Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Gráfica

Simulación del desagüe de agua de lluvia en un sistema de techos utilizando PhysX

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
por la Bachiller

Adriana Desirée Urdaneta Medina
para optar al título de Licenciado en Computación

Tutor:
Prof. Ernesto Coto

Caracas, 26 de Abril 2013

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Gráfica

ACTA DEL VEREDICTO

Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación para examinar el Trabajo Especial de Grado, presentado por la Bachiller Adriana Urdaneta, C.I.: 18.233.140, con el título **"Simulación del desagüe de agua de lluvia en un sistema de techos utilizando PhysX"**, a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los miembros del jurado, se fijó el día 26 de Abril de 2013, a las 10:00 am, para que su autor lo defendiera en forma pública, en el Centro de Computación Gráfica de la Escuela de Computación, lo cual esta realizó mediante una exposición oral de su contenido, y luego respondió satisfactoriamente a las preguntas que le fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió **APROBARLO**.

En fe de lo cual se levanta la presente acta, en Caracas el 26 de Abril de 2013, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Ernesto Coto, tutor del trabajo.

Prof. Ernesto Coto
(Tutor)

Prof. Rhadamés Carmona
(Jurado Principal)

Prof. Jaime Blanco
(Jurado Principal)

RESUMEN

Título:
Simulación del desagüe de agua de lluvia en un sistema de techos utilizando PhysX

Autor:
Adriana Desirée Urdaneta Medina

Tutor:
Prof. Ernesto Coto

PhysX SDK es un API de Nvidia para el cálculo de la física que permite simular, entre otras cosas, el comportamiento de un fluido por medio del algoritmo *Smoothed Particle Hydrodynamics* (SPH). En este trabajo se hace uso del API de PhysX para simular el comportamiento del agua de lluvia y su comportamiento físico en un sistema de drenaje para diferentes tipos de techos y canaletas.

El desarrollo de diversos proyectos arquitectónicos puede ser muy variable y existen muchos estándares que se adaptan a ellos en cuanto al control del drenaje de aguas de lluvia. Este trabajo contempló la disponibilidad de una serie de modelos variados para el sistema de canaletas y techos, con el fin de poder construir diferentes estructuras y evaluar el comportamiento del flujo al momento del drenaje, así como determinar la mejor configuración de los elementos que intervienen en el despliegue final para obtener un render eficiente y de calidad.

Palabras Claves: PhysX, partículas, sistema de techos y canaletas, fluido, simulación de lluvia.

AGRADECIMIENTOS

A Dios, a quien debo todo cuanto tengo en la vida, incluida la vida misma. Dios que me ha dado todas mis capacidades y quien me ha dado la fuerza para superar obstáculos y alcanzar cada una de las metas de mi vida.

A mi familia que me apoyó durante toda mi carrera y me orientaron para poder estudiar en esta casa de estudios, la UCV de la cual me lleno de orgullo y le doy gracias a Dios el haberme permitido estudiar aquí. Mis padres, que me han apoyado siempre en todos los aspectos de la vida, y aunque en este tramo final a veces parecían perder la paciencia, siempre me acompañaron y me ayudaron.

A Édgar, que me ha ayudado en todo momento, no solo en lo académico sino también en aquellos momentos en que estaba muy deprimida y sentía que nunca iba a poder terminar, él supo darme ánimos para seguir adelante, para continuar e incluso a veces siento que se tomó la molestia de esperarme…

A mi tutor quien me orientó en el desarrollo de este TEG cuando no sabía ya que hacer, me ayudó a mantener la calma, me tuvo mucha paciencia y me presionó cuando lo necesité.

A todos los profesores de la universidad que tanto me enseñaron, a los profesores, compañeros y amigos del CCG y de la universidad de quienes tanto aprendí y con quienes tanto compartí y sigo compartiendo, con especial cariño agradezco a Kijam, Pedro, Alex y Édgar.

