beard et Chuang décrivant la forme d'une goutte en chute

libre suivant son rayon est la suivante :

$$r(\theta) = a * \left(1 + \sum_{n=0}^{10} C_n * cos(n * \theta)\right) \tag{2.1}$$

où a est le rayon de la sphère non distordue, pris au centre de masse de la goutte. L'angle θ indique l'élévation polaire, $\theta = 0\,^\circ$ correspondant à la verticale orientée vers le bas. Les coefficients C_n sont donnés dans la table 2.1.

	Coefficients ($c_n \cdot 10^4$) pour n =										
a (mm)	0	1	2	3	4	5	6	7	8	9	10
0.5	-28	-30	-83	-22	-3	2	1	0	0	0	0
0.75	-72	-70	-210	-57	-6	7	3	0	-1	0	1
1.0	-134	-118	-385	-100	-5	17	6	-1	-3	-1	1
1.25	-211	-180	-592	-147	4	32	10	-3	-5	-1	2
1.5	-297	-247	-816	-188	24	52	13	-8	-8	-1	4
1.75	-388	-309	-1042	-221	53	75	15	-15	-12	0	7
2.0	-481	-359	-1263	-244	91	99	15	-25	-16	2	10
2.25	-573	-401	-1474	-255	137	121	11	-36	-19	6	13
2.5	-665	-435	-1674	-258	187	141	4	-48	-21	11	17
2.75	-755	-465	-1863	-251	242	157	-7	-61	-21	17	21
3.0	-843	-472	-2040	-240	299	168	-21	-73	-20	25	24
3.25	-930	-487	-2207	-222	358	175	-37	-84	-16	34	27
3.5	-1014	-492	-2364	-199	419	178	-56	-93	-12	43	30
4.0	-1187	-482	-2650	-148	543	171	-100	-107	2	64	32
4.5	-1328	-403	-2889	-106	662	153	-146	-111	18	81	31

Tab. 2.1 – Coefficients C_n pour la distorsion cosinusoïdale des gouttes (équation 2.1) [CB90].

2.1.2 Vitesse de chute

La vitesse de chute des gouttes dépend de leur rayon, comme l'indique la table 2.2. Les gouttes de plus gros diamètre ont une masse plus importante, et tombent

donc plus vite.

Les valeurs présentées sont établies une fois que la goutte a atteint sa vitesse de chute terminale, à laquelle l'action de la gravité et les forces de frottement s'équilibrent. La goutte cesse alors d'accélérer et adopte une vitesse constante.

Gouttes rondes		**Gouttes ellipsoïdales**			
rayon (mm)	vitesse (m/s)	rayon (mm)	vitesse (m/s)	rayon (mm)	vitesse (m/s)
0.1	0.72	0.5	4.0	2.5	9.2
0.15	1.17	0.75	5.43	2.75	9.23
0.2	1.62	1.0	6.59	3.0	9.23
0.25	2.06	1.25	7.46	3.25	9.23
0.3	2.47	1.5	8.1	3.5	9.23
0.35	2.87	1.75	8.58	3.75	9.23
0.4	3.27	2.0	8.91	4.0	9.23
0.45	3.67	2.25	9.11	4.25	9.23

Tab. 2.2 – Vitesse des gouttes suivant leur rayon [Ros00].

2.2 Propriétés optiques des gouttes d'eau

L'eau est un matériau aux propriétés optiques connues depuis longtemps. Son opacité est très faible pour des faibles volumes, et son indice de réfraction est tel que la réfraction est largement prédominante sur la réflexion dans l'apparence d'une goutte d'eau, comme l'illustre la figure 2.2. Cette figure présente deux photographies que nous avons réalisées en faisant goutter un robinet à faible distance de l'appareil photographique. La goutte visible dans l'image du haut n'a pas encore atteint sa forme stable. Les points blanc observables sur la première image sont dus au flash de l'appareil photographique.

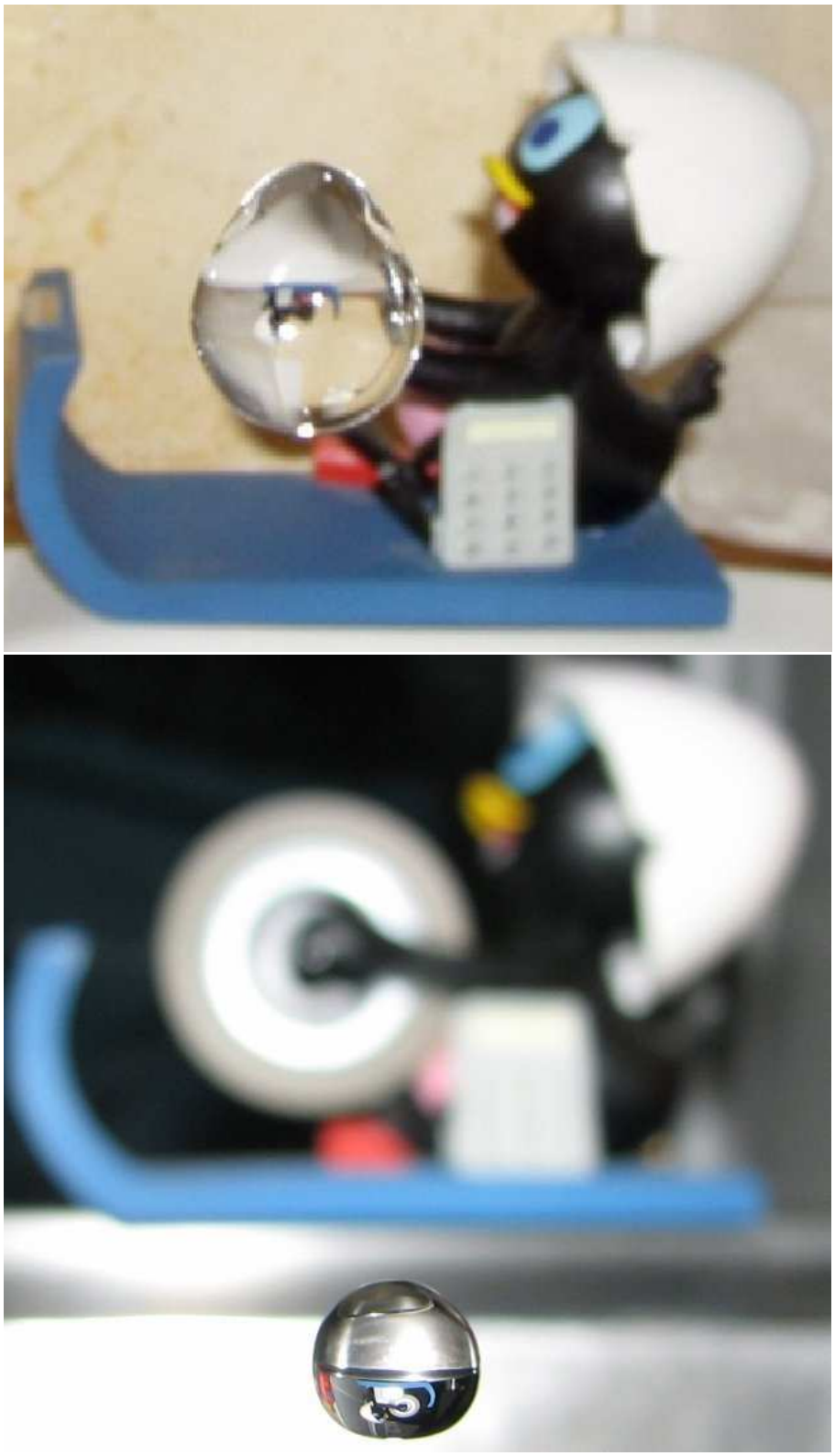

Fig. 2.2 – Deux photographies de gouttes d'eau réfractant l'objet placé en arrière plan. L'image du bas illustre parfaitement la forme décrite par l'équation 2.1

2.2.1 Optique ondulatoire

D'un point de vue physique rigoureux, la lumière doit être considérée comme ayant une nature corpusculaire. Dans cette approche, la lumière blanche peut être décomposée en une multitude de rayonnements lumineux, de longueurs d'ondes (et donc de couleurs) différentes. L'indice de réfraction d'un milieu est dépendant de la longueur d'onde considérée. Ceci implique qu'un rayon lumineux de couleur bleue (longueur d'onde courte du spectre visible) subira une déviation plus importante en pénétrant dans une goutte d'eau qu'un rayon de couleur rouge (longueur d'onde plus longue). En changeant de milieu de propagation, la lumière blanche se verra donc séparée en ses différentes composantes. Ce phénomène peut être observé dans les irisations produites par un prisme, ou, dans le contexte de la pluie, dans les arcs-en-ciel.

2.2.2 Optique géométrique

Les gouttes de pluie ont un diamètre de l'ordre de quelques millimètres, ce qui est très grand par rapport aux longueurs d'ondes de la lumière visible.

De nombreuses études indiquent que dans ces conditions, il est raisonnable de se placer dans l'approximation dite de l'optique géométrique, présentée en détail dans [Gla94]. Cette approximation permet de négliger les considérations de longueurs d'ondes. On utilisera ici des propriétés géométriques pour décrire le comportement des rayons lumineux. On utilisera un indice de réfraction moyen pour chaque milieu, et ce, quelle que soit la longueur d'onde de la lumière entrante.

On pourra éventuellement traiter les arcs-en-ciel comme un cas particulier, car ils sont la manifestation la plus visible de la nature ondulatoire de la lumière ; tous les autres phénomènes (réflexion, réfraction dans les gouttes d'eau) seront approximés de manière satisfaisante par les lois de l'optique géométrique.

Plusieurs lois simples régissent l'optique géométrique. Un rayon arrivant sur une interface entre deux milieux peut être soit réfléchi, soit réfracté (le terme « transmis » est également fréquemment rencontré dans la littérature scientifique).

Les directions empruntées par les rayons réfléchis et réfractés sont calculées à

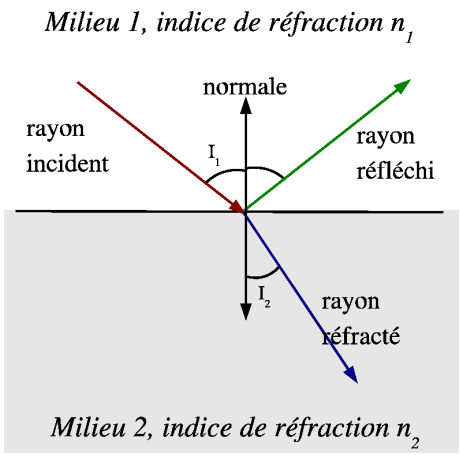

Fig. 2.3 – Réflexion / réfraction d'un rayon lumineux.

partir des *lois de Snell-Descartes*, relatives à la réflexion et à la réfraction.

- *Loi de Snell-Descartes de la réflexion* : le rayon est réfléchi dans le plan portant la normale à la surface et le rayon incident, dans la direction opposée à celle du rayon incident, à l'extérieur de la surface ; le rayon réfléchi forme avec la normale un angle égal à celui formé par la normale et le rayon incident, comme indiqué sur la figure 2.3.
- *Loi de Snell-Descartes de la réfraction* : le rayon est réfracté dans le plan portant la normale à la surface et le rayon incident, dans le demi plan intérieur à la surface, comme indiqué sur la figure 2.3. Le rayon réfracté forme avec la normale un angle défini par la relation :

$$n_1 * sin(I_1) = n_2 * sin(I_2)$$

où n_1 (resp. n_2) est l'indice de réfraction du premier (resp. deuxième) milieu de propagation, I_1 est l'angle formé entre le rayon incident et la normale à la surface, et I_2 est l'angle formé entre le rayon réfracté et l'opposé de la normale à la surface.

Le *facteur de Fresnel* permet de définir quelle proportion de l'intensité lumineuse du rayon incident est réfléchie ou réfractée. Il prend en considération la notion de polarisation de la lumière. Cette notion découle de la nature électromagnétique du rayonnement lumineux. On parle de polarisation parallèle quand le champ électrique

$\vec{E}$ du rayonnement lumineux se propage sur le plan portant le rayon incident et la normale à la surface. On parle au contraire de polarisation perpendiculaire si le champ électrique est porté par le plan orthogonal au plan incident.

Les équations suivantes définissent les rapports de l'intensité réfléchie ($r_\perp$ et $r_\parallel$) et transmise ($t_\perp$ et $t_\parallel$) sur l'intensité incidente pour chacun des deux types de polarisation.

$$\begin{aligned} r_\perp &= \frac{sin^2(I_1 - I_2)}{sin^2(I_1 + I_2)} \\ r_\parallel &= \frac{tan^2(I_1 - I_2)}{tan^2(I_1 + I_2)} \\ t_\perp &= 2 * \frac{sin^2(I_2) * cos^2(I_1)}{sin^2(I_1 + I_2)} \\ t_\parallel &= 2 * \frac{sin^2(I_2) * cos^2(I_1)}{sin^2(I_1 + I_2) * cos^2(I_1 - I_2)} \end{aligned} \tag{2.2}$$

La lumière naturelle est considérée comme n'étant pas polarisée ; elle est dans un état de polarisation intermédiaire aux deux présentés ci-dessus. Elle peut acquérir une polarisation en se réfléchissant, par exemple sur des étendues d'eau ou des fenêtres. Dans ce genre de cas précis, la polarisation a un impact visuel sur l'apparence des objets. [WUT+04] a introduit un modèle pour gérer la polarisation de la lumière dans le cas de scènes comportant de nombreux réflecteurs, et nécessitant un tel modèle. Les résultats sont convaincants, mais demandent des dizaines de minutes de calcul. Ces calculs ne sont en outre justifiés que dans des cas précis où la lumière acquiert une polarisation due à la présence de multiples réflecteurs, et peuvent être négligés dans le cas général.

En pratique, une bonne approximation consiste à réaliser une moyenne entre les valeurs obtenues pour les cas de polarisation parallèle et perpendiculaire. C'est d'ailleurs cette approche qui est préconisée par le constructeur de cartes graphiques Nvidia dans [Wlo02].

Les facteurs de réflexion et de réfraction peuvent alors s'exprimer :

$$\begin{aligned} r &= \frac{1}{2} * (r_\perp + r_\parallel) \\ t &= \frac{1}{2} * (t_\perp + t_\parallel) \end{aligned} \tag{2.3}$$

En injectant les équations 2.2 dans les équations 2.3, on obtient :

$$\begin{aligned} r &= \frac{sin^2(I_1 - I_2) * (cos^2(I_1 + I_2) + cos^2(I_1 - I_2))}{2 * sin^2(I_1 + I_2) * cos^2(I_1 - I_2)} \\ t &= \frac{sin^2(I_2) * cos^2(I_1) * (1 + cos^2(I_1 - I_2))}{sin^2(I_1 + I_2) * cos^2(I_1 + I_2)} \end{aligned} \tag{2.4}$$

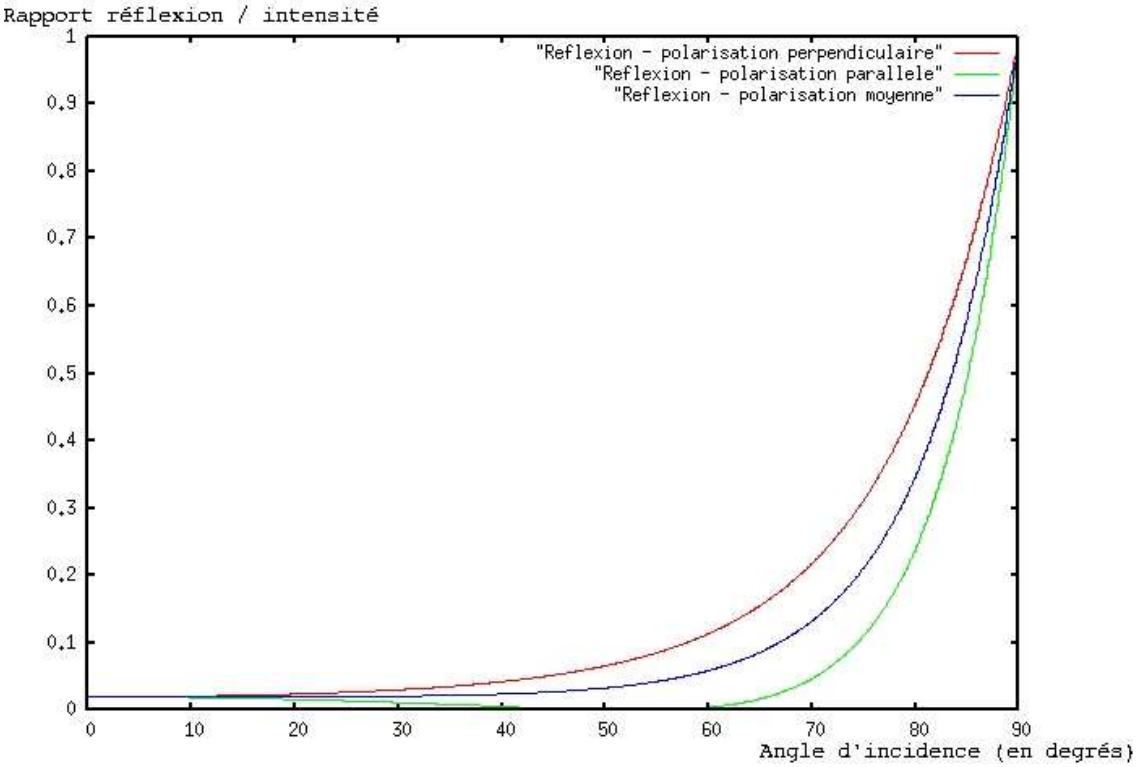

Fig. 2.4 – Contribution des rayons réfléchis selon l'angle d'incidence, pour les polarisations parallèles et perpendiculaires.

Les équations 2.4 nous permettent de déterminer que dans le cas d'une interface air-eau, milieux dont les indices de réfraction sont respectivement 1 et 1.33, l'influence de la réflexion sur l'apparence de la goutte est inférieure à 10% pour tout angle d'incidence inférieur à 67.5 °. Le graphique 2.4 illustre l'importance de la réflexion, pour les polarisations parallèles, perpendiculaires et moyennes, en fonction de l'angle d'incidence.

2.3 Conclusion

Ce chapitre a introduit les bases des propriétés physiques des gouttes d'eau. L'équation de Beard et Chuang nous a permis de connaître la forme adoptée par les gouttes pour différents rayons. Cette forme évolue de la sphère vers l'ellipsoïde à mesure que le rayon de la goutte augmente. Nous avons également pu connaître la vitesse de chute des gouttes, suivant leur rayon.

Enfin, le calcul du facteur de Fresnel nous a permis de constater que le phénomène de réflexion n'a une contribution significative à l'apparence des gouttes que pour les angles d'observation rasants ; la contribution du phénomène de réfraction à l'apparence de la goutte est très fortement prédominante dans les autres cas.

La faim est un nuage d'où il tombe une pluie de science et d'éloquence. La satiété est un autre nuage qui fait pleuvoir une pluie d'ignorance et de grossièreté.

PLATON

Chapitre 3

Rendu de pluie en temps-réel

Au cours de notre étude des travaux antérieurs portant sur le rendu de pluie (chapitre 1), nous avons pu constater que les méthodes existantes sont mal adaptées au cadre des jeux vidéos et des simulateurs de conduite. En effet, l'apparence des gouttes est généralement simplifiée à l'extrême (par des textures dessinées par un artiste), ou obtenue par des processus coûteux en temps de calcul (utilisant des techniques de lancer de rayons). Les jeux vidéos et les simulateurs de conduite présentant des scènes pluvieuses ont pourtant besoin d'un modèle temps-réel de visualisation réaliste des gouttes d'eau.

En s'appuyant sur les propriétés optiques, et sur la forme des gouttes d'eau introduites dans le chapitre précédent, nous avons développé un modèle de rendu de gouttes d'eau fonctionnant en temps-réel, centré sur le phénomène de réfraction, largement prédominant dans l'apparence des gouttes d'eau.

Nous avons par la suite étendu notre modèle à la prise en compte de la persistance rétinienne, et de l'interaction des gouttes avec les sources lumineuses disposées dans la scène.

Toutes ces méthodes sont implantées sur la carte graphique, et utilisent des *ombreurs de surface* ou *pixel shaders* programmés dans le langage *Cg*. Nous emploierons le terme de *pixel shader* dans la suite de ce mémoire pour désigner ces programmes.

Ce chapitre présente dans une première partie le modèle de rendu que nous avons développé, permettant de simuler la réfraction perceptible au travers des gouttes d'eau. Il détaille ensuite la technique de simulation de la persistance rétinienne que

nous avons mise en œuvre, puis notre méthode de simulation de l'interaction entre les gouttes et la lumière.

3.1 Modèle de rendu de gouttes

Un objectif de nos travaux était de proposer un modèle de l'apparence des gouttes d'eau, offrant des résultats visuellement réalistes en temps-réel. Comme indiqué dans la section 2.2, l'apparence d'une goutte est principalement due au phénomène de réfraction. Pour simuler ce phénomène, nous avons proposé une méthode utilisant une texture d'environnement, plaquée sur les gouttes dans un *pixel shader* en tenant compte de la direction dans laquelle sont réfractés les rayons lumineux sortant de la goutte.

La première partie de cette section présente les hypothèses sur lesquelles nous avons construit notre modèle. Une description complète de ce modèle est ensuite proposée, dans laquelle nous détaillons la phase de pré-calcul des directions de réfraction de la lumière au travers d'une goutte, l'extraction de la texture d'environnement, et son plaquage sur les gouttes d'eau. Cette section s'achève sur une discussion des résultats obtenus.

3.1.1 Hypothèses

Pour élaborer notre modèle de rendu des gouttes d'eau, nous avons formulé certaines hypothèses, et utilisé certaines approximations relatives à la géométrie et à l'apparence des gouttes, que nous allons introduire en préambule à la présentation du modèle.

Vers une géométrie simplifiée

Un des objectifs de notre modèle est d'être capable d'afficher des milliers, voire des dizaines de milliers de gouttes simultanément, en temps-réel. Un nombre plus important de gouttes n'est pas utile, les gouttes tombant sous le seuil du pixel au delà d'une certaine distance (un modèle multi-résolution, simulant la diminution de la profondeur observable dans la scène, devient alors plus approprié). Un modèle

géométrique des gouttes composé d'un maillage complexe calculé à partir de l'équation 2.1, et constitué de dizaines ou de centaines de triangles par goutte, nous est rapidement apparu comme n'étant pas envisageable. De plus, pour des gouttes n'occupant chacune que quelques pixels à l'écran, un tel modèle se serait également avéré inutilement coûteux en temps de calcul.

Nous avons donc cherché à adopter le modèle géométrique le plus simple et léger possible pour nos gouttes. [MS95] introduit un modèle d'approximation d'une géométrie complexe par un rectangle texturé, désigné sous le nom d'*imposteur* (*billboard* dans la littérature anglo-saxonne). Dans ce modèle, une géométrie est approximée sous la forme d'un quadrilatère faisant toujours face à la caméra.

Nous avons fait le choix d'utiliser ce modèle pour la représentation de nos gouttes, afin de diminuer au maximum la quantité de géométrie à afficher, et de préserver ainsi la rapidité d'exécution de notre modèle.

Vers un modèle de réfraction

Le calcul du facteur de Fresnel, présenté en section 2.2, nous a permis de constater que la réflexion ne participe de manière significative à l'apparence d'une goutte que pour les angles d'observation rasants.

Pour des angles d'incidence inférieurs à 67.5 °, l'influence de la réflexion sur l'apparence de la goutte est inférieure à 10%. Elle atteint 20% au delà de 74.4 °, et 50% seulement pour les angles d'incidence supérieurs à 83.5 °. Le phénomène de réflexion n'est ainsi observable que sur une fine bande du contour externe d'une goutte d'eau.

Dans notre application, une goutte typique, de rayon 1 mm, sera en moyenne observée sur l'écran sous la forme d'un ellipsoïde de 10 pixels de largeur. Dans ce cas, la réflexion n'atteindra une contribution significative dans l'apparence de la goutte que pour la bande de pixels la plus externe de l'enveloppe.

Ces constatations nous permettent de considérer qu'il est raisonnable de négliger la participation de la réflexion à l'apparence de la goutte. Nous avons donc fait le choix de centrer notre modèle sur le phénomène de réfraction.

Vers une réfraction « efficace »

Nous avons ensuite cherché à déterminer quelle portion d'une scène était « vue » au travers d'une goutte d'eau.

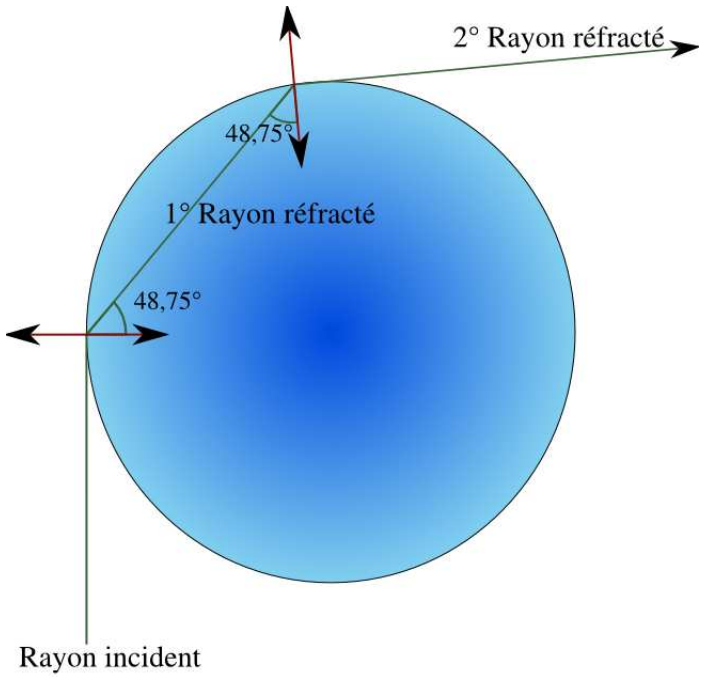

Fig. 3.1 – Déviation maximale de réfraction pour une goutte.

L'indice de réfraction de l'air est 1, celui de l'eau est 1.33. Sur les bords d'une goutte, là où la déviation due à la réfraction est maximale, l'angle entre le rayon incident (provenant de l'observateur) et la normale à la goutte, est de 90 °, comme illustré sur la figure 3.1.

D'après la loi de *Snell-Descartes*, l'angle entre le rayon réfracté et le rayon incident est de 48.75 °. La normale au point où le rayon sort de la goutte forme un angle de 7.5 ° avec le rayon incident initial. Le rayon réfracté forme un angle de 48.75 ° avec cette normale, et est réfracté de l'eau vers l'air suivant un angle de 90 ° par rapport à la normale, soit 82.5 ° par rapport au rayon incident initial.

Le « champ de vision » d'une goutte d'eau s'étend donc sur 165 °. Ce calcul confirme le chiffre avancé par Garg et Nayar dans [GN03]. Le fait que cet angle soit inférieur à 180 ° a été l'une des clés qui nous a permis d'imaginer le modèle que nous avons développé.

3.1.2 Description du modèle de rendu de gouttes

Ce que l'on perçoit au travers d'une goutte d'eau est une image de la scène située en arrière plan de la goutte, déformée et retournée, comme l'illustre la figure 2.2. Pour simuler cet effet, nous avons développé un modèle permettant de capturer une image de la scène située en arrière plan des gouttes vers une texture, puis de plaquer cette texture sur les gouttes en tenant compte de la réfraction qu'elles occasionnent.

Aperçu de la méthode

Comme expliqué dans la section 3.1.1, le champ de vision d'une goutte s'étend sur 165 °. Ceci nous permet de limiter la capture de l'environnement à une seule image, extraite par le biais d'une caméra virtuelle disposant d'un angle de vision suffisamment large. Nous évitons ainsi la lourdeur inhérente aux méthodes de cartes d'environnement conventionnelles, qui extraient 6 captures de la scène pour simuler un cube englobant.

Pour déterminer le mode de plaquage de cette capture de la scène sur les gouttes, nous avons recours à des masques indiquant la forme des gouttes, ainsi que les directions empruntées par les rayons réfractés hors des gouttes. Ces masques sont donc un ensemble de vecteurs de réfraction (nuls si l'on est en dehors de la goutte), calculés lors d'une phase de pré-traitement et stockés dans des textures.

À l'exécution du programme, pour chaque image rendue, la scène est capturée vers une texture grand-angle. La couleur de chaque pixel des gouttes est déterminée dans un *pixel shader* de la manière suivante :

- À l'aide du masque, on détermine si le pixel est à l'intérieur ou à l'extérieur de la goutte.
- S'il est à l'intérieur, on utilise le masque pour déterminer la direction du vecteur réfracté.
- On cherche ensuite le point dans la texture capturée vu au travers de la goutte dans la direction du rayon entrant sans tenir compte des déviations dues à la réfraction.
- Dans l'espace image, on ajoute le vecteur de réfraction à la position trouvée

au point précédent.

– Enfin, on extrait le pixel de cette nouvelle position, qui est celui vu au travers de la goutte.

Pré-calcul du masque

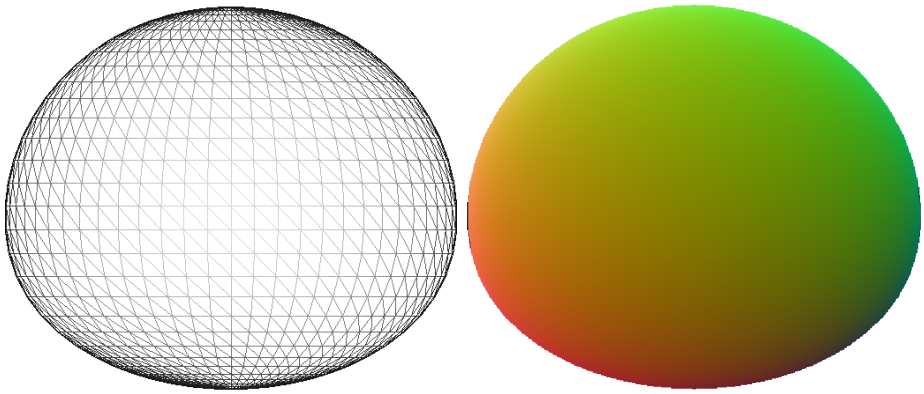

Fig. 3.2 – **Gauche** : vue tridimensionnelle filaire d'une goutte de rayon 1.5 mm. **Droite** : le masque pré-calculé pour cette goutte.

Un programme annexe utilise l'équation 2.1 pour calculer en trois dimensions la forme d'une goutte dont le rayon est fourni en paramètre. Une forme tridimensionnelle de goutte arbitraire, fournie par l'utilisateur, peut également être utilisée.

Pour chaque pixel de cette forme, un rayon incident est défini comme étant aligné sur l'axe Z, progressant en direction des Z positifs. Son point d'intersection avec la goutte, puis la direction dans laquelle il est réfracté à l'intérieur de la goutte sont calculés. Le point duquel ce rayon réfracté ressort de la goutte est ensuite déterminé, ainsi que la direction de réfraction hors de la goutte. Cette direction, normalisée, sera désignée sous le terme de *vecteur de réfraction* dans la suite de ce mémoire.

Les trois composantes X, Y et Z du vecteur de réfraction sont stockées sur les canaux rouge, vert et bleu d'une texture de résolution 512x512, dont une illustration est proposée sur la partie droite de la figure 3.2.

Dans le *pixel shader* chargé de plaquer la texture d'environnement sur les gouttes, le masque est utilisé simultanément pour donner la forme de la goutte, et pour

déterminer la direction de réfraction pour le simple coût d'un accès texture.

Au lieu de pré-calculer ce masque, nous aurions pu faire appel à la fonction *refract* du langage *Cg*, afin de calculer les vecteurs de réfraction à l'exécution. L'inconvénient de cette approche est qu'elle limite la forme des gouttes à des types de géométries particuliers comme les sphères parfaites, le calcul de la position des points de sortie des rayons réfractés étant trop complexe pour être effectué dans le cas général sans compromettre les performances de l'application.

Des masques ont été calculés pour toutes les tailles courantes de gouttes, allant de $0.5mm$ à $4.5mm$ par pas de $0.25mm$, et enregistrés dans des textures, afin de permettre de changer librement la taille des gouttes affichées lors de l'exécution de notre programme.

Capture de la scène vers une texture

Fig. 3.3 – **Gauche** : la scène vue par un observateur. **Droite** : la même scène capturée depuis le même point de vue par notre texture d'environnement.

Afin de générer la capture d'environnement qui sera ensuite plaquée sur les gouttes, une caméra virtuelle est positionnée à l'emplacement de l'observateur, avec la même direction de vue, et un champ de vision très large (165 °, angle déterminé précédemment dans la section 3.1.1). Cette caméra utilise une projection perspective, similaire à celle utilisée par la caméra principale fournissant la vision de l'observateur. Nous utilisons ici les possibilités de *rendu vers texture* des cartes graphiques,

pour obtenir la texture qui sera plaquée sur les gouttes dans notre *pixel shader*.

Considérant la taille et la vitesse de déplacement des imposteurs sur lesquels cette texture sera plaquée, notre expérience a montré que 512x512 est une résolution satisfaisante pour cette texture. La figure 3.3 illustre les détails visibles dans une capture de la scène réalisée par cette caméra à large champ de vision, comparée à la perception qu'a un observateur de cette scène.

Plaquage de la texture sur les gouttes

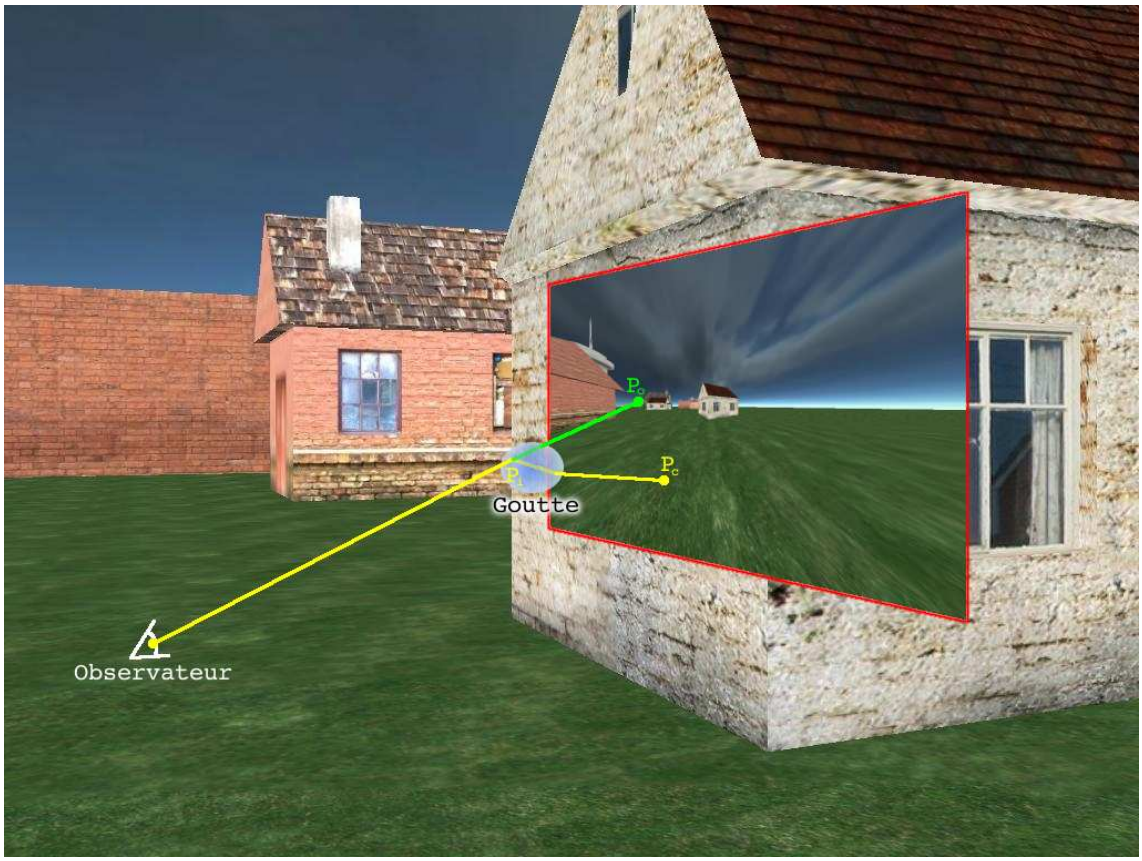

Fig. 3.4 – Extraction du pixel à partir de la texture capturée, ici délimitée en rouge.

Pour chaque pixel P_i de la goutte, un *pixel shader* extrait de la texture capturée le pixel réfracté vers l'observateur au travers de la goutte (le processus d'extraction est illustré sur la figure 3.4). Pour cela, le *pixel shader* localise dans un premier temps le pixel P_o dans la texture capturée qui est l'image de l'objet dans la scène vu par l'observateur suivant la direction de visée de P_i. P_o est le pixel qui serait vu si

la goutte d'eau avait le même indice de réfraction que l'air. Les coordonnées de P_o dans l'espace image sont obtenues par l'équation 3.1.

$$\overrightarrow{P_o} = 0.5 + (\overrightarrow{P_{i-ecran}} - 0.5) * \frac{tan(FOVy_{observateur}/2)}{tan(FOVy_{capture}/2)} \tag{3.1}$$

où

- $\overrightarrow{P_o}$ représente les coordonnées 2D de P_o dans la texture capturée.
- $\overrightarrow{P_{i-ecran}}$ représente les coordonnées du point P_i, relatives au repère de l'écran (le coin haut-gauche a pour coordonnées $(0, 0)$, et le coin bas-droit $(1, 1)$).
- $FOVy_{observateur}$ est l'ouverture du champ de vision de la caméra principale sur l'axe vertical, typiquement égale à 45 °.
- $FOVy_{capture}$ est l'ouverture du champ de vision de la caméra servant à la capture grand-angle de la scène, égale à 165 °.

Le vecteur de réfraction est ensuite extrait du masque, et combiné avec la position de P_o pour identifier le pixel P_c. C'est ce pixel qui donne la couleur de P_i, l'image réfractée par la goutte dans cette direction de visée. Les coordonnées de P_c sont obtenues en modifiant l'équation 3.1 pour qu'elle intègre le déplacement dans l'image effectué d'après la direction de réfraction extraite du masque ; elle devient alors :

$$\overrightarrow{P_c} = 0.5 + (\overrightarrow{P_{i-ecran}} - 0.5 - \frac{refract_{xy}}{refract_z}) * \frac{tan(FOVy_{observateur}/2)}{tan(FOVy_{capture}/2)} \tag{3.2}$$

où

- $\overrightarrow{P_c}$ représente les coordonnées 2D de P_c dans la texture capturée.
- $refract_{xy}$ et $refract_z$ sont les coordonnées xy et z du vecteur de réfraction extrait du masque. La division par la coordonnée z de ce vecteur permet d'allonger le décalage dans l'image pour les vecteurs subissant une forte déviation par la réfraction.

La figure 3.5 propose une comparaison entre une goutte simulée grâce à notre méthode (à gauche) et une vraie goutte (à droite). Une photographie de la scène originale a été utilisée comme image d'arrière-plan pour la goutte simulée. Les images du bas sont un agrandissement des gouttes. La goutte réelle n'a pas encore atteint

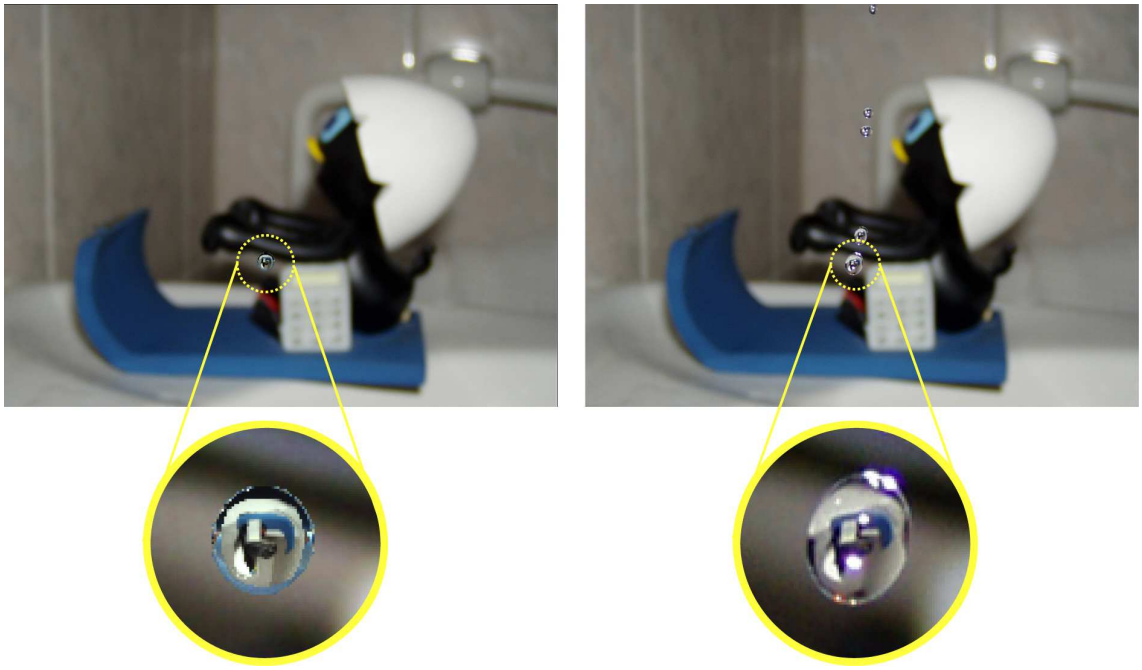

Fig. 3.5 – **Gauche** : une image simulée par notre méthode. **Droite** : photographie d'une vraie goutte.

sa forme stable sur la photographie ; les points spéculaires blancs sont causés par le flash de l'appareil photographique.

3.1.3 Discussion

Le fait que notre méthode n'emploie qu'une seule capture de la scène pour toutes les gouttes, induit une légère approximation par rapports aux lois physiques, mais permet un important gain de performance. Cette capture n'est pas réalisée pour chacune des gouttes de manière individuelle, mais une seule fois pour toutes les gouttes. Elle n'indique donc pas avec grande précision ce que les gouttes « voient » réellement. Dans certains cas, cela peut engendrer des occlusions non détectées, ou une distorsion accrue lors du plaquage de la texture sur la goutte. Dans le cadre de la simulation de pluie, cette approximation est un inconvénient tout à fait mineur, compte tenu de la faible taille et de la haute vitesse des particules représentées.

Dans notre approche, nous utilisons l'hypothèse (basée sur le calcul du facteur de Fresnel) que la réflexion peut être négligée du fait de sa contribution mineure à l'ap-

parence des gouttes. Cependant, aux angles rasants, cette contribution augmente ; comme expliqué en section 3.1.1, la couronne la plus externe de pixels d'une goutte rendue par notre méthode sera donc moins réaliste, mais n'aura qu'un impact limité sur la perception visuelle de l'utilisateur.

Pour obtenir des résultats rigoureusement exacts d'un point de vue physique, il faudrait effectuer un lancer de rayons vers chacun des objets présents dans la scène, ce qui peut difficilement être réalisé en temps-réel avec le matériel actuel.

3.2 Extension : persistance rétinienne

Dans la section précédente, nous avons défini un modèle général pour la simulation de pluie, qui ne prend pas en compte la perception de l'observateur. L'œil humain n'est pas habitué à observer des gouttes sphériques ; notre modèle, bien que basé sur les propriétés physiques des gouttes d'eau, semble ainsi manquer de réalisme. Nous avons donc étendu ce modèle afin qu'il prenne en compte le phénomène de persistance rétinienne, et permette le rendu de « cordes » conformes à la perception humaine en utilisant notre modèle de réfraction.

3.2.1 Problématique

Du fait de la persistance rétinienne, un œil humain, comme une caméra, perçoit généralement les gouttes de pluie comme des « cordes », allongées verticalement, comme l'illustre la figure 3.6. Deux phénomènes distincts peuvent provoquer cet effet : une caméra vidéo capture ses images en temps discret, tandis que l'œil humain fonctionne en temps continu.

Sur une photographie ou une séquence vidéo, les gouttes apparaissent allongées à cause de la vitesse d'obturation de l'appareil. Pendant que l'obturateur est ouvert, la goutte continue sa chute et laisse son empreinte sur une courte bande verticale de la pellicule. C'est effet est connu sous le nom de *flou de mouvement* (*motion blur* dans la littérature anglo-saxonne).