A mis otros amigos, que aún cuando a veces no entendían nada de lo que yo estaba haciendo en la tesis ni porque me tardaba tanto, siempre supieron comprenderme e incluso buscaban la manera de ayudarme.

En definitiva a todos, muchísimas gracias.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Introducción

En un proyecto arquitectónico intervienen una serie de elementos fundamentales que han de ser el sustento del diseño, planificación y construcción de cualquier tipo estructura. Cuando se habla de esto de inmediato se piensa en los cimientos, el techo, los muros, ventanas, acabado, sistemas eléctricos y sanitarios, entre otras muchas cosas que ya cuentan con un conjunto de reglas generales de arquitectura e ingeniería que van conduciendo la ejecución de la obra más allá de la diversidad de estilos y diseños que englobe el proyecto.

El sistema sanitario es una de las piezas claves de la construcción, por el hecho de tratar aspectos tan fundamentales como el acceso al agua en la estructura, así como la conducción de aguas residuales o de lluvia. En este caso, la recolección, conducción y disposición de las aguas de lluvia en un edificio es un problema no menos relevante, para el cual la arquitectura plantea una diversidad de soluciones estrechamente relacionadas al sistema de techos utilizado y la región geográfica que incide directamente en la intensidad de precipitaciones que la estructura debe soportar y por ende a la cantidad de agua de lluvia que recibirá.

En este trabajo se desarrolla una aplicación interactiva para la simulación del flujo del agua de lluvia sobre diversos tipos de techos y su drenaje, utilizando diferentes sistemas de desagües con canaletas y tuberías.

El documento está estructurado en capítulos. El Capítulo I plantea la propuesta del Trabajo Especial de Grado, la cual describe el planteamiento del problema, los objetivos, la metodología y la plataforma que ha de utilizarse durante el desarrollo del trabajo.

El Capítulo II expone brevemente los conceptos teóricos básicos de la Mecánica de Fluidos, se presentan las ecuaciones diferenciales de Navier-Stokes que simulan la dinámica de fluidos y el algoritmo *Smoothed Particle Hydrodynamics* que se basa en estas ecuaciones para simular el comportamiento del fluido.

En el Capítulo III se exponen todos los aspectos referentes a la simulación de la lluvia y de las estructuras con PhysX.

El Capítulo IV muestra el basamento teórico a partir del cual están sustentadas las operaciones arquitectónicas utilizadas en la construcción de las estructuras, especialmente en la conformación del sistema de techos y canaletas.

El Capítulo V expone las consideraciones teóricas de despliegue y en el Capítulo VI se explican todos los detalles de diseño e implementación utilizados en el TEG.

El Capítulo VII muestra todo lo referente a las pruebas de rendimiento cuantitativo y cualitativo a la cual fue sometida la aplicación y el análisis de los resultados obtenidos de las mismas. Finalmente el Capítulo VIII expone las conclusiones y los trabajos futuros planteados en este TEG.

CAPÍTULO I. Planteamiento del TEG

En este capítulo se presenta el planteamiento del TEG, los objetivos, la metodología utilizada, el alcance y las limitaciones de este trabajo.

1. Planteamiento del Problema

En el mundo del diseño arquitectónico y la construcción existen muchas variables que deben ser tomadas en cuenta a la hora de realizar un proyecto. Múltiples factores influyen en el hecho de que un diseño se adecúe correctamente al propósito para el que ha de construirse, e incluso en que el proyecto sea factible o no.

Para la elaboración de un proyecto arquitectónico es necesario diseñar detalladamente el sistema de construcción (cimentación, muros, pisos, etc.), instalaciones eléctricas e instalaciones sanitarias. El sistema de instalaciones sanitarias abarca tanto la dotación de agua a la estructura, como el drenaje de aguas de lluvia y aguas servidas, negras o cloacales.

Al igual que en los otros sistemas que engranan un proyecto arquitectónico, la recolección, conducción y disposición de las aguas de lluvia en un edificio puede ser tratada de diversas maneras. Dependiendo del diseño creado por el arquitecto será necesario aplicar la solución más adecuada.