L'œil humain fonctionne différemment, pour le même résultat. L'œil n'a évidemment pas d'obturateur au sens photographique du terme ; cependant, lorsqu'une

Fig. 3.6 – Photographie de pluie ; les gouttes apparaissent sous forme de cordes

image se forme sur la rétine, il lui faut 60 à 90 ms pour s'effacer. Pendant cet intervalle de temps, la goutte continue sa chute, formant ainsi une séquence continue d'images sur la rétine. On parle dans ce cas de *persistance rétinienne*.

Les techniques conventionnelles de simulation du flou de mouvement consistent à mélanger plusieurs images successives de l'objet en mouvement, en leur affectant une certaine transparence. L'objet observé apparaît ainsi comme ayant des contours flous, et laissant une traînée derrière lui. Cependant, ces techniques ne sont pas applicables dans le cas qui nous intéresse. En effet, elles supposent que le déplacement de l'objet observé soit suffisamment restreint (en terme de pixels affichés) pour que deux images

successives de l'objet couvrent une zone commune significative de l'écran par rapport à la taille de l'objet. Cette assertion s'effondre dans le cas de gouttes d'eau, à la fois très rapides, et de dimensions très réduites. Une goutte donnée aura parcouru dans l'image un déplacement équivalent à plusieurs fois sa hauteur d'une image à la suivante, ce qui rend les techniques basées sur un mélange d'images inopérantes.

Nous avons donc développé une méthode spécifique à la simulation de la persistance rétinienne dans le cas de gouttes de pluie.

3.2.2 Technique proposée

Pour simuler le phénomène de persistance rétinienne, nous avons tout d'abord modifié la forme des particules rendues, que nous avons allongées verticalement. La hauteur d'une corde est un paramètre contrôlable par l'utilisateur, initialisé à une longueur de départ prenant en compte le diamètre de la goutte (et donc implicitement sa vitesse, d'après la table 2.2).

Oscillations périodiques des gouttes en chute libre

Au cours de sa chute, une goutte d'eau ne reste pas figée sous la forme décrite par l'équation 2.1 et illustrée par les masques présentés en section 3.1.2. Elle subit des oscillations, à la fois dans le plan horizontal et dans le plan vertical, illustrées sur la figure 3.7.

Frohn et Roth ([FR00]) ont montré que ces oscillations peuvent être modélisées sous la forme d'une combinaison d'harmoniques sphériques :

$$r(t,\theta,\phi) = r_0 \left(1 + \sum_{n,m} A_{n,m} * sin(\omega_n * t) * P_{n,m}(\theta) * cos(m\phi)\right) \tag{3.3}$$

où

- r_0 est le rayon de la goutte avant distorsion.
- $A_{n,m}$ est l'amplitude de l'harmonique sphérique de coefficients (n,m).
- $P_{n,m}(\theta)$ est le polynôme de Legendre décrivant la dépendance de la forme à l'angle θ pour les coefficients (n,m) :

$$P_l^m(X) = \frac{(-1)^m}{2^l * l!} * (1 - X^2)^{m/2} * \frac{\partial^{m+l}}{\partial X^{m+l}}(X^2 - 1)^l$$

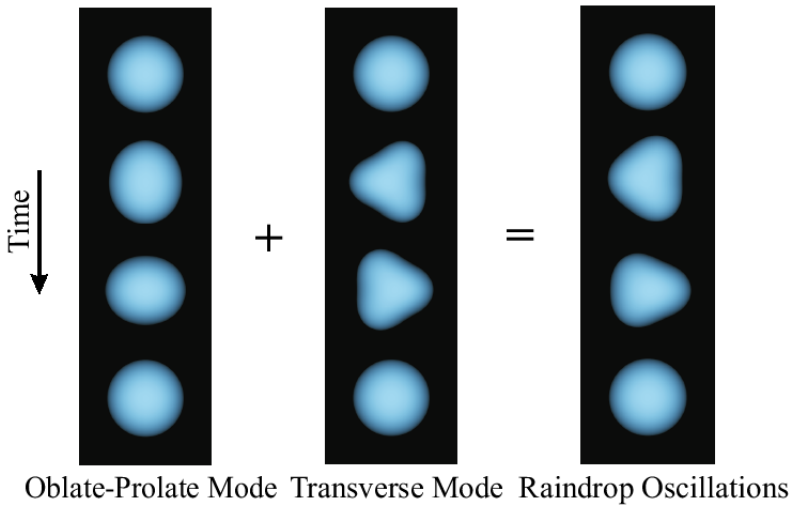

Fig. 3.7 – Oscillations d'une goutte d'eau en chute libre ([GN06])

– ω_n est la fréquence d'oscillation, calculée par l'équation :

$$\omega_n = \frac{2\pi n(n-1)(n+2)\sigma}{4\pi^2 \rho r_0^3}$$

σ étant la tension de surface ($\sigma = 0.0728 N/m^2$) et ρ la densité de l'eau ($\rho = 1000 kg/m^3$).

Garg et Nayar ([GN06]), après avoir noté que les oscillations se produisent presque exclusivement dans les deux plans illustrés par la figure 3.7, de coefficients $(n = 2, m = 0)$ pour les oscillations verticales, et $(n = 3, m = 1)$ pour les oscillations transversales, ont proposé une simplification de l'équation 3.3 sous la forme suivante :

$$r(t, \theta, \phi) = r_0\Big(1 + A_{2,0} * sin(\omega_2 t) * P_{2,0}(\theta) + A_{3,1} * sin(\omega_3 t) * cos(\phi) * P_{3,1}(\theta)\Big) \quad (3.4)$$

Une étude empirique a permis à Garg et Nayar de déterminer des valeurs de $A_{2,0}$ et $A_{3,1}$ permettant à leurs simulations d'adopter un comportement très proche de données réelles. Ils proposent deux couples, $(A_{2,0}, A_{3,1}) = (0.1, 0.1)$ et $(A_{2,0}, A_{3,1}) = (0.2, 0.1)$.

Tous les paramètres étant définis, nous avons pu utiliser l'équation 3.4 pour délimiter les contours de gouttes oscillantes soumises à la persistance rétinienne.

Après avoir expérimenté les deux couples $(A_{2,0}, A_{3,1})$ proposés par Garg et Nayar,

nous avons retenu le premier, $(A_{2,0}, A_{3,1}) = (0.1, 0.1)$, dont le comportement nous a semblé visuellement plus convaincant.

Nous avons modifié le *pixel shader* chargé de définir l'apparence des gouttes, afin qu'il établisse en premier lieu le contour des gouttes. Seule la forme de l'enveloppe externe nous intéresse ici. Nous avons donc pu fixer la valeur de θ à $\pi/2$ (calcul du rayon dans le plan horizontal).

ϕ est l'angle suivant lequel est mesuré le rayon, par rapport à la direction d'observation, dans le plan horizontal. N'importe quelle valeur permet donc d'obtenir des résultats physiquement plausibles, mais il importait que cette valeur soit différente d'une goutte à l'autre, afin d'éviter que toutes les gouttes ne suivent les mêmes contours, provoquant ainsi des similitudes visibles et manquant de réalisme à l'écran. Deux valeurs sont à calculer pour ϕ : celle délimitant le profil « gauche » de la goutte, et celle délimitant son profil « droit ». Nous avons lié le paramètre ϕ_{gauche} à la position de la goutte à l'écran, et défini $\phi_{droit} = \phi_{gauche} + \pi$. Ceci assure que les contours sont calculés en deux points diamétralement opposés, et différents d'une goutte à l'autre. Nous avons enfin remplacé le paramètre temporel t par l'altitude dans l'espace de la goutte (liée au temps par la vitesse de chute) afin de garantir la cohérence des profils des gouttes d'une image à l'autre. Ayant ainsi simplifié les calculs en fixant la valeur de certaines variables, le calcul du profil des gouttes peut se faire en temps-réel sans affecter les performances de la méthode.

Méthode de rendu

Une fois défini le contour de la goutte soumise au phénomène de persistance rétinienne, notre *pixel shader* doit désormais remplir ce contour. Pour cela, on considère que chaque pixel situé à l'intérieur du contour reçoit la contribution de plusieurs positions successives de la goutte, chacune à un stade plus ou moins avancé de sa chute, et épousant les contours délimités, comme illustré sur la figure 3.8.

Chaque corde est obtenue par le déplacement d'une seule et unique goutte ; ceci implique que les cordes doivent apparaître semi-transparentes. Ce type de rendu nécessite en général un tri des particules de l'arrière vers l'avant, coûteux en temps de calcul. Pour nous en dispenser, nous simulons cette baisse d'opacité en procédant

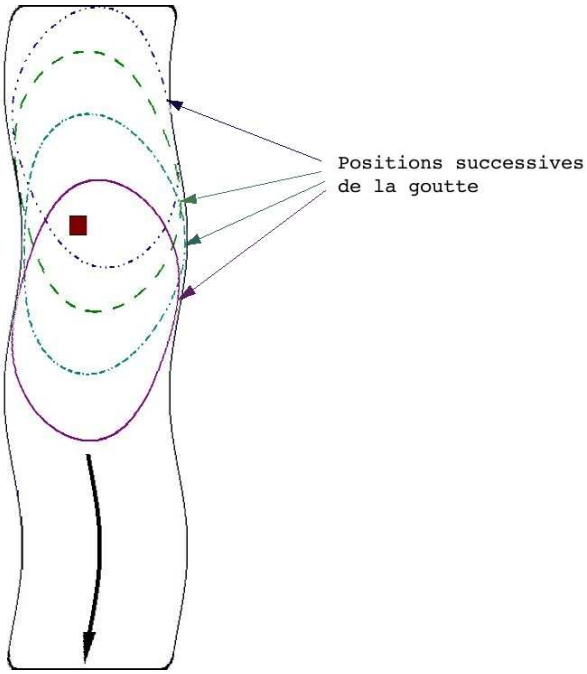

Fig. 3.8 – Le pixel marqué en rouge reçoit la contribution de chacune des positions successives de la goutte.

au mélange de la couleur moyenne des contributions des positions successives de la goutte, et de la couleur de l'arrière-plan, extraite de la texture d'environnement par l'équation 3.1.

Nous avons modifié le *pixel shader* utilisé pour l'affichage des gouttes, de manière à ce qu'il opère de la façon suivante pour chaque pixel :

- Calcul des contours de la goutte, et détermination de l'appartenance du pixel à ce contour.
- Calcul du pixel réfracté pour plusieurs positions successives de la goutte, suivant la méthode décrite dans la section 3.1.2.
- Calcul de la moyenne de ces valeurs, qui est la couleur affichée pour le pixel traité.
- Diminution de l'opacité de la particule par combinaison avec la texture d'environnement.

Cette méthode fournit des résultats satisfaisants ; son impact sur les performances de l'application est intimement lié au nombre d'échantillons calculés, i.e. au nombre

de positions successives de gouttes utilisées pour chaque pixel. Expérimentalement, un nombre compris entre 5 et 15 échantillons s'est avéré être un bon compromis, offrant des résultats visuels satisfaisants tout en ayant un impact limité sur les performances. En deçà de 5 échantillons, des artefacts sont visibles ; au delà de 15, les temps de calcul peuvent compromettre le caractère temps-réel de la méthode lorsque beaucoup de particules (>20000) sont affichées.

3.2.3 Discussion

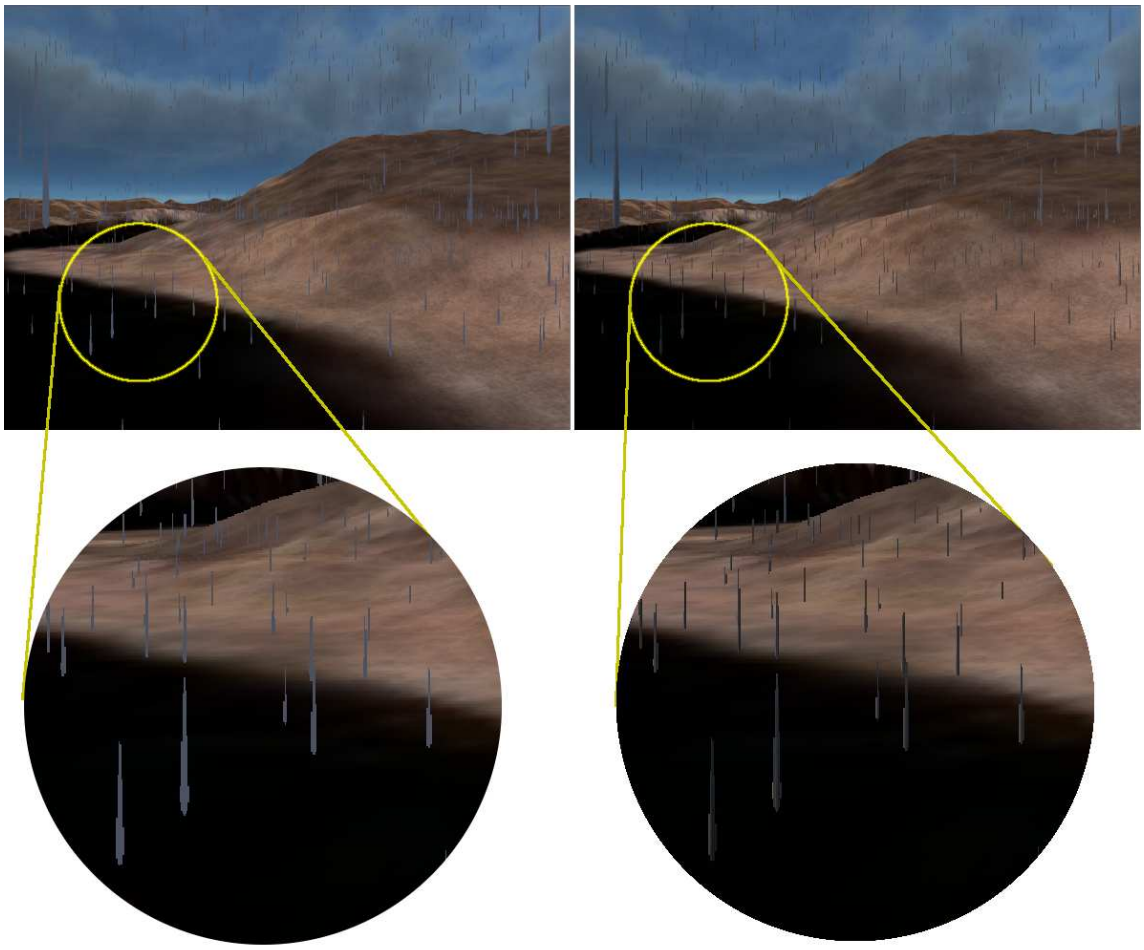

Fig. 3.9 – **Gauche** : une couleur unique pour toutes les cordes de la scène. **Droite** : avec notre méthode. Les agrandissements de la ligne du bas montrent que notre méthode modifie la teinte des gouttes, sans même qu'il n'y ait de déplacement dans la scène.

Bien que la réfraction ne soit plus visible dans les gouttes affichées par le biais de notre modèle de gestion de la persistance rétinienne, notre approche offre l'avantage de permettre aux gouttes d'adopter automatiquement la teinte et la luminosité adaptées, quelle que soit la scène, améliorant ainsi la qualité visuelle du rendu de la pluie pour un surcoût limité.

La figure 3.9 souligne l'intérêt de cette extension, en comparant des cordes générées par notre méthode à des cordes de couleur fixe. Les images de droite utilisent notre méthode, tandis que les images de gauche ont été générées en utilisant la même couleur pour toutes les particules. Cette couleur a été choisie de sorte à ce que les différences entre les deux images dans le coin haut droit soient minimales. On peut clairement observer des différences entre les images dans le coin bas gauche, dont un agrandissement est présenté sur la ligne du bas. En cas de mouvement dans la scène, ou de changement dans la composition de la scène, la différence entre l'apparence des gouttes générée par notre méthode et celle obtenue par une texture ou une couleur unique serait évidemment encore plus importante. Cette différence justifie le surcoût en temps de calcul (de l'ordre de 20%) lié à l'emploi de l'extension proposée.

3.3 Extension : interaction pluie/lumière

3.3.1 Problématique

Lorsque la pluie tombe devant des phares de voitures, ou des lampadaires, les gouttes présentent des reflets de la couleur de ces sources lumineuses, comme l'illustre la figure 3.10. Nos travaux se placent dans le double cadre des jeux vidéos, et des simulateurs dédiés à la sécurité routière. Dans ce second domaine, les conditions de circulation très dégradées, telles que la conduite de nuit par temps de pluie, sont étudiées avec une attention particulière. Un modèle de simulation réaliste de l'apparence des gouttes par luminosité réduite et en présence de sources lumineuses s'avère donc crucial.

Les lois de l'optique présentées en section 2.2 restent bien évidemment valides lorsqu'une source lumineuse est incluse dans la scène. Lorsque l'observateur se situe à

Fig. 3.10 – Photographie de pluie éclairée par les phares d'un véhicule

proximité d'une source lumineuse, les rayons émanant de cette source ont cependant une intensité lumineuse bien plus forte que ceux émanant d'autres points de la scène, et influent de manière non négligeable sur l'apparence de la goutte.

En utilisant la méthode décrite en section 3.1.2, une source lumineuse positionnée derrière l'observateur n'aurait aucune influence sur l'apparence des gouttes, notre modèle ne prenant pas en charge la réflexion. Cependant, dans ce cas, les réflexions internes et externes ne peuvent plus être ignorées, car leur participation à l'apparence de la goutte ne peut être négligée dans ce cas précis.

Nous avons ajouté une seconde extension à notre modèle d'apparence des gouttes d'eau, afin qu'il gère l'illumination par des sources lumineuses.

3.3.2 Analyse du problème

Le modèle défini en section 3.1.2 nous a poussé à utiliser ici une approche phénoménologique. En effet, le calcul exact de plusieurs réflexions internes en temps-réel dans une goutte d'eau est un problème complexe ; il devient encore bien plus com-

plexe lorsque l'on ne dispose pas d'un maillage géométrique pour les particules, représentées dans notre modèle par des imposteurs.

Afin de définir un modèle d'illumination des gouttes réaliste, nous avons cherché à savoir s'il était possible de catégoriser l'impact des réflexions sur l'apparence d'une goutte d'eau. Dans ce but, nous avons étudié à l'aide d'un lancer de rayons les trajectoires empruntées par les rayons lumineux subissant jusqu'à 3 réflexions internes (à ce stade, la majeure partie de l'intensité lumineuse a été réfractée hors de la goutte).

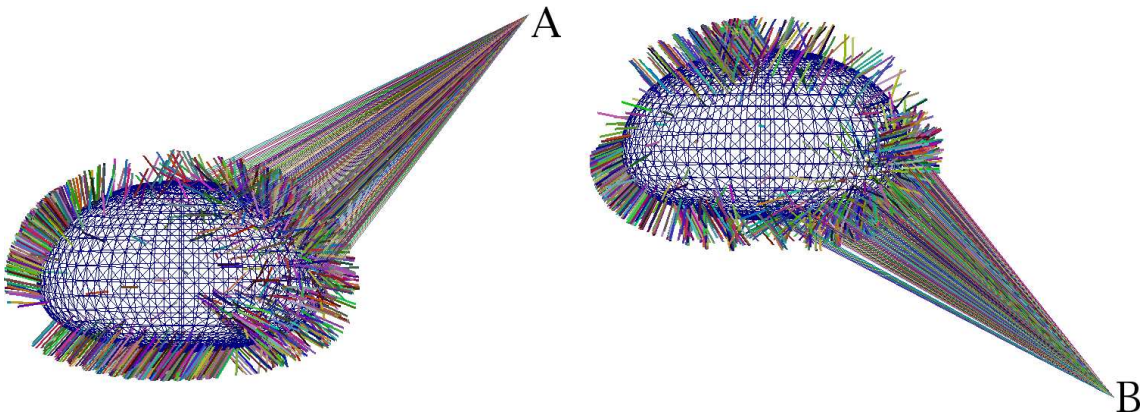

Fig. 3.11 – Répartition spatiale des rayons lumineux sortant d'une goutte de rayon 3.5 mm. les rayons partent des sources A et B, pénètrent la goutte par la droite, subissent jusqu'à trois réflexions internes, puis sont réfractés hors de la goutte.

La figure 3.11 met en évidence les directions empruntées par les rayons lumineux au travers d'une goutte d'eau. Les bâtonnets colorés indiquent les directions suivies par les rayons réfractés hors de la goutte après avoir subi jusqu'à 3 réflexions internes. Les résultats obtenus nous ont permis de constater que ces directions peuvent être catégorisées en trois groupes :

- Rayons renvoyés dans la direction du rayon entrant initial (réflexion externe, ou après 3 réflexions internes).
- Rayons réfractés du côté opposé de la goutte, dans la direction d'incidence.
- Rayons réfractés du côté opposé de la goutte, perpendiculairement à la direction d'incidence.

Il ressort également que très peu de rayons sont réfractés dans les directions latérales.

Le calcul d'un petit nombre de réflexions (internes et externes) des rayons lumineux serait de toute évidence le meilleur moyen pour obtenir des images physiquement satisfaisantes ; il peut cependant difficilement être effectué dans une optique de rendu temps-réel. De plus, il serait également difficilement employable avec le modèle d'imposteurs que nous utilisons.

En se basant sur la répartition des rayons réfractés présentée ci-dessus, nous avons chercher à simuler l'interaction entre les gouttes et les sources de lumière par une approche phénoménologique.

3.3.3 Modèle proposé

L'idée maîtresse qui a guidé la conception de notre modèle phénoménologique, est de modifier la couleur des pixels de chaque goutte, en fonction de la distance séparant la goutte de la source lumineuse.

Après avoir catégorisé le comportement de la lumière à l'intérieur d'une goutte d'eau, nous avons établi un ensemble d'équations empiriques, chacune visant à simuler l'une des directions principales empruntées par les rayons réfractés hors de la goutte.

Les rayons sortant de la goutte en direction de l'utilisateur doivent de toute évidence apparaître plus intenses que les autres. Ce premier point est pris en charge par l'équation suivante :

$$A_1 = C_L * Att * \overrightarrow{V_{P->E}} \cdot \overrightarrow{V_N} \tag{3.5}$$

Où :

- C_L est la couleur de la source lumineuse.
- Att est un facteur d'atténuation, basé sur la distance entre la source lumineuse et le pixel considéré. Il permet également de prendre en compte les conditions atmosphériques (brouillard diminuant la portée de la source lumineuse).
- $\overrightarrow{V_{P->E}}$ est la vecteur allant du pixel vers l'observateur.
- $\overrightarrow{V_N}$ est la normale à la goutte au pixel considéré.

A_1 définit la modification de couleur maximale qui pourra être appliquée au pixel, en donnant plus d'intensité aux pixels dont la normale fait face à l'observateur.

Les rayons sortants peuvent être engendrés par une réflexion externe, et donc retourner dans la direction du rayon initial. L'équation suivante assure la gestion de ce comportement :

$$A_2 = max(0, \overrightarrow{V_{P->L}} \cdot \overrightarrow{V_N}) \tag{3.6}$$

Où :

– $\overrightarrow{V_{P->L}}$ est le vecteur allant du pixel vers la source lumineuse.

Lorsque le rayon lumineux retourne dans sa direction d'origine, le produit scalaire présent dans l'équation 3.6 est positif. Cela garantit que les rayons renvoyés de la manière la plus directe seront ceux qui recevront le plus d'énergie.

Les rayons sortant peuvent également être le résultat d'une ou deux réflexions internes, et sortir de la goutte perpendiculairement au rayon incident, dans le plan vertical d'incidence. Ce comportement est pris en charge par l'équation suivante :

$$A_3 = max(0, (-\overrightarrow{V_{P->L_{2D}}}) \cdot \overrightarrow{V_{N_{2D}}} * (1 - (-\overrightarrow{V_{P->L}}) \cdot \overrightarrow{V_N})) \tag{3.7}$$

Où :

– $\overrightarrow{V_{P->L_{2D}}}$ est la projection normalisée de $\overrightarrow{V_{P->L}}$ dans le plan horizontal.

– $\overrightarrow{V_{N_{2D}}}$ est la projection normalisée de $\overrightarrow{V_N}$ dans le plan horizontal.

A_3 est employé pour les réflexions internes. Le produit scalaire dans le plan horizontal permet d'éviter que les rayons ne soient réfractés latéralement, ce qui ne doit pas être le cas d'après la figure 3.11. Le deuxième produit scalaire présent dans l'équation 3.7 favorise les rayons sortant de la goutte perpendiculairement au rayon incident, vers le haut ou le bas.

Finalement, la combinaison des équations 3.5, 3.6 et 3.7 forme le modèle d'interaction goutte-source lumineuse utilisé dans notre approche :

$$C_F = (C_O * C_{amb}) + \sum_{sources} \Big(A_1 * (A_2 + A_3) \Big) \tag{3.8}$$

Où :

– C_F est la couleur finale du pixel.

– C_O est la couleur extraite de la texture d'environnement.

– C_{amb} est la couleur de la lumière ambiante de la scène.

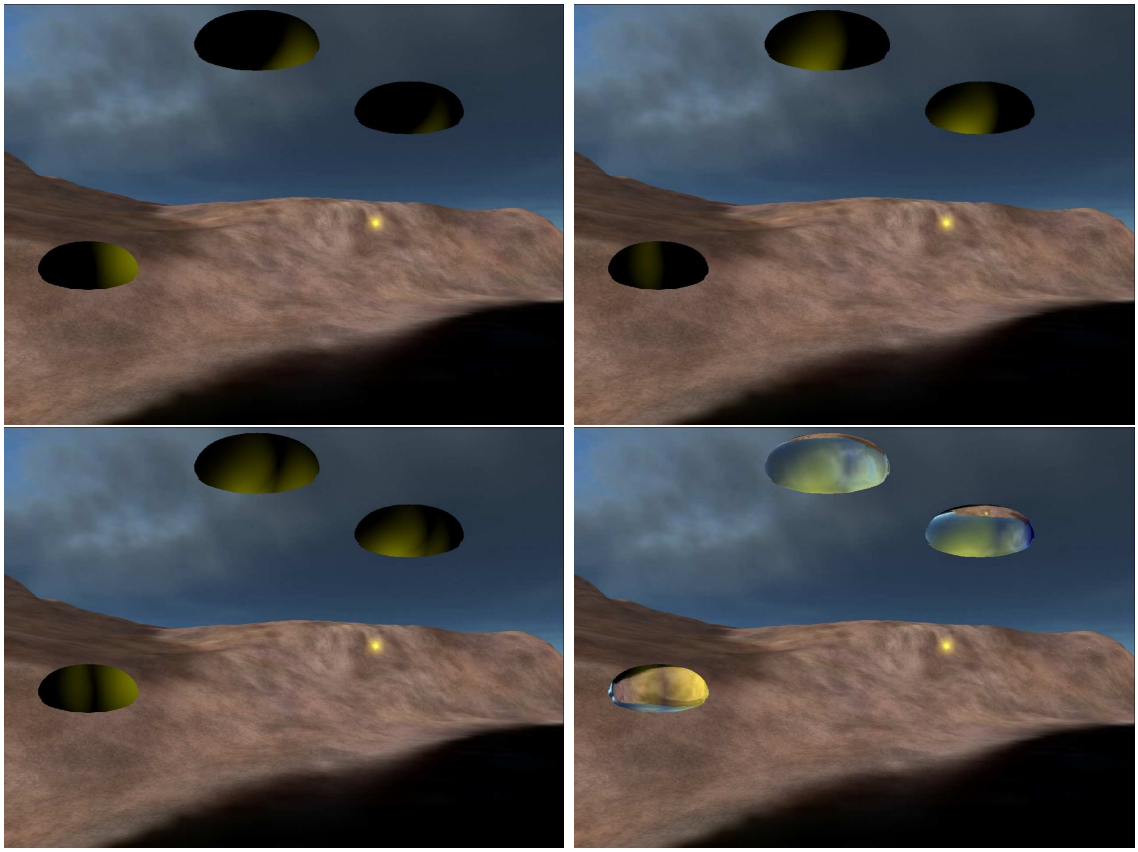

Fig. 3.12 – **Haut-gauche** : réflexion externe $(A_1 * A_2)$. **Haut-droit** : réflexion interne $(A_1 * A_3)$. **Bas-gauche** : somme des réflexions internes et externes $\Big(A_1 * (A_2 + A_3)\Big)$. **Bas-droit** : résultat final.

– La somme permet de prendre en compte plusieurs sources lumineuses simultanément.

La figure 3.12 illustre la contribution de chacun des termes de l'équation 3.8 sur des gouttes démesurées pour l'exemple, illuminées par une source ponctuelle de couleur jaune.

3.3.4 Discussion

Pour simuler l'illumination de gouttes d'eau par des sources lumineuses, nous avons proposé un modèle phénoménologique offrant des résultats visuellement satisfaisants, illustrés par la figure 3.13. Notre implantation de cette technique permet de gérer plusieurs sources ponctuelles simultanément sans diminution notable des per-

formances (la vitesse d'affichage est diminuée de l'ordre de 1% dans le cas du modèle de réfraction). Ce modèle peut être utilisé avec ou sans l'extension de persistance rétinienne décrite en section 3.2.

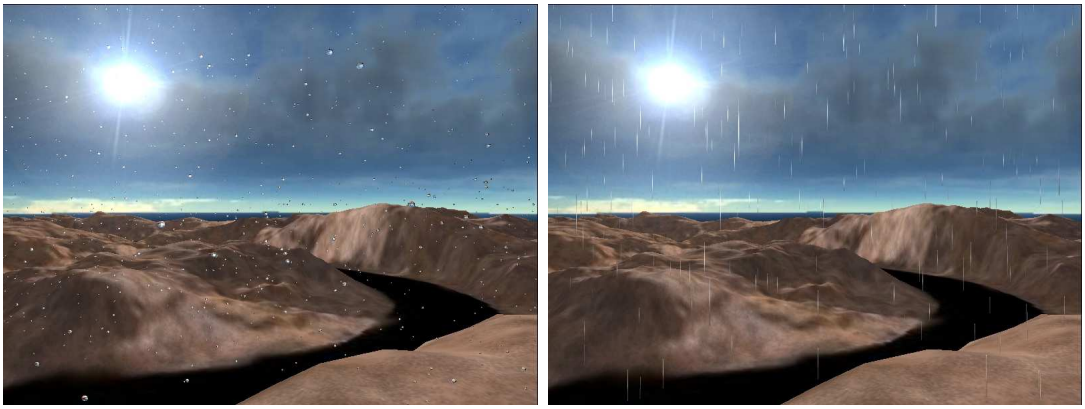

Fig. 3.13 – Notre modèle d'illumination. **Gauche** : appliqué à 10000 gouttes de rayon $1mm$. **Droite** : appliqué à 4000 cordes de rayon $0.5mm$.

Ce modèle reste cependant fondé sur des constatations empiriques, et ne livre qu'une approximation visuellement plausible de l'apparence d'une goutte placée dans le champ d'une source lumineuse. Un modèle effectuant un calcul basé sur les lois de la physique de plusieurs réflexions internes des rayons lumineux, offrirait à n'en pas douter de meilleurs résultats visuels ; il ne serait cependant pas compatible avec les contraintes de rapidité d'une application temps-réel, ni avec le modèle d'imposteur utilisé pour nos gouttes d'eau. Notre modèle phénoménologique se présente donc comme un compromis satisfaisant, atteignant des résultats visuellement convaincants sans impliquer un surcoût prohibitif.

3.4 Conclusion

L'étude des propriétés optiques des gouttes d'eau réalisée dans le chapitre précédent nous a permis de concevoir un modèle fonctionnant en temps-réel pour l'affichage de gouttes de pluie. Ce modèle, centré sur la simulation de la réfraction observable dans une goutte d'eau, présente toutefois un certain manque de réalisme visuel.

Pour remédier à ce manque, nous avons étendu notre modèle afin qu'il gère le phénomène de persistance rétinienne. Cette extension permet aux gouttes de prendre l'apparence de cordes, conformément à la perception humaine, et prend en compte les oscillations périodiques subies par les gouttes au cours de leur chute.

Nous nous sommes ensuite intéressés à l'apparence des gouttes d'eau placées dans un champ lumineux. Ce problème présente des enjeux cruciaux dans le cadre de la sécurité routière. Afin de satisfaire aux contraintes de notre application, une approche phénoménologique a été retenue pour étendre notre modèle de goutte d'eau. Par cette approche, nous avons proposé un modèle permettant de prendre en compte les modifications de l'apparence des gouttes causées par des sources lumineuses.

Une nuit où je rêvais, couché dans l'herbe, intégralement exposé aux cieux, je fus pris dans une averse. C'était l'une de ces pluies cataclysmiques où le ciel soudain se déchire en déversant des trombes d'eau, dans une prodigieuse furie sonore.

Moon Palace
Paul Auster

Chapitre 4

Simulation de la dynamique des gouttes sur la carte graphique

Le chapitre précédent a introduit le modèle de rendu des gouttes de pluie que nous avons proposé. Ce modèle présente toutefois un intérêt limité s'il n'est pas animé.

Ce chapitre vise à introduire le modèle d'animation que nous avons développé, fonctionnant entièrement sur la carte graphique de l'ordinateur. Diverses optimisations ont été introduites par rapport aux modèles génériques de systèmes de particules utilisant le matériel graphique, et seront présentées dans ce chapitre.

Une fois notre modèle opérationnel, nous l'avons complété afin de simuler certains phénomènes pouvant influer sur la dynamique des gouttes de pluie. L'un des phénomènes les plus visibles est la modification de la trajectoire des gouttes engendrée par le vent. Pour modéliser ce phénomène, nous avons cherché à définir une méthode peu coûteuse en temps de calcul, mais aussi à spécifier des outils intuitifs permettant à un utilisateur de simplement caractériser et contrôler la « forme » du vent, tout en conservant une approche phénoménologique. Nous avons également défini un modèle simple de détection et de réponse aux collisions, peu coûteux en temps de calcul, afin de proposer un système complet de simulation de pluie. Ces deux extensions seront également présentées dans ce chapitre.

4.1 Système de particules sur la carte graphique

Pour animer nos gouttes de pluie, nous avons utilisé un système de particules géré par la carte graphique. Cette section a pour objectif d'introduire les bases de cette technique, les détails des textures de position et de vélocité que nous avons employées, le modèle de « boîte à gouttes » que nous avons mis en œuvre pour optimiser l'affichage des gouttes, et le processus de formation des imposteurs à l'écran.

4.1.1 Description

L'animation de pluie est un cas d'école pour les systèmes de particules. Dans notre application, les divers tests effectués ont montré que plusieurs milliers de gouttes sont requis pour obtenir une impression réaliste de pluie. La gestion du système de particules sur le processeur central de l'ordinateur implique la nécessité d'un transfert de toutes les particules vers la carte graphique pour chaque image rendue. Ce transfert provoque un goulet d'étranglement, et réduit drastiquement les performances dès lors que le nombre de particules augmente.

Pour outrepasser cette limite, nous avons utilisé l'approche décrite dans [KLRS04] et [KSW04]. Ces articles proposent des implantations de systèmes de particules basées sur les possibilités des cartes graphiques actuelles. Les positions des particules sont stockées dans une texture, actualisée par l'intermédiaire d'un *pixel shader*. Cette technique permet d'éviter la majorité des transferts entre la mémoire centrale et la carte graphique, toute la simulation dynamique se déroulant sur la carte graphique.

Notre implantation de cette fonctionnalité diffère légèrement de celles décrites dans [KLRS04] et [KSW04]. Nous avons ciblé notre travail sur la simulation de pluie, et n'avons donc pas utilisé un système aussi général que ceux proposés dans ces articles. Nous avons fait l'hypothèse que les particules partagent la même vitesse, et que les collisions entre particules peuvent être ignorées.

Les positions des gouttes sont stockées dans une texture en point flottant (dont la résolution dépend du nombre de particules désiré). Chaque pixel de la texture indique la position d'une particule. La formation de cette texture est détaillée en section 4.1.2.

Nous considérons que les particules évoluent dans une « boîte à gouttes », qui se déplace avec l'utilisateur. Un positionnement judicieux de cette boîte permet que la plupart des gouttes soient à tout instant dans le champ de vision de l'observateur, maximisant ainsi l'efficacité du système. Le fonctionnement de cette boîte est décrit en section 4.1.4.

Le processus de simulation peut être résumé de la manière suivante :

- Au lancement de l'application, les particules nécessaires sont stockées dans la mémoire de la carte graphique.
- Mise à jour de la texture de position grâce à un *pixel shader*.
- Pour chaque particule, modification de sa position dans un *vertex shader*, en utilisant la texture de position.
- Rendu de chaque particule, en utilisant la technique de rendu décrite dans le chapitre précédent.

4.1.2 Texture de position

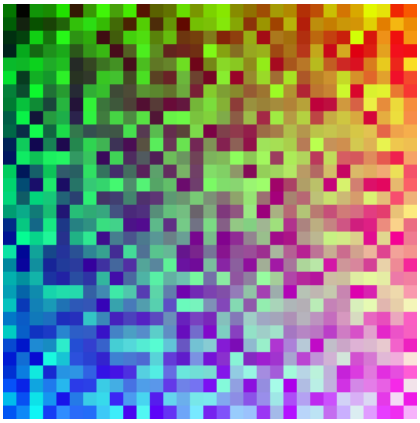

Fig. 4.1 – Exemple de texture de position, ramenée à l'espace affichable [0-1]

Dans la texture de position, les composantes rouge, vert et bleu de chacun des pixels correspondent aux coordonnées X, Y et Z d'une particule. La résolution de cette texture est guidée par le nombre de particules souhaité : elle sera par exemple de 100x100 pixels si 10000 particules évoluent dans le système. Cette texture utilise des coordonnées en point flottant, occupant 32 bits par canal. Chaque canal n'est donc

pas limité à l'intervalle [0-1], et les valeurs des positions peuvent ainsi être stockées sans changement d'échelle (même si elles subissent une translation de repère, détaillée en section 4.1.4). La figure 4.1 illustre un exemple de texture de position, ramenée ici à l'espace affichable [0-1].

Lors de la mise à jour de cette texture, le déplacement effectué depuis l'image précédente est ajouté aux coordonnées de chaque pixel. Ce déplacement est calculé à l'aide d'une seconde texture, indiquant la vélocité de chaque particule.

4.1.3 Texture de vélocité

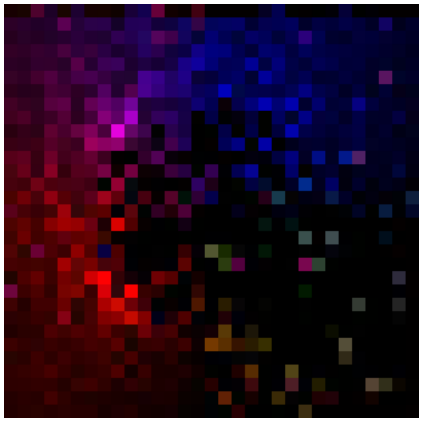

Fig. 4.2 – Exemple de texture de vélocité.

Afin de mettre à jour les positions des particules, nous employons une texture complémentaire de la texture de position, désignée sous le nom de « texture de vélocité ». Elle définit la direction de chute et la vitesse de chaque particule, et permet la définition de comportements complexes, tels que l'advection des particules par un vent balayant la scène, ou encore la gestion des collisions et des rebonds.

Chaque pixel de cette texture enregistre la direction courante dans laquelle la particule évolue, sur ses composantes RVB ; cette direction n'est pas normalisée, et indique donc également l'amplitude du déplacement. Un exemple de texture de vélocité est proposé sur la figure 4.2. La nouvelle position de la particule est obtenue en utilisant l'équation suivante :

$$P_{nouvelle} = P_{precedente} + \Delta t * \overrightarrow{V}$$

où

- $P_{nouvelle}$ est la position de la particule calculée au pas de temps courant.
- $P_{precedente}$ est la position de la particule calculée au pas de temps précédent.
- Δt est l'intervalle de temps écoulé depuis le pas de temps précédent.
- $\overrightarrow{V}$ est la vélocité de la particule.

Le type de particule (goutte obtenue par le modèle de réfraction, ou corde simulant la persistance rétinienne) est enregistré dans le canal alpha de cette texture. Si l'utilisateur n'impose pas un type de rendu particulier, les cordes sont utilisées pour la plupart des particules, à l'exception de celles dont la vitesse est très réduite (typiquement, celles en train de rebondir), qui sont affichées sous la forme de gouttes obtenues par le modèle de réfraction. Ces types de rendus sont présentés en détails dans le chapitre 3.

Du fait de contraintes matérielles, une texture ne peut être utilisée simultanément en entrée (lecture de données) et en sortie (écriture de données) d'un calcul par la carte graphique. Nous avons donc défini deux textures pour la position, ainsi que deux pour la vélocité. Ces deux textures se répondent dans une approche « ping-pong » : l'une est utilisée en lecture pendant que l'autre l'est en écriture, les deux étant permutées entre chaque pas de calcul.

Les textures de position et de vélocité actives en écriture à un instant donné, sont mises à jour en une seule passe par un seul et même *pixel shader*. La fonctionnalité de « multiples cibles de rendu » (*Multiple Render-Target* dans la littérature anglo-saxonne), disponible sur les cartes graphiques récentes et permettant la mise à jour de plusieurs textures en une seule passe, est utilisée ici, économisant ainsi du temps de calcul.

4.1.4 Le modèle de « boîte à gouttes »

Dans une scène de pluie réelle, les particules tombent généralement en tout point de l'espace observable ; pour une pluie virtuelle, il est de toute évidence inutile d'animer et rendre des particules qui se trouvent hors du champ de vision de l'observateur.

Nous avons donc proposé de définir une contrainte particulière pour notre système, qui fait que nous considérons les particules comme évoluant à l'intérieur de

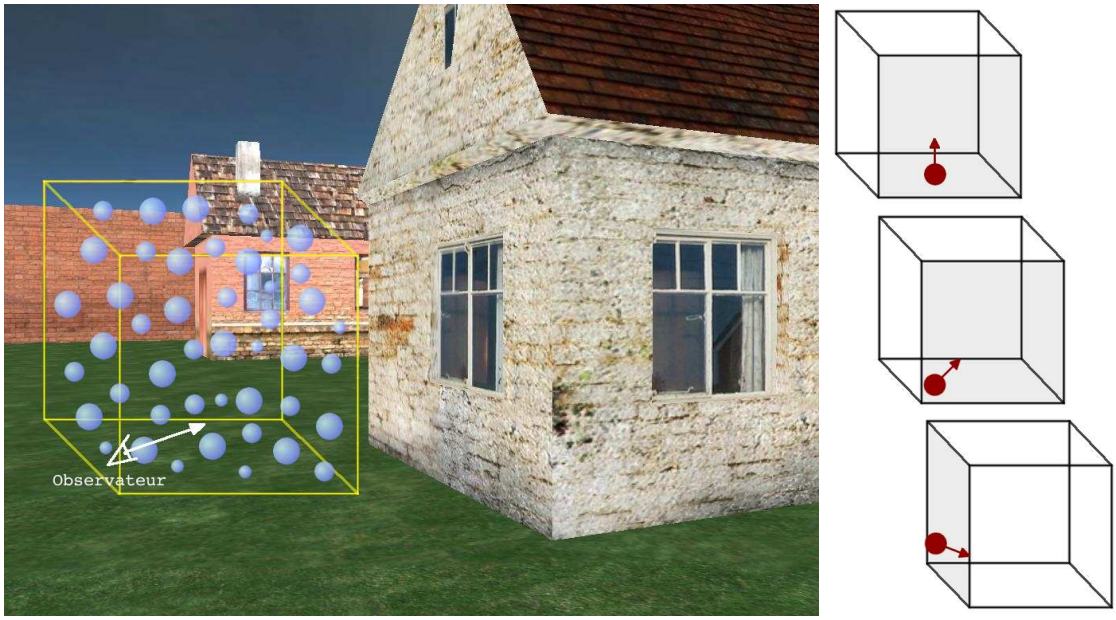

Fig. 4.3 – **Gauche** : Les particules ne tombent que dans la boîte à goutte, visible de l'observateur. **Droite** : Déplacement de la boîte lorsque l'observateur (en rouge) change son axe de vue. L'utilisateur ne se déplace pas, et reste en permanence en contact avec la face grisée de la boîte.

ce que nous appelons une « boîte à gouttes ». Cette boîte s'étend sur l'équivalent de 100m de côté dans une scène typique de notre application. Ses bords sont en permanence alignés avec les axes du repère monde. Elle se déplace avec l'utilisateur, et est disposée de sorte à ce que ce dernier soit en permanence sur un bord de la boîte, avec le regard tourné vers le centre. De cette manière, la majeure partie des particules se trouve à tout instant dans le champ de vision de l'observateur. La boîte est repositionnée à chaque déplacement de l'utilisateur, mais également à chaque changement de l'orientation de son axe de vision.