Por ello, a pesar de que existen estándares y soluciones que se adaptan a cada proyecto arquitectónico, el sistema utilizado para el drenaje de aguas de lluvias estará muy ligado al diseño elegido y a la cantidad de agua de lluvia que ha de recibir el techo de acuerdo a la intensidad de precipitación habitual en dicha región.

Para la elección del sistema de drenaje se calculan todas estas variables y, junto al diseño del arquitecto, es posible determinar así la morfología del techo, el material que ha de utilizarse, las canaletas o tuberías necesarias para desaguar la cantidad máxima de lluvia estimada, su diámetro y longitud, etc.

Todos estos cálculos serán expresados en tablas que permiten justificar la viabilidad del proyecto. Aún cuando el proyecto puede ser modelado en 3D con la ayuda de aplicaciones y herramientas computacionales, sería interesante desde el punto de vista pedagógico simular físicamente las condiciones climáticas a las que puede estar sometido dicho diseño.

Para el TEG se quiere desarrollar una aplicación interactiva que permita simular el flujo del agua de lluvia sobre diversos tipos de techos y su drenaje utilizando diferentes sistemas de desagües con canaletas y tuberías.

2. Objetivo General

Simular el flujo del agua de lluvia y su drenaje sobre diversos sistemas de techo.

3. Objetivos Específicos

- Utilizar el SDK de PhysX para simular el comportamiento físico del agua de lluvia con diferentes intensidades de precipitación.
- Desarrollar una interfaz que permita configurar diferentes tipos de techos y sistemas de drenaje para el agua de lluvia.
- Realizar un estudio del rendimiento de la aplicación desarrollada y de la calidad visual de la simulación.

4. Metodología

- Se utilizará una estrategia de programación orientada a objetos para el desarrollo del simulador.
- Se utilizará el método *Smoothed Particles Hydrodynamics* que provee el API[1] de PhysX[1] para la simulación.
- Con el fin de que la simulación sea lo más realista posible, se utilizarán diversas técnicas de visualización y se ajustarán los parámetros del fluido, para asemejarlo a la lluvia lo más posible.

5. Plataforma de hardware y software

- Tarjeta de video Nvidia® de la serie GeForce 8000 en adelante (con soporte para PhysX)
- Sistema operativo Windows XP o superior
- Microsoft Visual C++ 2010
- QT 4.7[2] para el desarrollo de la interfaz
- OpenGL 2.0 en adelante para el despliegue gráfico

[1] *Application Programming Interface*, en español Interfaz de programación de aplicaciones.

- PhysX SDK 2.8 para el procesamiento de los cálculos relacionados con física

6. Alcances y limitaciones de este trabajo

En urbanismo, el sistema de drenaje de aguas de lluvia en edificaciones no contempla únicamente la disposición de agua que recibe el techo sino también la conducción de esas aguas hasta su destino final, una planta potabilizadora o un río no contaminado. El alcance de este trabajo solo contempla el drenaje de agua de lluvia en una edificación a través de un sistema de canaletas o drenes.

La construcción de edificaciones es muy variante y compleja. Una estructura puede tener una o más estructuras de techo, pueden combinar techos inclinados con techos planos a diferentes alturas, se pueden utilizar diversos materiales con o sin revestimiento y múltiples extremos inclinados y lados que se intersecan en un ángulo inclinado que se proyecta. Para este trabajo solo se construyen estructuras simples de un solo módulo con techos planos, inclinados ó a dos aguas.

El tipo de techo y el material con el que esté hecho o del que esté revestido influye también en el drenaje del agua. Lo mismo ocurre con la permeabilidad y las capas aislantes para protegerse de la humedad, la rugosidad que afecta el roce del agua, entre otras cosas. En este trabajo no son tomados en cuenta los materiales del techo porque no es posible emular con PhysX la porosidad de los materiales y la absorción del agua. La rugosidad es considerada definiendo una media entre varios materiales como coeficiente de roce dinámico.