Les coordonnées des particules, stockées dans la texture de position, sont exprimées dans un repère dont l'origine est située au centre de cette boîte, ce qui implique qu'elles doivent être rectifiées à chaque déplacement de la boîte, afin que les particules semblent poursuivre leur chute naturellement dans un repère indépendant de l'observateur. Il suffit pour cela de soustraire aux positions des particules le déplacement subi par le centre de la boîte depuis l'image précédente, afin de compenser

les déplacements de la boîte induits par les mouvements de l'observateur.

Au lancement du programme, la texture de position est initialisée de sorte à ce que la boîte soit remplie de particules disposées sur une grille régulière dans le plan horizontal, et à une altitude aléatoire.

Chaque fois qu'une particule est amenée à sortir de la boîte, elle est immédiatement déplacée du côté opposé à celui de sa sortie. Ceci simplifie la gestion de la « naissance » et de la « mort » des particules. Les gouttes meurent lorsqu'elles quittent la boîte, et sont immédiatement recrées. Un cas particulier à cette règle est celui des gouttes sortant de la boîte par le bas (ou impliquées dans une collision, voir section 4.3). Dans ce cas, la goutte est replacée non pas au sommet de la boîte, mais à une altitude aléatoire comprise entre un seuil minimum et le sommet de la boîte. Ceci permet d'éviter un mouvement cyclique, dans lequel des trains de gouttes se répéteraient dans le temps, causant une similitude visuellement gênante. Le test de sortie de la boîte est extrêmement simple à mettre en œuvre, les positions des particules étant définies dans un repère lié à la boîte.

Par cette technique, un nombre minimal de particules est requis. Lorsque l'utilisateur se déplace, il est amené à « dépasser » des gouttes. Toutes celles ci, qui devraient alors se trouver dans son dos (donc hors de son champ de vision), sont dans ce cas replacées devant lui, à la distance maximale autorisée par les dimensions de la boîte, qui (en l'absence de vent et d'obstacles, voir sections 4.2 et 4.3), abrite ainsi en permanence une distribution homogène de particules, sans espace mort.

4.1.5 Formation des particules à l'écran

Une fois la texture de position mise à jour, les particules peuvent être affichées à l'écran. Les coordonnées des particules doivent alors être transférées de la texture de position vers une forme géométrique.

Contrairement aux articles ayant étudié la simulation de systèmes de particules sur la carte graphique ([KSW04], [KLRS04] et [KKKW05]), qui se contentent de mentionner son existence, nous avons fait usage de la fonctionnalité appelée *Vertex Texture Fetch* disponible sur les cartes graphiques compatibles avec le *Shader Model 3.0*. Cette fonctionnalité permet d'accéder à une texture à partir d'un *vertex shader*,

et nous l'avons utilisée pour transférer les positions des gouttes de la texture de position vers les coordonnées géométriques des particules.

Au lancement de l'application, les particules nécessaires sont stockées dans la mémoire de la carte graphique, sous forme de particules fixes réparties uniformément dans l'espace. Cette répartition permet simplement que le moteur que nous utilisons ait en permanence l'impression qu'au moins une de ces particules est dans le champ visible, faute de quoi il pourrait les masquer toutes pour optimiser les temps de rendu.

Un *vertex shader* est utilisé pour obtenir les positions effectives de chaque particule grâce à la texture de position, et les affecter aux coordonnées de chaque particule. Les particules sont ensuite transmises au *pixel shader* qui leur applique le type de rendu approprié (voir chapitre 3).

L'utilisation de deux textures pour la position et la vélocité implique qu'il est nécessaire après chaque pas de simulation, de signaler au *vertex shader* quelles sont les textures de position et de vélocité les plus récemment mises à jour.

Le placement d'une particule doit tenir compte de son orientation par rapport à l'observateur. La texture de position ne comprend qu'un seul ensemble de coordonnées par particule, les coordonnées de son centre. Une consistance est donnée aux particules dans le *vertex shader*, suivant la nature des particules, comme illustré sur la figure 4.4.

Dans le cas des gouttes visualisées à l'aide du modèle de réfraction, une fois le centre converti dans le système de coordonnées écran, l'imposteur symbolisant la goutte est « étalé » sur l'écran ; chaque coin de l'imposteur est décalé de sorte à ce qu'il soit en permanence dans le plan orthogonal à la direction de vue de l'observateur. Ainsi, les coordonnées du coin haut-gauche seront diminuées sur l'axe horizontal du repère lié à l'écran (le coin haut-gauche a pour coordonnées (0,0)) ainsi que sur l'axe vertical, d'une quantité dépendant de la taille souhaitée des gouttes.

Dans le cas des « cordes », les gouttes doivent sembler suivre un mouvement vertical, indépendamment des déplacements ou de la direction de visée de l'observateur. Les imposteurs sont donc étirés verticalement avant leur transformation en coordonnées écran (donc dans le repère monde), et sont étirés horizontalement après

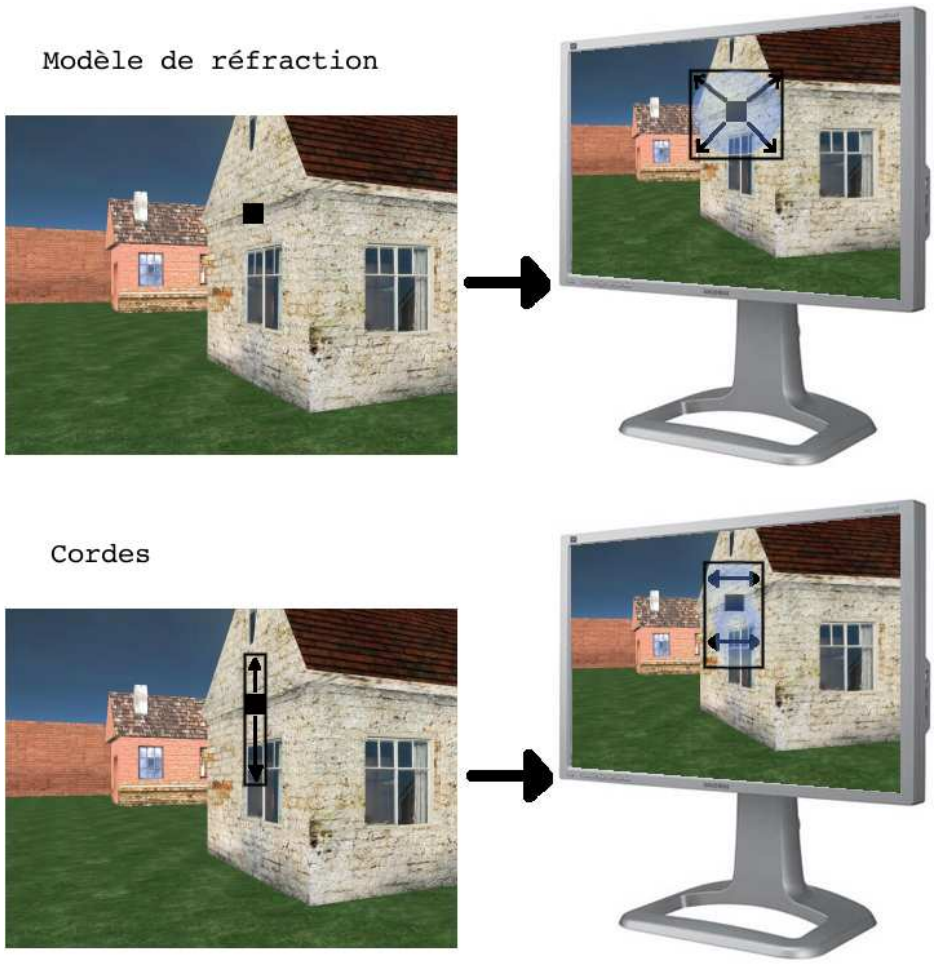

Fig. 4.4 – **Haut** : La consistance des particules est donnée après la projection dans l'espace écran, pour le modèle de réfraction. **Bas** : Pour le modèle de cordes, la goutte est étalée verticalement avant sa projection vers l'écran.

leur transformation. Ils restent ainsi en permanence dans un plan orthogonal à la direction de vue horizontale de l'observateur. Ces particules présentent le maximum de surface visible à l'observateur, tout en donnant l'impression de tomber à la verticale.

4.1.6 Discussion

Cette section a présenté les détails du système de particules que nous avons développé. Il s'agit d'une spécialisation des méthodes proposées dans [KLRS04] et [KSW04]. Nous avons appliqué ces méthodes au cas pratique de l'animation de pluie.

Notre modèle de boîte à gouttes permet de restreindre la zone dans laquelle évoluent les particules, tout en autorisant cette zone à se déplacer dans l'espace pour accompagner l'observateur dans ses mouvements. Hors des limites définies par la boîte, les particules tombent sous le seuil du pixel, et un modèle particulaire n'est plus adapté. Une approche multi-résolution pourrait être employée pour définir un modèle alternatif adapté à la représentation de pluie distante.

4.2 Extension du système de particules : gestion du vent

À ce stade de l'algorithme, l'animation manque encore de réalisme ; en effet, toutes les particules tombent suivant une seule direction et une seule vitesse, aucune force extérieure ne leur étant appliquée. Pour enrichir ce mouvement et le rendre plus réaliste, nous avons développé une interface permettant de modéliser simplement un vent basé sur des primitives simples, qui pourra influer sur les trajectoires des particules.

4.2.1 Texture de vent

Pour interagir avec le *pixel shader* chargé de la mise à jour des positions des particules, nous avons fait le choix d'enregistrer notre flux de vent tridimensionnel dans une texture, la direction du vent étant encodée dans les canaux rouges, verts et bleus de chaque pixel. Cette texture établit une grille régulière tridimensionnelle, de mêmes dimensions que la boîte à gouttes, et définissant dans chaque cellule de la grille la direction du vent.

Représenter le vent dans une texture amène naturellement à l'idée de mettre à jour cette texture par le biais d'un *pixel shader*. Les textures tridimensionnelles ne peuvent cependant pas être utilisées comme cibles de rendu, du moins en l'état actuel des avancées matérielles. Nous avons donc utilisé l'approche de « Textures 3D aplaties » (*Flat 3D textures* dans la littérature anglo-saxonne), introduite par Harris *et al.* dans [HBSL03]. Toutes les couches de profondeur de la texture 3D sont

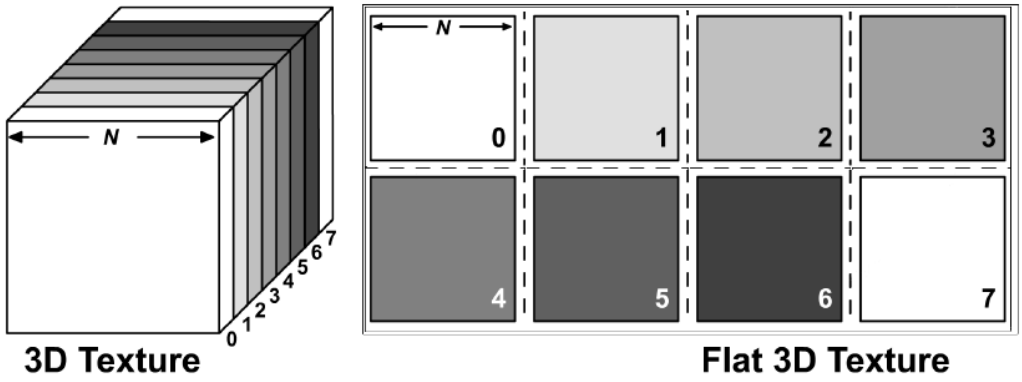

Fig. 4.5 – Texture 3D aplatie ([HBSL03])

placées côte à côte dans une texture 2D, comme illustré sur la figure 4.5. Une texture tridimensionnelle de résolution 4x4x4 peut ainsi être stockée sous la forme d'une texture bidimensionnelle de résolution 8x8. Cette simple conversion nous permet d'animer la texture de vent au travers d'un *pixel shader*, dans son intégralité et en une seule passe. La texture reste tridimensionnelle dans sa conception, même si elle est en pratique stockée sous la forme d'une texture bidimensionnelle.

Au cours de nos travaux, nous avons pu constater qu'une résolution de 64x64x64 s'avérait suffisante pour la texture de vent, et permettait d'obtenir des résultats visuellement satisfaisants et dépourvus d'artefacts. Cette texture subit une interpolation tri-linéaire dans le *pixel shader* chargé de la mise à jour des vélocités des particules, afin d'éviter des sauts de direction lors du passage de la particule d'une cellule de la grille à une autre.

4.2.2 Modélisation par primitives de vent

Après avoir évalué les avantages et inconvénients des méthodes basées sur les équations de Navier-Stokes et de la méthode treillis Boltzmann, nous avons fait le choix d'adopter une méthode basée sur une modélisation procédurale de primitives de vent, similaire à celle introduite dans [WH91].

Le modèle de [WH91]

Cette technique vise à combiner des primitives simples pour façonner des flux complexes, et effectue la simulation pour un coût en temps de calcul nettement inférieur à celles basées sur les équations équations de Navier-Stokes, ou sur la méthode treillis Boltzmann.

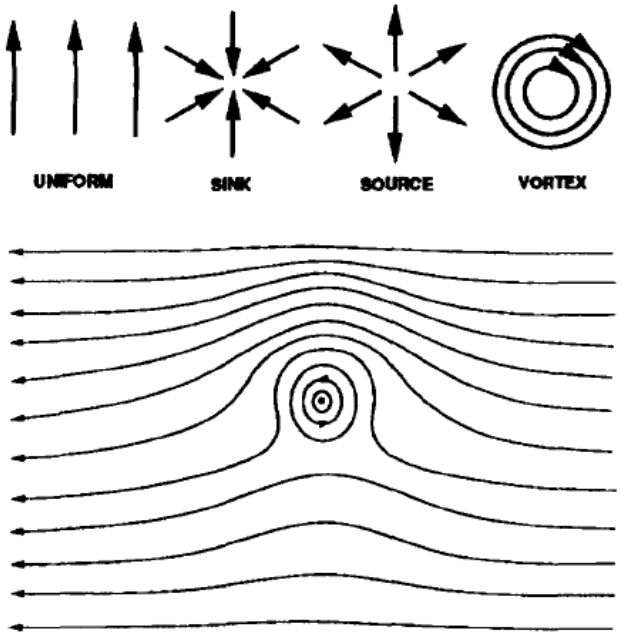

Fig. 4.6 – **Haut** : illustrations des primitives disponibles dans [WH91]. **Bas** : combinaison d'un vent uniforme et d'un vortex.

[WH91] introduit les primitives suivantes (illustrées sur la partie haute de la figure 4.6) :

– Source : Le vent souffle dans toutes les directions à partir d'un axe de l'espace. Cette primitive est décrite en coordonnées cylindriques par les équations :

$$V_r = \frac{a}{2\pi r} \quad ; \quad V_\theta = 0 \quad ; \quad V_z = 0$$

où a est la force de la source ($a > 0$).

– Puits (*Sink* dans[WH91]) : Le vent se rassemble vers un axe de l'espace, depuis toutes les directions. Cette primitive est décrite par la même équation que la primitive source, avec $a < 0$.

– Vortex : Le vent tourne autour d'un axe. Cette primitive est décrite en coordonnées cylindriques par les équations :

$$V_r = 0 \quad ; \quad V_\theta = \frac{b}{2\pi r} \quad ; \quad V_z = 0$$

où b est la force du vortex.

– Vent uniforme : Vent homogène soufflant avec la même force et direction en tout point de la scène.

Ces primitives sont définies de manière à être des solutions des équations de Navier-Stokes, et présentent la propriété qu'une combinaison linéaire de ces solutions est également une solution. Le flux modélisé par combinaison de primitives (illustré en partie basse de la figure 4.6), est donc physiquement plausible.

Adaptation du modèle

Nous avons adapté ce modèle à notre situation, afin de mieux refléter le comportement que nous souhaitions voir affecté à l'animation de nos particules.

Nous avons modifié les primitives source et puits, afin qu'elles opèrent en coordonnées sphériques, et non en coordonnées cylindriques. De cette manière, le vent se trouve émis depuis, ou concentré vers, un point de l'espace, et non un axe comme dans le modèle de [WH91].

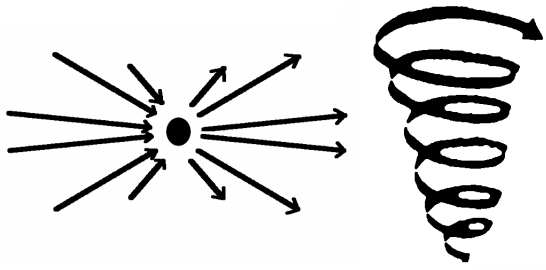

Fig. 4.7 – Les primitives de vent que nous avons ajoutées. **Gauche** : ventilateur. **Droite** : tornade.

Nous avons également ajouté deux nouvelles primitives à ce modèle. La première, désignée sous le nom de « ventilateur » (illustrée sur la partie gauche de la figure 4.7), permet d'obtenir un vent directionnel localisé, pouvant par exemple être émis

par un ventilateur. Cette primitive est modélisée en coordonnées cartésiennes par l'équation :

$$V = \frac{\overrightarrow{D}}{r} * |\overrightarrow{CP_u}.\overrightarrow{D}| * f$$

où $\overrightarrow{D}$ est la direction dans laquelle souffle la primitive, $\overrightarrow{CP_u}$ est le vecteur unitaire (normalisé) allant du centre de la primitive au point où elle est évaluée, r est la distance séparant le centre de la primitive et le point d'évaluation, et f est la force de la primitive.

La deuxième primitive que nous avons ajouté est une « tornade » (illustrée sur la partie droite de la figure 4.7), obtenue très simplement à partir des autres primitives. Nous avons pour cela associé un puits, destiné à rassembler les particules autour d'un point à la base du cône, une source, chargée d'écarter les particules d'un point au sommet du cône afin d'en élargir le rayon, et un vortex permettant d'imprimer un mouvement rotatif aux particules.

Implantation sur la carte graphique

Dans l'optique d'une méthode globale d'animation fonctionnant en temps-réel, ce modèle de vent a été implanté sur la carte graphique. Pour cela, les équations décrivant les primitives ont été transposées en coordonnées cartésiennes, compatibles avec la représentation du vent dans notre texture tridimensionnelle.

Comme présenté dans la section 4.2.1, cette texture est mise à jour par le biais d'un *pixel shader* spécifique. Pour chaque pixel de la texture, i.e. pour chaque cellule de la grille tridimensionnelle représentant le vent soufflant sur la scène, la contribution des primitives présentes dans la scène est évaluée en utilisant les équations dans leur forme cartésienne, puis ces contributions sont additionnées pour calculer le vent soufflant dans la cellule.

Les informations concernant chacune des primitives présentes dans la scène sont transmises au *pixel shader* chargé de la mise à jour de la texture de vent par le biais de matrices, considérées comme de simples tableaux de données. Le *pixel shader* reçoit ainsi en entrée un tableau de matrices, contenant autant d'éléments que de primitives utilisées. Chaque matrice contient les informations indispensables au traitement d'une primitive, comme son type, sa force, son orientation et sa position.

Le *pixel shader*, suivant le type de la primitive, utilise les autres paramètres pour déterminer l'influence de la primitive en chaque cellule, et itère à travers le tableau de matrices pour additionner les contributions de chaque primitive.

Des contraintes liées à la syntaxe de *pixel shader* utilisée, interdisent que le nombre de primitives reste indéfini, la boucle itérant le tableau de matrices devant nécessairement être déroulée. Nous avons donc défini une limite fixée à 10 primitives maximum. Ce nombre peut être étendu jusqu'à 24 sans outrepasser le nombre maximal d'instructions utilisables dans un *pixel shader*. Le chiffre de 10 primitives nous a cependant semblé être un bon compromis, offrant une grande liberté dans la modélisation du vent sans pour autant avoir un impact trop important sur les temps de calcul.

4.2.3 Interface de conception du vent

Afin de permettre à l'utilisateur de manipuler les primitives de vent simplement, nous avons développé une interface de conception (illustrée sur la partie haute de la figure 4.8), permettant de visualiser sous la forme de maillages tridimensionnels l'impact de chaque primitive sur les trajectoires des particules. L'utilisateur peut, grâce à cette interface, combiner diverses primitives pour modéliser un vent complexe.

Dans cette interface, l'utilisateur a la possibilité d'ajouter les primitives qu'il souhaite. Il peut ensuite librement déplacer, incliner, modifier le type, la force et l'orientation de chacune des primitives.

À chaque action de l'utilisateur, l'ensemble de matrices comportant les propriétés des primitives est mis à jour, et transmis au *pixel shader* chargé de la ré-actualisation de la texture de vent. L'utilisateur peut ainsi constater en temps-réel l'impact des modifications qu'il apporte aux primitives sur la dynamique des particules, comme l'illustre la partie basse de la figure 4.8.

4.2.4 Influence du vent sur les particules

La texture de vent peut très simplement influer sur la texture de vélocité. La position de la particule est connue grâce à la texture de position, et permet de

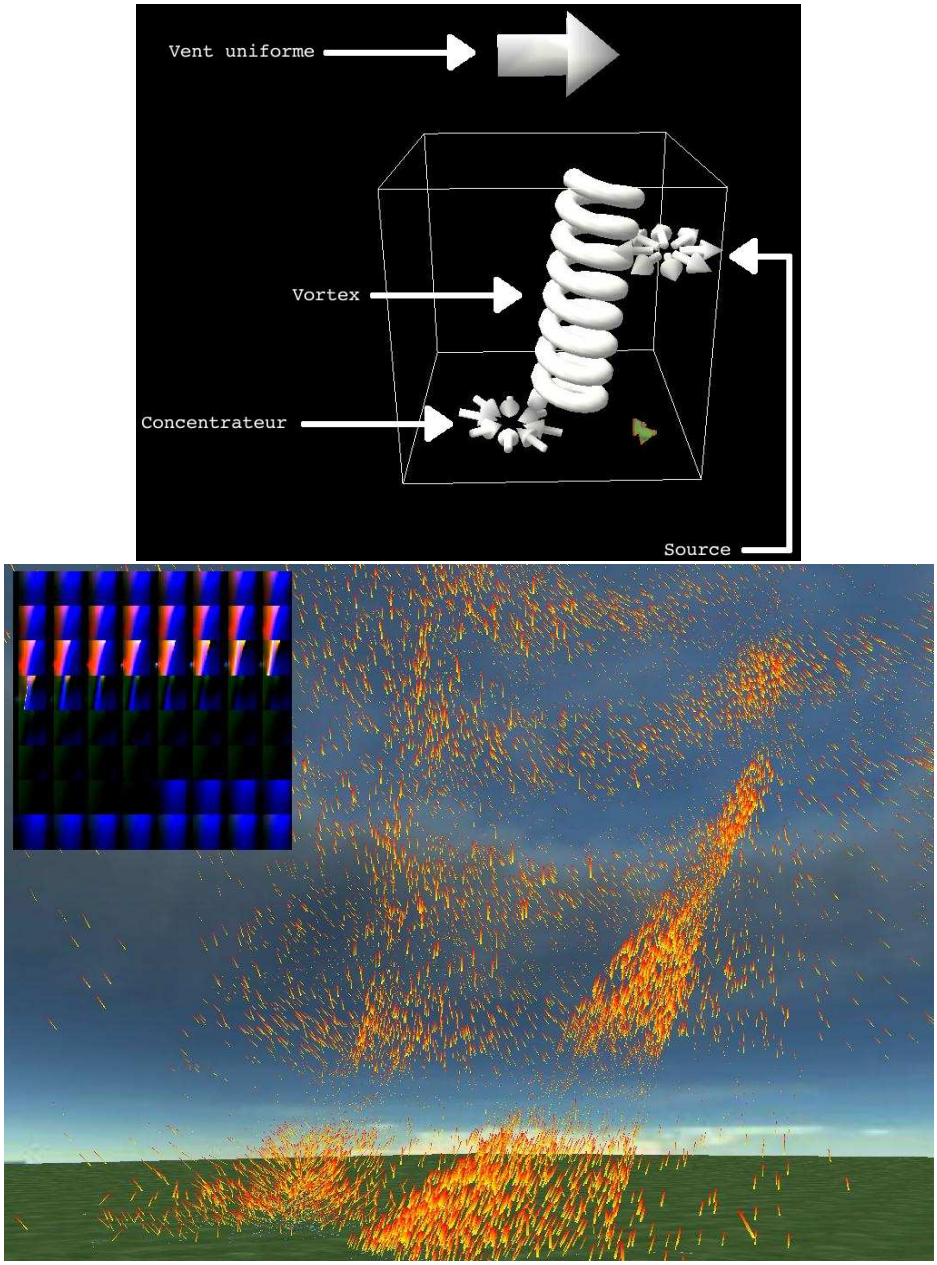

Fig. 4.8 – **Haut** : plusieurs primitives dans l'interface de conception du vent. **Bas** : l'influence du vent sur des particules simples, avec dans le coin haut-gauche la texture de vent générée à partir des primitives.

déterminer la cellule de la texture de vent dans laquelle se trouve la particule. Une interpolation tri-linéaire est réalisée avec les valeurs des cellules voisines de la texture de vent afin d'obtenir une animation plus fluide. La vélocité de la particule est ensuite simplement mise à jour en faisant la moyenne entre la vélocité du pas de temps

précédent, et la direction du vent. De cette manière, la vent influe sur le mouvement des particules, sans empêcher les rebonds.

Le vent a également une influence sur l'orientation des particules, particulièrement dans le cas de cordes. Les particules entraînées dans un flux soufflant latéralement doivent ainsi être inclinées, suivant un angle permettant que la pente de la particule indique la direction dans laquelle elle sera déplacée aux pas de temps suivants. Dans le *vertex shader* chargé de former la géométrie des particules, les coins des imposteurs sont pour cela décalés d'après l'orientation donnée par la texture de vélocité, avant leur conversion vers le repère écran (donc dans le repère monde). Cette inclinaison des particules, observable sur la figure 4.8, permet d'améliorer le réalisme de l'animation d'une image à l'autre.

4.2.5 Discussion

Cette section a introduit les méthodes que nous avons définies pour simuler l'influence du vent sur la scène où évoluent les particules simulées. Notre approche trouve son point de départ dans les travaux présentés dans [WH91], auxquels nous avons apporté quelques nouveautés. L'objectif de proposer une méthode fonctionnant en temps-réel nous a amené à nous placer dans une approche phénoménologique, ce qui nous a contraint à nous éloigner d'une certaine rigueur physique en ce qui concerne les nouvelles primitives de vent que nous avons définies. Elles nous ont cependant permis d'obtenir des résultats visuellement convaincants, et conformes à la dynamique que nous souhaitions insuffler à nos particules.

Ces travaux ont été transposés vers une implantation sur la carte graphique, utilisant un *pixel shader* pour créer un flux de vent complexe stocké dans une texture conceptuellement tridimensionnelle. Nous avons également proposé une interface simple et intuitive permettant à l'utilisateur de modéliser un vent par combinaison de primitives, en ayant un retour visuel immédiat, sur les trajectoires des particules évoluant dans le système.

4.3 Gestion des collisions

Pour améliorer encore le réalisme de notre animation particulaire, nous avons également introduit une détection simplifiée des collisions à notre approche.

4.3.1 Texture de collision

Nous nous intéressons ici à un cas très spécifique de collisions, où les trajectoires suivies par les particules sont très proches les unes des autres, avec une composante verticale fortement prédominante.

Ces considérations nous ont amené à l'idée que les collisions inter-particulaires peuvent être négligées sans risques d'artefacts visuels, de même que les collisions entre les particules et les surfaces verticales, telles que des murs : les particules seront dans ce cas considérées comme ayant été absorbées par la surface, et recréées à une autre position (de la même manière que si elles avaient quitté la boîte à gouttes).

L'approche que nous utilisons peut être considérée comme une extension de celle développée dans [TC00]. Un caméra orthographique est placée à la verticale du milieu de la boîte, de manière à observer tout le contenu de la boîte en vue de dessus. Les plans de clipping de cette caméra sont ajustés avec les bords de la boîte à gouttes. Cette caméra rend chacun des objets présents dans son champ de vision vers une « texture de collision » (à l'exception du système de particules, puisque nous avons fait le choix de ne pas prendre en compte les collisions inter-particules).

Pour cela, un *pixel shader* particulier est appliqué à chacun des objets de la scène. Ce *pixel shader* se substitue au rendu normal des objets, et permet d'enregistrer la normale en chaque pixel de l'objet sur les canaux rouges, vert et bleu de la texture, ainsi que l'altitude au point considéré sur le canal alpha, comme l'illustre la figure 4.9.

Cette texture est en point flottant, l'altitude n'a donc pas à être bornée dans l'intervalle [0-1], et peut être stockée en coordonnées monde pour simplifier les calculs. La composante alpha peut ainsi être utilisée comme une carte de hauteur dynamique, tandis que les canaux RVB permettront de déterminer la direction d'un éventuel rebond si une collision était détectée.

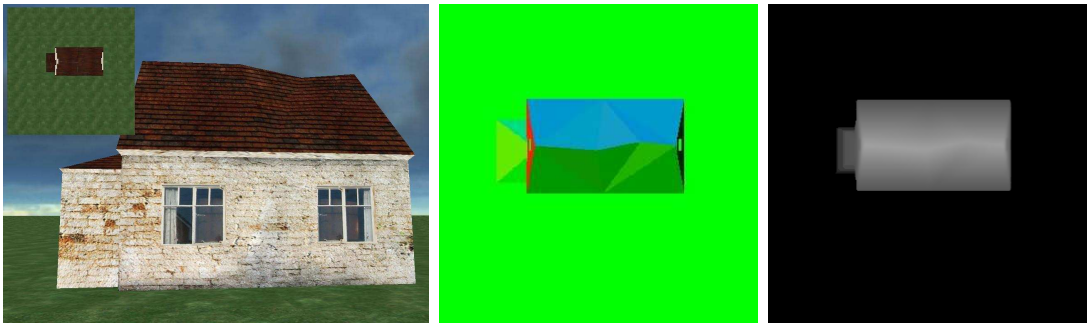

Fig. 4.9 – La texture de collision. **Gauche** : le modèle observé, et dans le coin haut-gauche l'angle sous lequel il est vu de la texture de collision. **Milieu** : normales encodées comme couleurs RVB. **Droite** : altitude encodée sur le canal alpha (ramenée ici à l'espace [0-1] affichable).

Cette texture est re-générée avant chaque mise à jour de la texture de position. Ceci permet de s'adapter aux déplacements de l'observateur, par lesquels de nouveaux objets ou reliefs pourraient être amenés à pénétrer la boîte ; ce procédé autorise également l'utilisation d'objets animés, tels que des véhicules traversant la scène et pouvant être impactés par les gouttes. Dans le cas où la scène n'évolue pas d'une image à l'autre, cette mise à jour n'est pas gênante pour autant ; le temps de rendu de la texture de collision est très court, et son impact sur les performances de l'application est inférieur à 1%.

La résolution requise pour cette texture de collision dépend évidemment des dimensions de la boîte à gouttes. Des expérimentations ont montré qu'une résolution de 128x128 pour cette texture de collision offrait des résultats satisfaisants pour les boîtes typiques (100m sur chaque axe), sans impliquer d'impact notable sur les performances du système.

4.3.2 Interactions

Une fois la texture de collision rendue, elle est utilisée par le *pixel shader* chargé de calculer les nouvelles positions des particules. La position calculée est comparée avec l'altitude du plus haut point situé aux mêmes coordonnées horizontales, donnée par la texture de collision. Lorsque la particule se situe en dessous, une collision est

détectée, et la normale à l'objet est utilisée pour déterminer la direction de rebond de la particule, qui est enregistrée dans la texture de vélocité. Une nouvelle position est alors calculée pour la particule, tenant compte de la vélocité mise à jour. Ce processus est illustré par la figure 4.10.

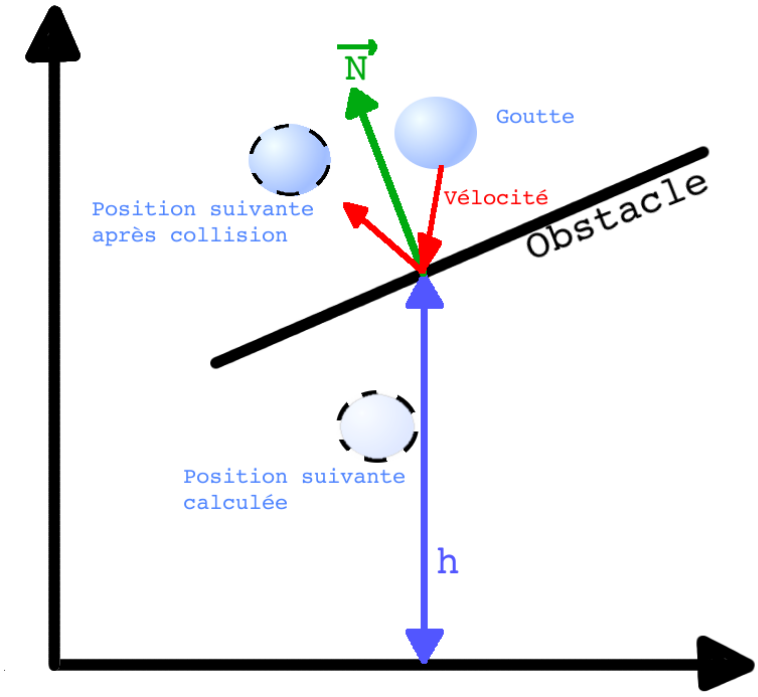

Fig. 4.10 – La vélocité de la goutte l'entraîne sous l'obstacle ; une collision est détectée. Une nouvelle vélocité et une nouvelle position sont calculées. La hauteur h et la normale $\overrightarrow{N}$ de l'obstacle sont obtenues depuis la texture de collision.

La texture de collision affecte également le processus de mise à jour de la texture de vent. Les cellules de la grille tridimensionnelle de vent qui sont déterminées comme étant situées à l'intérieur d'un objet (en comparant leurs coordonnées avec la texture de collision) ne sont pas affectées par les primitives de vent, et l'on considère qu'aucun vent n'y souffle. Du fait que la texture de vent subit une interpolation tri-linéaire dans le *pixel shader* de mise à jour de la texture de vélocité, cette précaution implique que le vent est nul à la surface des objets, puis augmente graduellement dès lors que l'on s'éloigne de cette surface vers l'extérieur de l'objet. Ce comportement nous semble en adéquation avec celui que l'on peut observer dans la nature.

4.3.3 Discussion

Dans cette dernière partie, nous avons complété notre méthode de simulation de la dynamique des particules par une gestion des collisions.

Nous avons pour cela utilisé quelques assertions simplificatrices, permettant au traitement des collisions de n'avoir qu'un impact infime sur les performances du système. En particulier, nous avons considéré que les collisions entre les particules pouvaient être négligées, et que seul le plus haut point situé à une latitude et une longitude données pouvait être impacté, abritant ainsi tous ceux situés en dessous de lui.

La validité de cette deuxième assertion dépend fortement du type de scène utilisé. Nous avons conçu notre système avec l'idée de l'appliquer à des espaces urbains, composés principalement de bâtiments et de véhicules. Dans ce type de cas, notre technique offre d'excellents résultats, la géométrie des objets s'y prêtant remarquablement. Notre méthode est en revanche moins adaptée dans le cas d'arbres, par vent fort : la projection du feuillage dans le plan horizontal sera alors utilisée, et les gouttes ne pourront pas évoluer sous le feuillage sans être détectées comme étant entrées en collision avec l'objet. On notera toutefois qu'en l'absence de vent, les inconvénients de la méthode disparaissent.

4.4 Conclusion

L'objectif de ce chapitre était d'introduire le modèle complet d'animation de particules que nous avons développé pour ajouter un aspect dynamique à la simulation de pluie mise en œuvre dans cette thèse. Nous avons pour cela repris des travaux proposés en 2004 ([KLRS04], [KSW04]) permettant la gestion de systèmes de particules sur la carte graphique, que nous avons adaptés au cas particulier de l'animation de pluie.

Ce modèle a été complété par une gestion du vent, proposée à l'utilisateur sous la forme d'une interface intuitive permettant de combiner des primitives simples pour modéliser un flux de vent complexe.

Nous avons enfin proposé une méthode simple de traitement des collisions, per-

mettant d'obtenir des résultats satisfaisants en des temps de calcul très restreints. Cette méthode n'est pas adaptée à tout type de scène ; elle est cependant efficace pour les applications envisagées pour notre système, telles que des paysages urbains composés principalement de bâtiments verticaux.

Les petites pluies sont longues, les tempêtes soudaines sont courtes.

La Vie et la mort du roi Richard II
William Shakespeare

Chapitre 5

Résultats

Ce chapitre présente les résultats que nous avons obtenus à l'aide des méthodes décrites dans les chapitres 3 et 4. Nous présentons dans un premier temps une vue d'ensemble du modèle de pluie défini dans le cadre de cette thèse. Une étude sur les performances du modèle en fonction des extensions activées et des paramètres choisis, aussi bien en terme de temps de calcul que de qualité visuelle, est présentée dans un second temps.

5.1 Vue d'ensemble

La figure 5.1 propose une vue d'ensemble du système de simulation de pluie avec toutes les extensions que nous avons présentés dans les chapitres 3 et 4.

La première partie de cette figure illustre le déroulement de la simulation de la dynamique des particules (présentée dans le chapitre 4) :

- (1) Permutation entre les deux textures de vélocité et de position (approche ping-pong).
- (2) Mise à jour de la texture de collision.
- (3) Mise à jour de la texture de vent, en utilisant les informations tirées de la texture de collision.
- (4) Calcul de la vélocité, en prenant en compte la vélocité lors du pas de temps précédent, et la direction du vent.
- (5) Calcul de la position, en utilisant la position précédente, la vélocité cou-

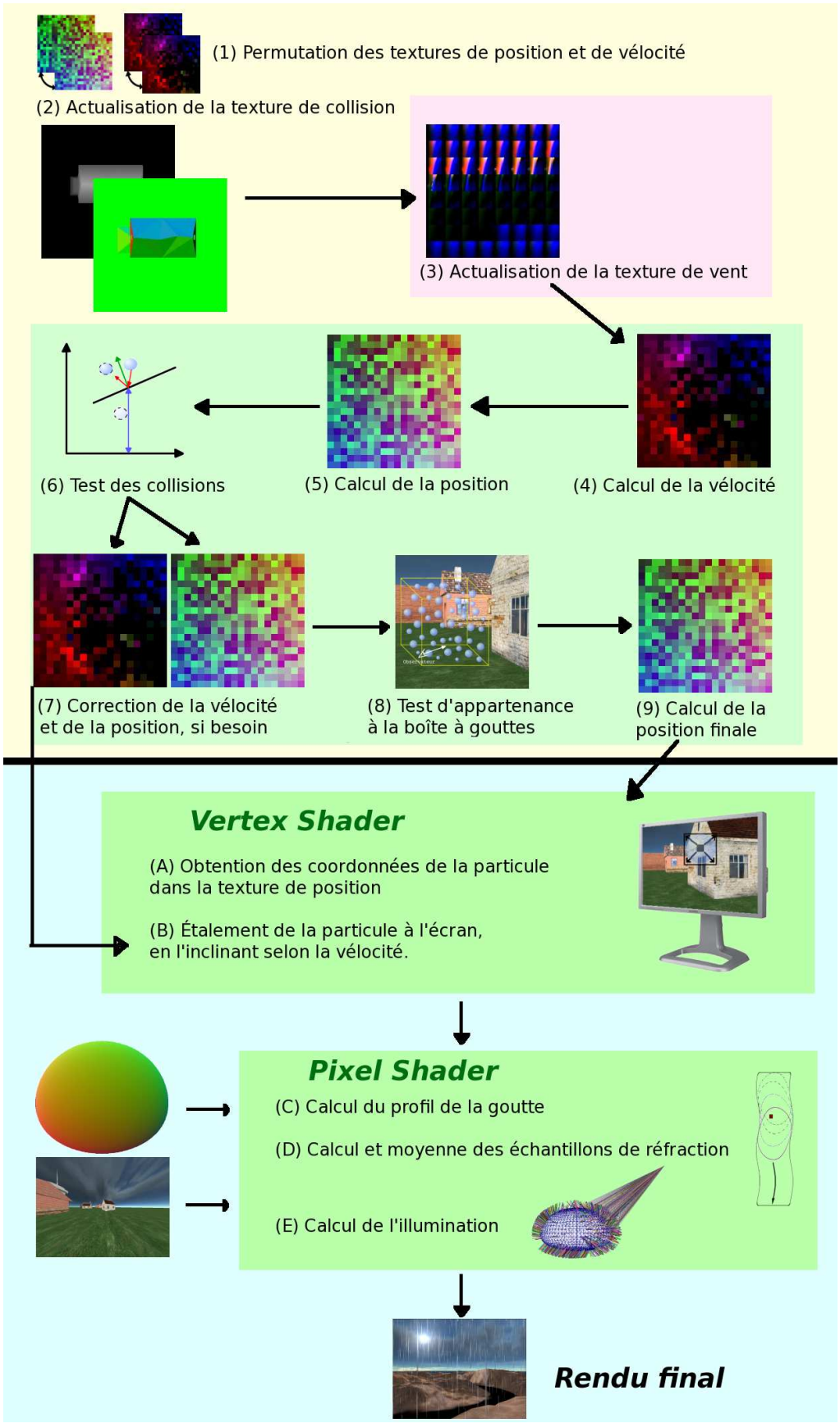

Fig. 5.1 – Vue d'ensemble du système proposé

rante, et le temps écoulé depuis le pas de temps précédent.

- (6) Comparaison de la position avec l'altitude du plus haut objet, tirée de la texture de collision.
  - Si la particule est au dessus de la surface, la position est conservée.
  - Si elle est sous la surface, mais à une distance inférieure à un seuil, une collision se produit. Une direction de rebond est calculée d'après la normale à la surface et la vélocité présupposée.

    (7) Cette direction est utilisée pour rectifier la vélocité, et la position de la particule.
  - Si la particule est sous la surface de l'objet, à une distance supérieure au seuil, la particule est considérée comme ayant été absorbée, par exemple par un mur.

    (7) La particule est détruite et recréée au sommet de la boîte.
- (8) Vérification que la particule se trouve toujours dans les limites imposées par les dimensions de la boîte. Si ce n'est pas le cas, la particule est déplacée du côté opposé de la boîte par rapport à celui par lequel elle vient de sortir.
- (9) Écriture de la nouvelle vélocité et de la nouvelle position dans les textures.

La seconde partie de la figure 5.1 illustre les travaux relatifs au rendu des gouttes (chapitre 3) :

- (A) Dans le *vertex shader*, les positions des particules sont obtenues depuis la texture de position.

  (B) Les imposteurs sont ensuite étalés sur l'écran, en tenant compte de la texture de vélocité (pour incliner les particules suivant leur direction d'évolution)
- (C) Dans le *pixel shader*, le profil de la goutte est calculé.

  (D) La couleur de chaque pixel à l'intérieur du profil est obtenue en calculant la moyenne des couleurs issues des réfractions pour plusieurs échantillons de gouttes contribuant à l'apparence du pixel ; chaque contribution est calculée par le biais du modèle de réfraction, en utilisant le masque correspondant à la taille souhaitée, et la texture d'environnement.

  (E) Enfin, l'influence des sources lumineuses présentes dans la scène est calculée pour modifier l'apparence du pixel.

5.2 Performances

Au cours de nos travaux, nous avons pu constater qu'en utilisant une boîte à gouttes de 100m de côté, 4000 gouttes au minimum étaient nécessaires pour obtenir une impression crédible de pluie. Selon l'importance de la pluie souhaitée, un nombre compris entre 10000 et 20000 gouttes offre des résultats visuellement réalistes.