El área servida a considerar va desde los $36m^2$ hasta $400m^2$ en el caso de los techos planos o inclinados, lisos o corrugados. Cuando el techo inclinado tiene revestimiento de tejas alcanza únicamente hasta los $256m^2$ por la complejidad del mallado. Los techos a dos aguas pueden ser vistos como dos techos inclinados que convergen en un brocal de techo, por lo que pueden alcanzar un área de $800m^2$, desaguados por dos sistemas de canaletas (uno para cada lado).

CAPÍTULO II. Fluidos

En este capítulo se explican brevemente los aspectos más resaltantes de la mecánica de fluidos y las características más importantes de un fluido. Así mismo, se explican brevemente el modelo matemático de Navier-Stokes que permite simular la dinámica de fluidos y el algoritmo *Smoothed Particle Hydrodynamics*, basado en dicho modelo matemático.

1. Conceptos Fundamentales de Mecánica de Fluidos

De acuerdo a [3], un fluido es una sustancia que se deforma continuamente en el tiempo, al ser sometida a un esfuerzo cortante (esfuerzo tangencial), sin importar cuán pequeño sea. Todos los fluidos están compuestos de moléculas que se encuentran en movimiento constante. La característica fundamental que define a los fluidos es su incapacidad para resistir esfuerzos cortantes (lo que provoca que carezcan de forma definida).

En la naturaleza, existen varios estados de agregación de la materia, que van desde el más libre (aquellos en donde sus partículas prácticamente no interactúan entre sí) hasta el más estático: gases, plasma y líquidos; respectivamente. Todos ellos cumplen las propiedades de los fluidos en mayor o menor medida.

Se denomina ***gas*** al estado de agregación de la materia que no tiene forma ni volumen propio. Este tipo de fluido tiene como característica principal una composición basada en moléculas no unidas, expandidas y con poca fuerza de atracción, haciendo que no tenga volumen ni forma definida, provocando que este se expanda para ocupar todo el volumen del recipiente que la contiene. Con respecto a los gases, las fuerzas gravitatorias y de atracción entre partículas resultan insignificantes.

De acuerdo a [4], en física y química se denomina ***plasma*** a un fluido en forma gaseosa constituido por partículas cargadas de iones libres y cuya dinámica presenta efectos colectivos dominados por las interacciones electromagnéticas de largo alcance entre las mismas. Con frecuencia se habla del plasma como un estado de agregación de la materia con características propias, diferenciándolo de este modo del estado gaseoso, en el que no existen efectos colectivos importantes. Este elemento se encuentra presente en las auroras boreales, los bombillos fluorecentes, las pantallas de televisor, etc.

El ***líquido*** es un estado de agregación de la materia en forma de fluido altamente incompresible. Esto significa que su volumen es, muy aproximadamente, constante en condiciones de temperatura y presión moderadas.

La mecánica de fluidos es la rama de la física que estudia el comportamiento de estos fluidos, su movimiento y las fuerzas que influyen en ellos. También estudia las interacciones entre el fluido y el contorno que lo limita, basándose en todas las hipótesis referentes a los medios continuos.

Dado que existen bastantes coincidencias entre unos y otros tipos de fluidos, no existe una clasificación exacta de los mismos. La Figura 1 intenta mostrar una clasificación general de la mecánica de fluidos sobre la base de las características físicas observables de los campos de flujo.