	Rendu simple	Réfraction	Cordes	Réfraction + Lumière	Cordes + Lumière
0 particules	950 fps				
5000 particules	95 fps	89 fps	72 fps	88 fps	62 fps
10000 particules	83 fps	75 fps	56 fps	74 fps	46 fps
20000 particules	68 fps	59 fps	40 fps	59 fps	29 fps
40000 particules	49 fps	42 fps	24 fps	41 fps	17 fps

Tab. 5.1 – Vitesses d'exécution, suivant le nombre de particules et les extensions activées (résolution : 1024x768, rayon des gouttes : 1 mm)

La vitesse d'exécution atteinte, suivant le nombre de particules et les extensions de rendu activées, est présentée dans le tableau 5.1. Les résultats sont ici comparés à la vitesse d'affichage de la scène en l'absence de pluie ; aucun calcul d'animation ou de rendu n'est alors effectué. Ils sont également comparés aux performances obtenues en utilisant un rendu très simplifié (simple texture appliquée sur les particules), permettant de déterminer l'impact de la gestion du système de particules sur les performances. Les résultats utilisant la méthode de simulation de la persistance rétinienne sont mesurés en utilisant 10 échantillons par pixel. La plate-forme de test sur laquelle ont été mesurés ces résultats est un ordinateur fonctionnant sous Linux, doté d'un processeur AMD cadencé à 2GHz et d'une carte graphique Nvidia Geforce 6800 GT.

Lorsque l'extension de gestion de la persistance rétinienne est activée, le nombre d'échantillons par pixel, ainsi que la taille des gouttes (donc le nombre de pixels traités), ont un impact important sur les performances du système. Le tableau 5.2 permet de constater cet impact, dans le cas d'une animation portant sur 10000

Rayon	1 échantillon	5 éch.	10 éch.	20 éch.	50 éch.
0.5 mm	75 fps	72 fps	65 fps	58 fps	46 fps
1 mm	73 fps	66 fps	56 fps	48 fps	33 fps
2 mm	70 fps	59 fps	47 fps	37 fps	24 fps
5 mm	63 fps	47 fps	35 fps	25 fps	15 fps

Tab. 5.2 – Vitesses d'exécution avec gestion de la persistance rétinienne, suivant le nombre d'échantillons par pixel des cordes, et leur rayon (résolution : 1024x768)

cordes. Les résultats sont visuellement satisfaisants à partir de 5 échantillons par pixel.

5.3 Rendu des gouttes

La figure 5.2 illustre la réfraction observable par notre modèle, dans une scène typique, puis sur des gouttes démesurément grandes afin de mettre en évidence l'effet obtenu. Selon leur position dans l'image, les gouttes ne réfractent pas les mêmes portions de la scène.

La figure 5.3 propose une comparaison entre les résultats que nous avons obtenus en utilisant notre extension de gestion de la persistance rétinienne, et une image tirée du jeu *Unreal Tournament 2004*, développé par la société *Epic Games*, illustrant le type de rendu de pluie généralement observable dans les jeux vidéos récents. On peut observer que la pluie dans le jeu apparaît trop claire, là où notre méthode tient compte de l'illumination de la scène.

La figure 5.4 permet de constater l'apport de notre extension de traitement de l'illumination des gouttes par des sources lumineuses. L'image du haut présente les résultats obtenus sans l'extension, et l'image du bas avec l'extension.

La figure 5.5 illustre la compatibilité des deux extensions que nous avons apportées à notre méthode de rendu des gouttes. Deux images utilisant ces deux extensions sont présentées, l'une illustrant une forte pluie, avec des gouttes larges et longues, l'autre une pluie plus fine en présence de vent.

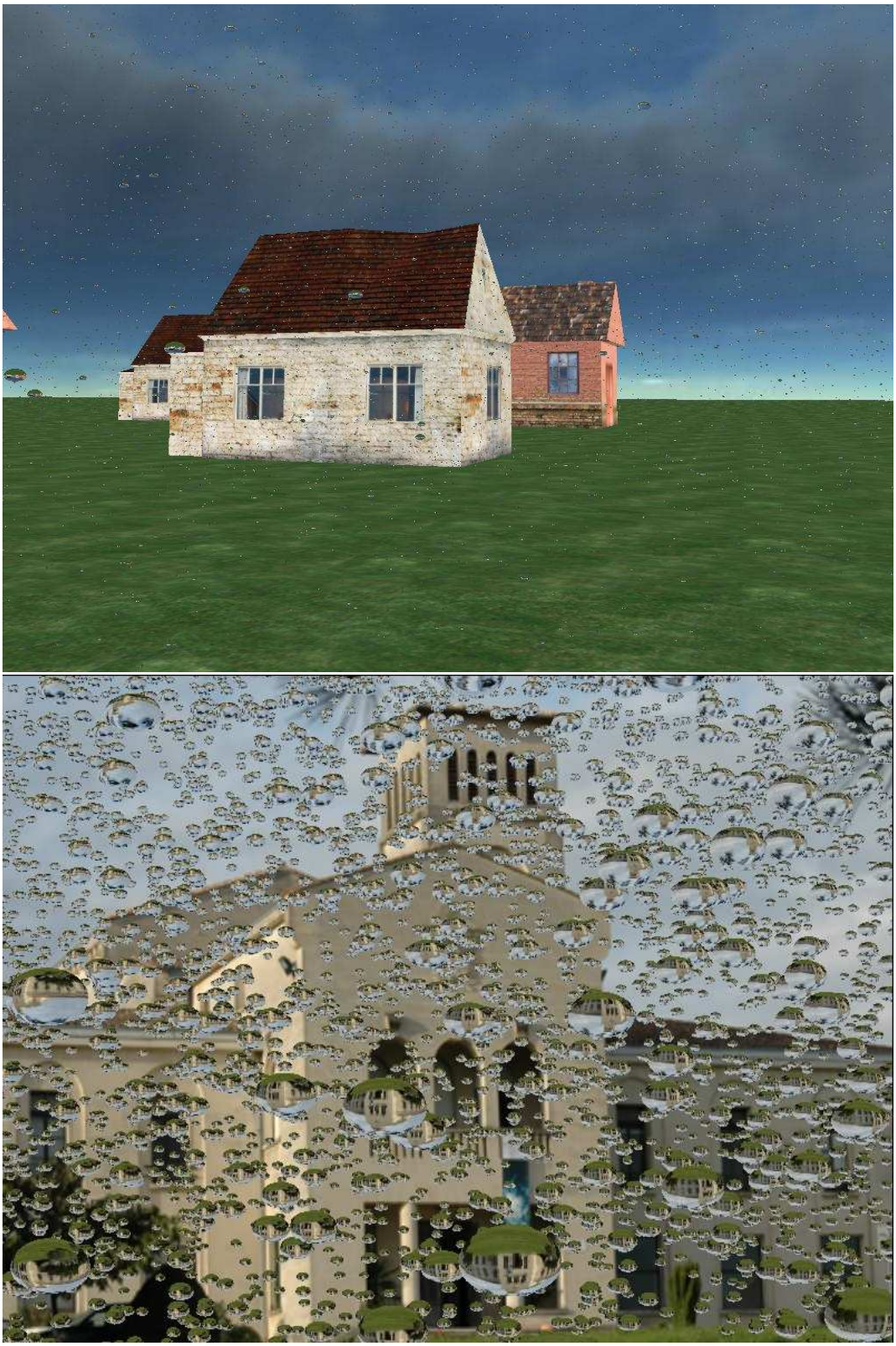

Fig. 5.2 – **Haut** : Une scène typique avec le modèle de réfraction. **Bas** : réfraction au travers de très grosses gouttes, devant une photographie.

Fig. 5.3 – **Haut** : une image obtenue par notre méthode. **Bas** : une image tirée du jeu *Unreal Tournament 2004*.

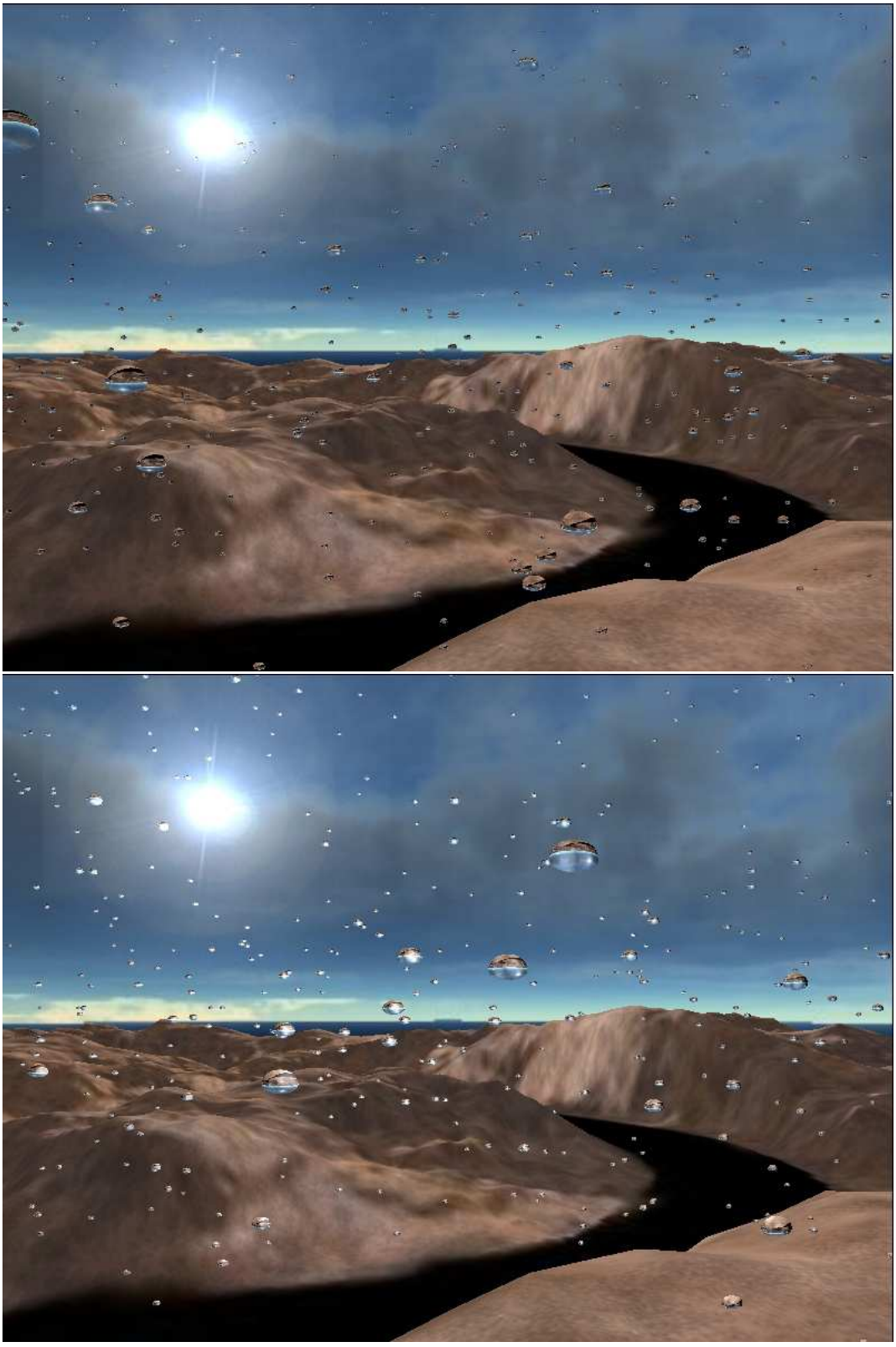

Fig. 5.4 – Extension de gestion de l'illumination des gouttes. **Haut** : sans l'extension. **Bas** : avec l'extension

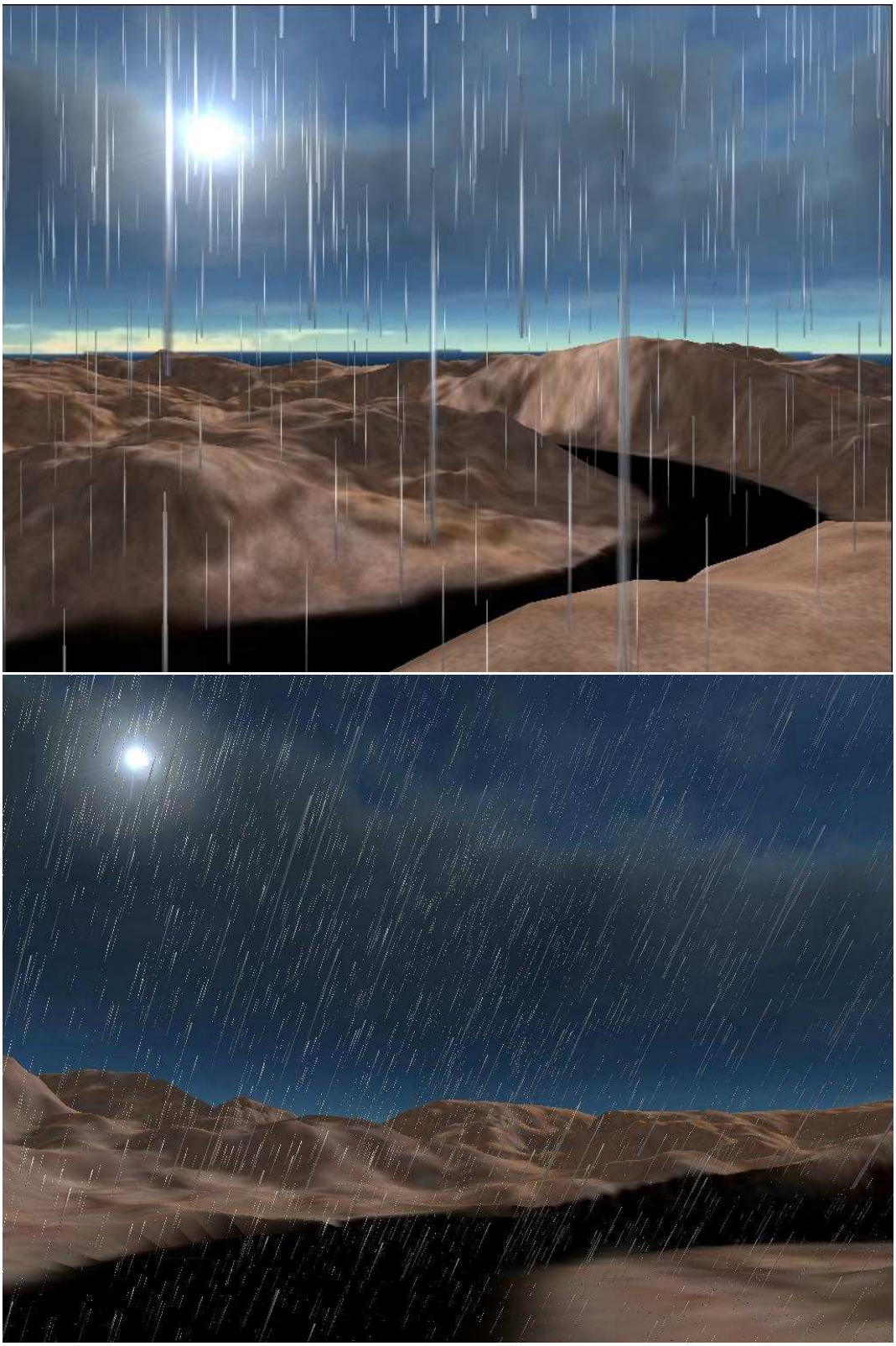

Fig. 5.5 – Résultats obtenus en utilisant simultanément l'extension de gestion de la persistance rétinienne et l'extension de gestion de l'illumination des gouttes. **Haut** : cordes larges et longues. **Bas** : cordes fines, en présence de vent.

5.4 Animation des particules

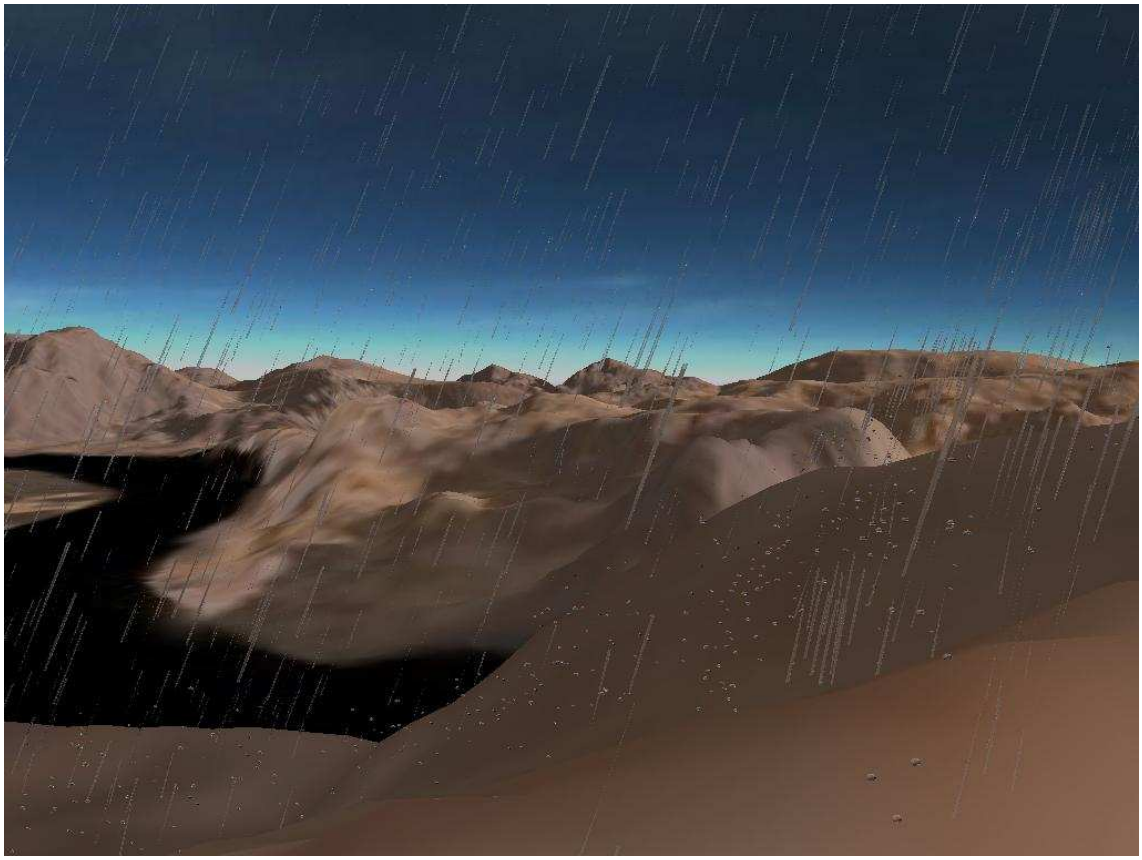

Fig. 5.6 – Les cordes se transforment en gouttes en rebondissant sur le sol.

La figure 5.6 illustre l'action du vent et des collisions sur les particules. Les cordes sont inclinées, du fait du vent uniforme soufflant de la droite de l'image vers la gauche. Lorsqu'elles entrent en collision avec le sol, elles rebondissent sous la forme de gouttes visualisées en utilisant le modèle de réfraction.

La figure 5.7 illustre le changement de trajectoire suivi par les gouttes lorsque l'utilisateur ajoute une primitive de vent au système (ici une tornade). Les gouttes sont ici remplacées par des particules colorées très visible afin de mettre en évidence leurs trajectoires. L'orientation des particules aide beaucoup à percevoir le sens du mouvement.

La figure 5.8 illustre les rebonds de deux particules entrant en collision avec un toit de bâtiment. Les cordes très visibles utilisées ici sont transformées en gouttes

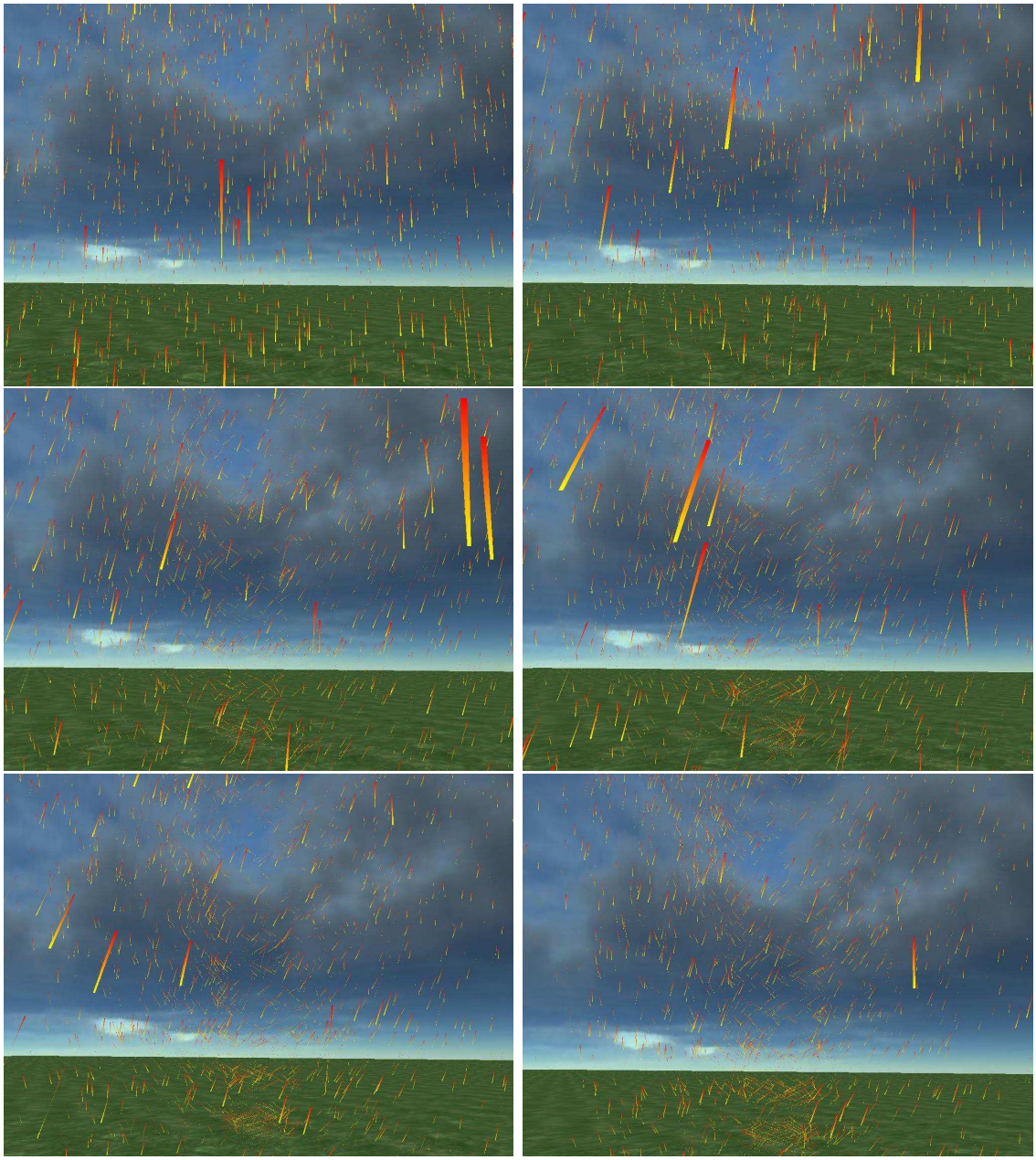

Fig. 5.7 – Six images d'une séquence illustrant la formation d'une tornade par notre méthode d'animation.

rondes dès que la collision est détectée. On peut constater que les collisions sont détectées avec une précision très satisfaisante, en dépit de la résolution relativement faible (128x128) utilisée pour la texture de collision.

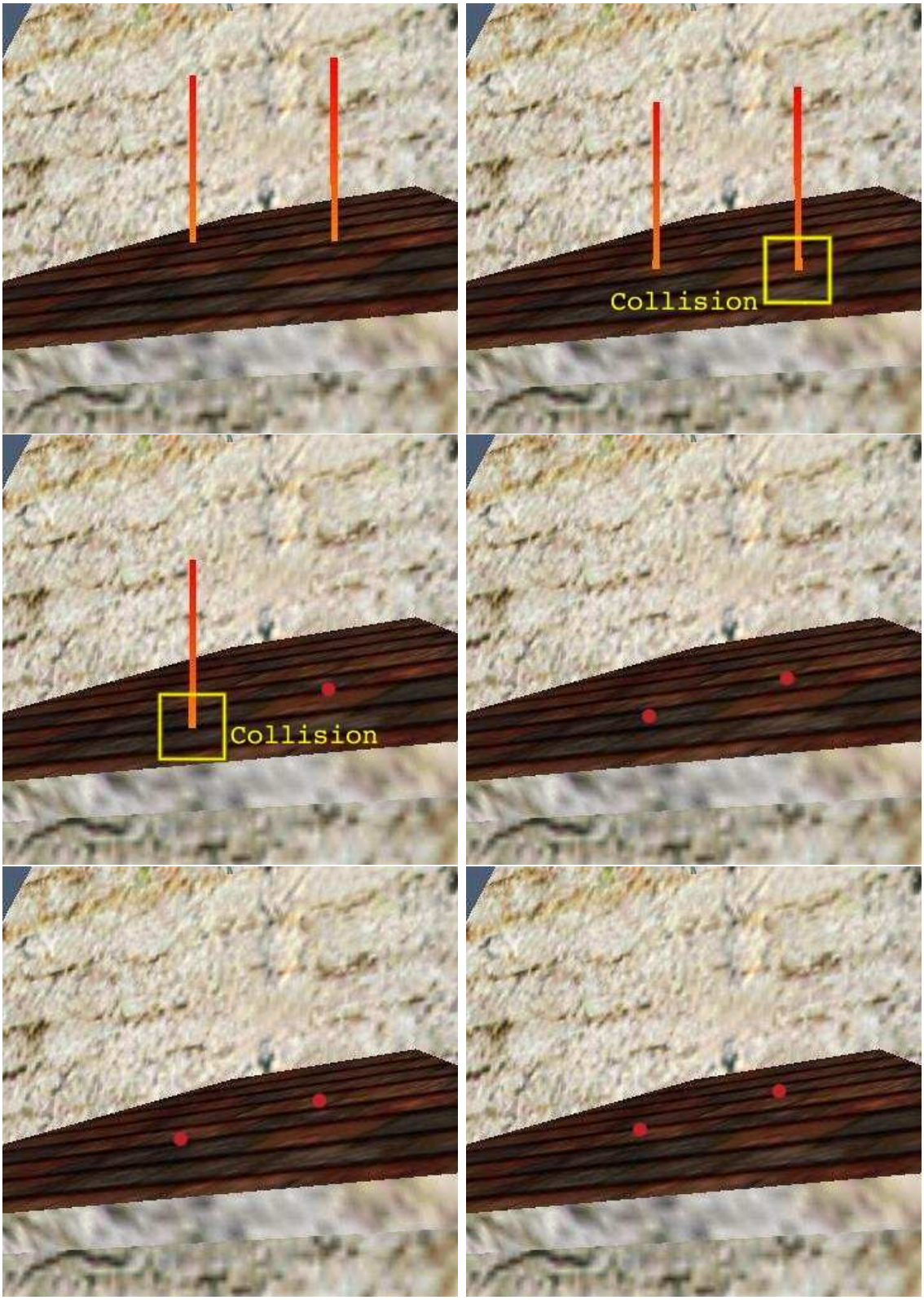

Fig. 5.8 – Six images d'une séquence illustrant la détection des collisions par notre méthode.

La tempête était de retour. Elle avait passé un temps fou à apprendre le métier. Des années à rester cachée dans des vallées loin de tout. Elle avait répété des heures devant un glacier. Étudié les grandes tempêtes d'autrefois. Poli son art jusqu'à la perfection. Et aujourd'hui, ce soir, devant un public visiblement au fait qui l'attendait, la tempête allait faire un. . .oui, un tabac.

Trois sœurcières

Terry Pratchett

Conclusion

La pluie est un phénomène complexe. Déprimante quand elle tombe sur nos vacances, salvatrice quand elle irrigue les cultures, elle est omniprésente dans la vie humaine. Pourtant, très peu de travaux y avaient été consacrés en synthèse d'images avant 2004, en dépit de l'intense activité portant sur la simulation des phénomènes naturels.

Travaux réalisés

Nos travaux ont été partiellement financés par le projet Européen *GameTools*, qui vise à développer des bibliothèques logicielles adaptées aux dernières innovations matérielles du monde du jeu vidéo. Les modèles proposés devaient être utilisables dans le contexte des jeux vidéos et des simulateurs de conduite. L'objectif de nos travaux était de proposer des méthodes permettant de simuler la pluie au cours de sa chute, de manière réaliste et en temps-réel. Nous souhaitions pour cela utiliser les propriétés physiques des gouttes d'eau dans la construction de nos modèles.

Après avoir constaté que la réfraction est très fortement prédominante dans l'apparence d'une goutte d'eau, nous avons cherché à proposer un modèle permettant d'obtenir une bonne approximation de la réfraction en temps-réel. Nous avons pour cela proposé de plaquer sur les gouttes une texture d'environnement dynamique, extraite par une caméra à l'angle de vue très large. Ce plaquage prend en compte la direction dans laquelle sont réfractés les rayons traversant la goutte.

Nous avons ensuite étendu notre modèle afin qu'il gère le phénomène de persistance rétinienne, et offre ainsi des résultats visuellement plus convaincants. Pour cela, nous avons développé un modèle utilisant la contribution de plusieurs posi-

tions successives adoptées par une goutte au cours de sa chute, afin d'obtenir une visualisation réaliste des « cordes » floues que l'on perçoit en observant la pluie tomber.

Dans le cadre des simulateurs de conduite dédiés à la sécurité routière, l'étude de la vision des conducteurs lorsque les conditions de circulation sont très dégradées est un problème très important. Nous avons étudié le comportement de la lumière à l'intérieur d'une goutte d'eau, afin de proposer un modèle empirique offrant une approximation satisfaisante de l'apparence d'une goutte d'eau illuminée par une ou plusieurs sources lumineuses.

Nous nous sommes par la suite intéressés à l'animation de la pluie, problème tout aussi important que son rendu. Nous avons étudié les méthodes d'animation de systèmes de particules gérés par la carte graphique, introduites dans la littérature depuis 2004, et avons adapté ces méthodes au cas pratique de la simulation de pluie. Nous avons ensuite développé un système permettant à l'utilisateur de contrôler un vent soufflant sur la scène, par le biais d'une interface de conception intuitive et efficace. Nous avons enfin développé un système de détection et de réponse aux collisions, utilisant une approche requérant des temps de calcul très réduits, et offrant des résultats satisfaisants pour le type d'applications visé.

Ces travaux ont été valorisés par le biais d'une publication en revue internationale [RJG06a], ainsi qu'une communication en conférence nationale [RJG06b].

Perspectives futures

Dans l'avenir, nous projetons de poursuivre nos travaux par une étude de la perception de nos modèles par des utilisateurs. Ces recherches, en partenariat avec le *Laboratoire Central des Ponts et Chaussées*, auront pour objectif de valider les modèles que nous avons proposés. Pour cela, nos résultats, ainsi que des données réelles, seront présentées à des utilisateurs afin de juger le réalisme de l'altération de la perception liée à la pluie dans notre modèle. Ces méthodes d'évaluation de la perception utilisent des critères objectifs ; on demandera par exemple aux utilisateurs de localiser un objet. Une expérience courante consiste à utiliser un *anneau de*

Landolt, qui s'ouvre périodiquement à une position aléatoire ; les utilisateurs doivent alors localiser l'ouverture. Ces expériences seront menées en utilisant des vidéos filmées par temps de pluie, et des séquences simulées en utilisant les modèles que nous avons proposé.

Nous prévoyons également de prolonger nos travaux par une étude des phénomènes connexes à la simulation de pluie, tels que les écoulements d'eau sur des surfaces ou les éclaboussures. Enfin, nous envisageons de compléter nos travaux par un modèle adapté au rendu des gouttes situées au delà d'une certaine distance. En effet, dans ce cas les gouttes passent sous le seuil du pixel en utilisant notre modèle, et ne sont plus affichées. Un modèle alternatif, utilisé dans un approche multi-résolution, présenterait donc un intérêt. On s'intéressera en particulier aux modèles de milieux participants, permettant la simulation de phénomènes tels que le brouillard.

Bibliographie

[Bat68] K. E. Batcher. Sorting networks and their applications. In *AFIPS Spring Joint Computing Conference*, pages 307–314, 1968.

[BC87] K. V. Beard and C. Chuang. A new model for the equilibrium shape of raindrops. *J. Atmos. Sci.*, 44(11) :1509–1524, 1987.

[BFMF06] R. Bridson, R. Fedkiw, and M. Muller-Fischer. Fluid simulation : Siggraph 2006 course notes. In *SIGGRAPH '06 : ACM SIGGRAPH 2006 Courses*, pages 1–87, New York, NY, USA, 2006. ACM Press.

[Bir03] V. Biri. *Techniques d'animation dans les méthodes globales d'illumination.* PhD thesis, Université de Marne La Vallée, 2003.

[BN04] A. Bouthors and F. Neyret. Modeling clouds shape. In E. Galin M. Alexa, editor, *Eurographics '04 (short papers)*, August 2004.

[BSK+07] K. Bürger, J. Schneider, P. Kondratieva, J. Krüger, and R. Westermann. Interactive visual exploration of instationary 3D-flows. In *Eurographics/IEEE VGTC Symposium on Visualization (EuroVis)*, 2007.

[CB90] C. Chuang and K.V. Beard. A numerical model for the equilibrium shape of electrified raindrops. *J. Atmos. Sci.*, 47(11) :1374–1389, 1990.

[CDE94] S. Chen, G. D. Doolen, and K. G. Eggert. Lattice-Boltzmann fluid dynamics - a versatile tool for multiphase and other complicated flows. *Los Alamos Science*, 22 :99–111, 1994.

[DKY+00] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, and T. Nishita. A simple, efficient method for realistic animation of clouds. In *SIGGRAPH '00 : Proceedings of the 27th annual conference on Computer graphics*

and interactive techniques, pages 19–28. ACM Press/Addison-Wesley Publishing Co., 2000.

[Dum02] E. Dumont. *Caractérisation, modélisation et simulation des effets visuels du brouillard pour l'usager de la route.* PhD thesis, Université Paris V, 2002.

[FHP98] P. Fournier, A. Habibi, and P. Poulin. Simulating the flow of liquid droplets. In *Proceedings of Graphics Interface 98*, pages 133–142, june 1998.

[FR00] A. Frohn and N. Roth. *Dynamics of Droplets.* Springer, 2000.

[GBWQ02] P. G. Gennes, F. Brochard-Wyart, and D. Quéré. *Gouttes, bulles, perles et ondes.* Belin, 2002.

[GKN07] K. Garg, G. Krishnan, and S.K. Nayar. Material Based Splashing of Water Drops. In *Proceedings of Eurographics Symposium on Rendering*, Jun 2007.

[Gla94] A. S. Glassner. *Principles of Digital Image Synthesis.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[GN03] K. Garg and S. K. Nayar. Photometric model of a rain drop. Technical report, Columbia University, 2003.

[GN04] K. Garg and S. K. Nayar. Detection and removal of rain from videos. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), with CD-ROM, 27 June - 2 July 2004, Washington, DC, USA*, volume 1, pages 528–535, 2004.

[GN05] K. Garg and S.K. Nayar. When Does a Camera See Rain ? In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1067–1074, Oct 2005.

[GN06] K. Garg and S.K. Nayar. Photorealistic Rendering of Rain Streaks. *ACM Trans. on Graphics (also Proc. of ACM SIGGRAPH)*, Jul 2006.

[GN07] K. Garg and S.K. Nayar. Vision and Rain. *International Journal on Computer Vision*, pages 1–25, Feb 2007.

[GPR+03] S. Guerraz, F. Perbet, D. Raulo, F. Faure, and M. P. Cani. A procedural approach to animate interactive natural sceneries. In *Computer Animation and Social Agents (CASA)*, 2003.

[Gre75] A. W. Green. An approximation for the shapes of large raindrops. *J. Appl. Meteor.*, 21 :1578–1583, 1975.

[Har03] M. J. Harris. *Real-Time Cloud Simulation and Rendering.* PhD thesis, University of North Carolina, 2003.

[Har04] M. J. Harris. Fast fluid dynamics simulation on the gpu. In *GPU GEMS*, pages 637–665, Mar. 2004.

[HBSL03] M. J. Harris, W. V. Baxter, T. Scheuermann, and A. Lastra. Simulation of cloud dynamics on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics hardware*, pages 92–101. Eurographics Association, 2003.

[JH02] M. Jonnson and A. Hast. Animation of water droplet flow on structured surfaces. In *SIGRAD2002, The Annual SIGRAD Conference*, pages 17–22, Nov 2002.

[JTT01] P. Jiménez, F. Thomas, and C. Torras. 3D Collision Detection : A Survey. *Computers and Graphics*, 25(2) :269–285, apr 2001.

[KIY99] K. Kaneda, S. Ikeda, and H. Yamashita. Animation of water droplets moving down a surface. *Journal of Visualization and Computer Animation*, 10(1) :15–26, 1999.

[KJ01] A. Kolb and L. John. Volumetric model repair for virtual reality applications. In *Proceedings of Eurographics 2001, Short presentations*, 2001.

[KKKW05] J. Krüger, P. Kipfer, P. Kondratieva, and R. Westermann. A particle system for interactive visualization of 3D flows. *IEEE Transactions on Visualization and Computer Graphics*, 11(6), 11 2005.

[KKW05] P. Kondratieva, J. Krüger, and R. Westermann. The application of gpu particle tracing to diffusion tensor field visualization. In *Proceedings IEEE Visualization 2005*, 2005.

[KKY93] K. Kaneda, T. Kagawa, and H. Yamashita. Animation of water droplets on a glass plate. In *Computer Animation*, pages 177–189, 1993.

[KLRS04] A. Kolb, L. Latta, and C. Rezk-Salama. Hardware-based simulation and collision detection for large particle systems. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics hardware*, pages 123–131, New York, NY, USA, 2004. ACM Press.

[KSW04] P. Kipfer, M. Segal, and R. Westermann. Uberflow : A GPU-based particle engine. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics hardware*, pages 115–122, 2004.

[KZYN96] K. Kaneda, Y. Zuyama, H. Yamashita, and T. Nishita. Animation of water droplet flow on curved surfaces. In *Pacific Graphics '96*, pages 50–65, 1996.

[LFWK05] W. Li, Z. Fan, X. Wei, and A. Kaufman. Flow simulation with complex boundaries. In *GPU GEMS 2*, pages 747–764, 2005.

[LG98] M. C. Lin and S. Gottschalk. Collision detection between geometric models : a survey. In *IMA Conference on Mathematics of Surfaces*, pages 37–56, 1998.

[LWK03] W. Li, X. Wei, and A. E. Kaufman. Implementing lattice boltzmann computation on graphics hardware. *The Visual Computer*, 19(7-8) :444–456, 2003.

[LZK+04] M. S. Langer, L. Zhang, A. W. Klein, A. Bhatia, J. Pereira, and D. Rekhi. A spectral-particle hybrid method for rendering falling snow. In *Rendering Techniques 2004 (Eurographics Symposium on Rendering)*. ACM Press, june 2004.

[MOK95] K. Myszkowski, O. G. Okunev, and T. L. Kunii. Fast collision detection between complex solids using rasterizing graphics hardware. *The Visual Computer*, 11(9) :497–511, 1995.

[MS95] P. W. C. Maciel and P. Shirley. Visual navigation of large environments using textured clusters. In *Symposium on Interactive 3D Graphics*, pages 95–102, 211, 1995.

[Ney00] F. Neyret. A phenomenological shader for the rendering of cumulus clouds. Technical Report RR-3947, INRIA, May 2000.

[NIDN97] T. Nishita, H. Iwasaki, Y. Dobashi, and E. Nakamae. A modeling and rendering method for snow by using metaballs. In *EG'97 Computer Graphics Forum*, volume 16, number 3, pages 357–364, 1997.

[OLG+05] . D. Owens, D. Luebke, N. Govindaraju, M. J. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. In *Eurographics 2005, State of the Art Reports*, pages 21–51, aug 2005.

[OLG+07] . D. Owens, D. Luebke, N. Govindaraju, M. J. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1) :80–113, 2007.

[PC01] F. Perbet and M. P. Cani. Animating prairies in real-time. In *ACM-SIGGRAPH Symposium on Interactive 3D Graphics (I3D)*, USA, Mar 2001.

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

[RJG06a] P. Rousseau, V. Jolivet, and D. Ghazanfarpour. Realistic real-time rain rendering. *Computers & Graphics*, 30(4) :507–51, 2006. Special issue on Natural Phenomena Simulation.

[RJG06b] P. Rousseau, V. Jolivet, and D. Ghazanfarpour. Rendu réaliste de pluie en temps-réel. In *Actes des 19èmes Journées AFIG*, pages 105–112, nov 2006.

[Ros00] O. N. Ross. Optical remote sensing of rainfall micro-structures. Master's thesis, Freie Universität Berlin, 2000. In partnership with University of Auckland.

[SDY02] T. Sato, Y. Dobashi, and T. Yamamoto. A method for real-time rendering of water droplets taking into account interactive depth of field effects. In *Entertainment Computing : Technologies and Applications, IFIP First International Workshop on Entertainment Computing (IWEC 2002)*, volume 240 of *IFIP Conference Proceedings*, pages 125–132. Kluwer, 2002.

[Skj07] J. Skjermo. Gpu-based wind animation of trees. In *Theory and Practice of Computer Graphics 2007*, pages 29–36, June 2007.

[SW03] S. Starik and M. Werman. Simulation of rain in videos. In *3rd international workshop on texture analysis and synthesis (Texture03)*, pages 95–100, Nice, France, 2003.

[Tar07] S. Tariq. Rain. Technical report, Nvidia, 2007.

[Tat06a] N. Tatarchuk. Artist-directable real-time rain rendering in city environments. In *Proceedings of Game Developers Conference*, 2006.

[Tat06b] N. Tatarchuk. Artist-directable real-time rain rendering in city environments. In *SI3D '06 : Proceedings of the 2006 symposium on Interactive 3D graphics and games Poster*, page 30, New York, NY, USA, 2006. ACM Press.

[TC00] F. Tecchia and Y. Chrysanthou. Real-time visualisation of densely populated urban environments : a simple and fast algorithm for collision detection. In *Eurographics UK*, Swansee, April 2000.

[TI06] N. Tatarchuk and J. Isidoro. Artist-directable real-time rain rendering in city environments. In *Eurographics Workshop on Natural Phenomena*, New York, NY, USA, 2006. ACM Press.

[WH91] J. Wejchert and D. Haumann. Animation aerodynamics. In *SIGGRAPH '91 : Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 19–22, New York, NY, USA, 1991. ACM Press.

[WLMK04] X. Wei, W. Li, K. Mueller, and A. E. Kaufman. The lattice-boltzmann method for simulating gaseous phenomena. *IEEE Transactions on Visualization and Computer Graphics*, 10(2) :164–176, 2004.

[Wlo02] M. Wloka. Fresnel reflection. Technical Report TR-00534-001_v01, Nvidia, june 2002.

[WMT05] H. Wang, P. J. Mucha, and G. Turk. Water drops on surfaces. *ACM Trans. Graph.*, 24(3) :921–929, 2005.

[WUT+04] A. Wilkie, C. Ulbricht, R. F. Tobler, G. Zottiand, and W. Purgathofer. An analytical model for skylight polarisation. In *Rendering Techniques 2004 (Eurographics Symposium on Rendering)*. ACM Press, june 2004.

[WW04] N. Wang and B. Wade. Rendering falling rain and snow. In *ACM SIGGRAPH 2004 Technical Sketches Program*, 2004.

[WZF+03] X. Wei, Y. Zhao, Z. Fan, W. Li, S. Yoakum-Stover, and A. E. Kaufman. Blowing in the wind. In *SCA '03 : Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 75–85, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[WZF+04] X. Wei, Y. Zhao, Z. Fan, W. Li, F. Qiu, and S. Yoakum-Stover. Lattice-based flow field modeling. *IEEE Transactions on Visualization and Computer Graphics*, 10(6) :719–729, 2004. Fellow-Arie E. Kaufman.

[YJC98] Y.J. Yu, H.Y. Jung, and H.G. Cho. A new rendering technique for water droplet using metaball in the gravitation force. In *6th International Conference in Central Europe on Computer Graphics and Visualization (WSCG '98)*, pages 448–454, 1998.

[YJC99] Y.J. Yu, H.Y. Jung, and H.G. Cho. A new water droplet model using metaball in the gravitational field. *Computers and Graphics*, 23(2) :213–222, apr 1999.

[YZZ04] Y. Yang, C. Zhu, and H. Zhang. Real-time simulation : Water droplets on glass windows. *Computing in Science and Eng.*, 6(4) :69–73, 2004.

[Zdr04] D. Zdrojewska. Real time rendering of heterogeneous fog based on graphics hardware acceleration. In *CESCG 2004*, april 2004.