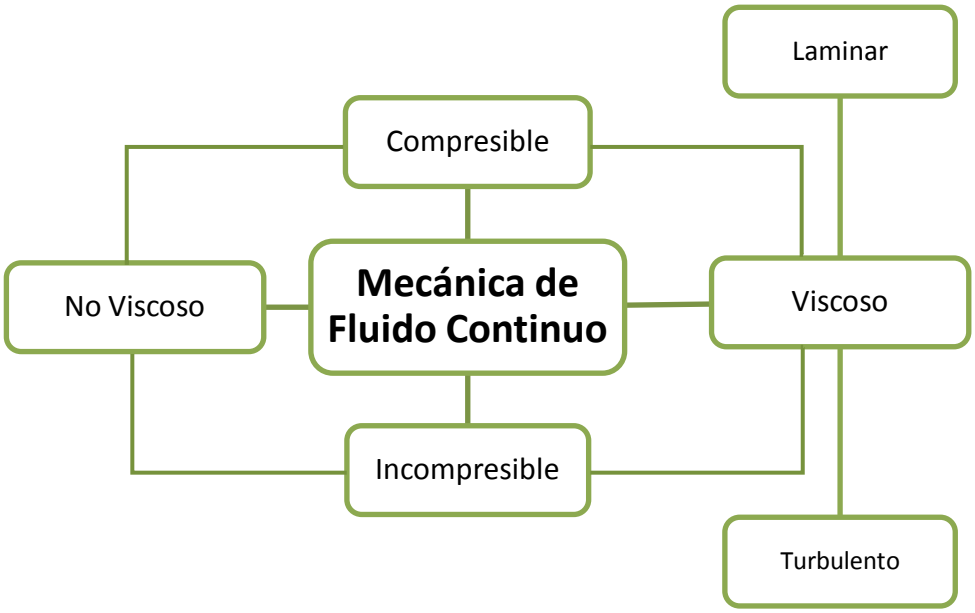


Figura 1: Diagrama de Mecánica de fluido continuo.

A continuación se describen las propiedades más importantes de un fluido.

a) Compresibilidad

La compresibilidad es una propiedad existente en todos los fluidos que está determinada básicamente por el cambio de densidad. En un fluido usualmente hay cambios en la presión,

asociados con cambios en la velocidad. En general, estos cambios de presión inducirán a cambios de densidad, los cuales influyen en el fluido.

Cuando es posible que el volumen de un fluido cambie drásticamente, se dice que el fluido es compresible. En este caso, las variaciones de densidad dentro del fluido no son depreciables.

Algunos fluidos tienen una tasa de compresibilidad muy baja, en cuyo caso el cambio de densidad es insignificante por lo que se considera al fluido incompresible.

La mayoría de los gases son fluidos compresibles, mientras que los líquidos, generalmente son incompresibles.

Para determinar si un fluido es compresible o no, es necesario evaluar la velocidad del fluido en relación a la velocidad del sonido en el fluido. El parámetro utilizado para evaluar esto se denomina número de Mach M:

$$M = V/c \qquad (2.1)$$

En donde M es la razón de la velocidad del flujo V, a la velocidad del sonido c. Para que el flujo pueda ser considerado incompresible, debe cumplirse que $M < 0.3$.

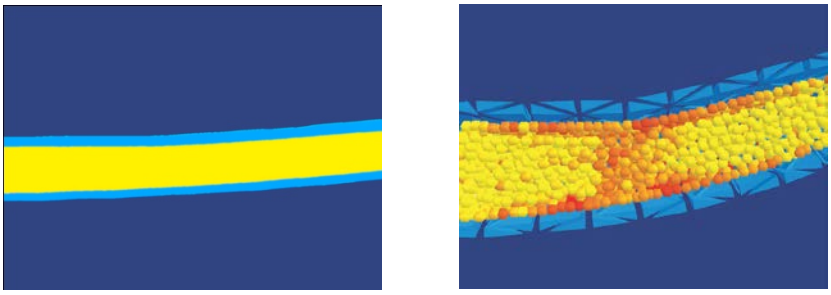
A efectos de este trabajo de investigación solo trabajaremos con flujo incompresible, por ser el agua, el objeto de estudio, un fluido incompresible.

b) Continuidad

En la naturaleza los materiales no son perfectamente continuos, sino que cada elemento que conforma un material es discreto en magnitudes de escalas muy pequeñas. Sin embargo, en los fluidos la continuidad se presenta en mayor escala.

Un fluido está compuesto por millones de moléculas que interactúan entre sí de acuerdo a un conjunto de fuerzas externas e internas. Gracias a la acción de dichas fuerzas, las moléculas se encuentran en constante movimiento.

En muchas aplicaciones de ingeniería, es relevante únicamente lo que ocurre a nivel global entre las moléculas que conforman el fluido, es decir, lo que realmente podemos percibir y visualizar. Debido a esto, se considera que el fluido está idealmente compuesto de una sustancia infinitamente divisible (es decir, como un continuo) y se pasa por alto el comportamiento de las moléculas individuales. En otras palabras, cuando analizamos el fluido externamente, la velocidad de cada molécula es depreciable, por lo que podemos asumir que todas las moléculas tienen una velocidad media. Esta es la base de la mecánica de fluidos clásica. Sin embargo, al analizar el fluido de cerca, sería un error asumir que todas las moléculas se desplazan a la misma velocidad. En este caso, la hipótesis del continuo no es válida. En la Figura 2.a se muestra el fluido como un continuo, pero al hacer un acercamiento en la Figura 2.b podemos apreciar las partículas que se desplazan a diferentes velocidades. La hipótesis de un continuo resulta válida para estudiar el comportamiento de los fluidos en condiciones normales.



(a) (b)

Figura 2: La figura (a) muestra el fluido como un continuo. En la figura (b) se pueden apreciar las partículas rápidas que se muestran en color amarillo (claro), las partículas más lentas en rojo (oscuro).

Una de las consecuencias de la hipótesis del continuo es que cada una de las propiedades de un fluido debe tener un valor definido en cada punto del espacio. De esta manera, propiedades como la densidad, temperatura, velocidad, etc; pueden considerarse como funciones continuas de la posición y del tiempo. Al considerar al fluido como un continuo en materia, entonces es posible estudiar sus propiedades como un continuo también. Así es posible evaluar su valor en un punto determinado en el espacio y en un momento dado, a pesar de que varíen constantemente en ambas.

c) Fluido Newtoniano y no Newtoniano

Cuando no es aplicado sobre el fluido un esfuerzo cortante, no existe deformación. Según la relación que existe entre el esfuerzo cortante aplicado y la rapidez de deformación resultante, los fluidos se pueden clasificar en:

- **Fluidos Newtonianos:** Aquellos fluidos donde el esfuerzo cortante es directamente proporcional a la rapidez de deformación.
- **Fluidos no Newtonianos:** Fluidos donde el esfuerzo cortante no es directamente proporcional a la rapidez de deformación. En este caso, la deformación depende de otros factores, como la viscosidad, por lo que cuando el fluido sea afectado por una fuerza externa se comportará como un sólido por pequeños instantes de tiempo. En un fluido no newtoniano la viscosidad no es constante, sino que varía según el gradiente de presión que se aplique sobre él.

Todos los fluidos tienen un valor de viscosidad, algunos más pequeños que otros. El agua tiene un valor de viscosidad bastante bajo por lo que se considera un fluido newtoniano. El valor de viscosidad varía de acuerdo a la temperatura del fluido. En la Tabla 1 podemos apreciar la variación en el valor de viscosidad del agua a diferentes temperaturas.

$\boldsymbol{t(C°)}$	$\boldsymbol{\mu(mPa \cdot s)}$
0	1,8
20	1
25	0,894
60	0,65

Tabla 1: Coeficientes de viscosidad dinámica del agua [5].

Para este trabajo no se tomará en cuenta los efectos termodinámicos de los fluidos, se asumirá una temperatura ambiental constante de 20° y por lo tanto un valor de viscosidad de 1 $mPa \cdot s$.

En el SI (Sistema Internacional de unidades), la unidad de viscosidad dinámica está representada por $Pa \cdot s$, donde s representa un segundo y Pa, una unidad de presión que recibe el nombre de pascal, la cuál es equivalente a un newton por metro cuadrado N/m^2.

El coeficienciente de viscosidad dinámica que se presenta en un fluido homogéneo puede entenderse como una diferencia de velocidad de un metro (m) por segundo (s) entre dos

planos paralelos separados a una distancia de un metro, el movimiento rectilíneo y uniforme de una superficie plana de un metro cuadrado provoca una fuerza retardatriz de un newton[5].

d) Viscosidad

Si evaluamos la deformación de dos fluidos newtonianos distintos, es probable que se deformen con diferente rapidez para una misma fuerza cortante. Esto ocurre porque la viscosidad de uno es mayor que la del otro. A mayor viscosidad, mayor será la resistencia que imponga el fluido a la deformación.

La viscosidad es una manifestación del movimiento molecular dentro del fluido. Las moléculas de regiones con alta velocidad global chocan con las moléculas que se mueven con una velocidad global menor, y viceversa. Estos choques permiten transportar cantidad de movimiento de una región del fluido a otra (**advección**). Como la energía se conserva, al chocar las moléculas entre sí, parte de la energía se transfiere de unas a otras y esto se traduce en una variación en la velocidad y en la temperatura. Ya que los movimientos moleculares aleatorios se ven afectados por la temperatura del medio, la viscosidad resulta ser una función de la temperatura.

En la mecánica de fluidos se emplea muy frecuentemente la viscosidad cinemática v:

$$v = \frac{\mu}{\rho} \qquad (2.2)$$

en donde μ es el coeficiente de viscosidad dinámico y ρ la densidad de la partícula.

Un fluido viscoso puede clasificarse como **laminar** o **turbulento** de acuerdo a la estructura interna del mismo. Se dice que es laminar cuando su estructura se caracteriza por el movimiento de láminas o capas, mientras que en el fluido turbulento su estructura se caracteriza por los movimientos tridimensionales aleatorios de las partículas de fluido, superpuestos al movimiento promedio. En la Figura 3 se puede apreciar un fluido turbulento.

Figura 3: Comparación de las formas de una llama para diferentes grados de expansión de gases. De izquierda a derecha se puede apreciar como el fluido se hace más turbulento en la simulación [6].

Dependiendo de las condiciones físicas o ambientales en las que se encuentre el agua podría comportarse como un fluido turbulento o laminar. En el caso del agua de lluvia podemos observar un comportamiento turbulento generado por la colisión de múltiples gotas de agua, sobre pequeñas masas de agua acumuladas como charcos o fluyendo por canales. Para simular esta propiedad física se desarrollan algoritmos que tomen en cuenta propiedades como la tensión superficial, refracción de la luz, cáustica, entre otros.

2. Ecuaciones de Navier-Stokes

A continuación se explica el modelo teórico planteado por Navier-Stokes para la simulación de fluidos. La descripción que se incluye en este capítulo está basada en las notas de curso de Matthias Müller y Robert Bridson [7].

Claude-Louis Navier y George Gabriel Stokes desarrollaron una serie de ecuaciones en derivadas parciales no lineales que simulan el comportamiento físico de un fluido y describen el movimiento del mismo, a partir de algunos principios de conservación de la energía mecánica y de la termodinámica.

a) Ecuación del momentum

Una forma sencilla de imaginar el comportamiento de un fluido es como una colección muy grande de partículas que se mueven conducidas por una serie de fuerzas externas. Para analizar como el fluido se acelera debido a la influencia de las fuerzas externas que actúan sobre él, partimos de la tercera ley de Newton $\mathrm{F} = \mathrm{ma}$, en donde F es la sumatoria de todas las fuerzas, m es la masa del fluido y a la aceleración.

Como ocurre con todos los cuerpos, la principal fuerza que actúa sobre ellos es la gravedad $\vec{g}$. Esta fuerza ha de afectar a cada una de las partículas que conforman el fluido. A su vez, el fluido tiene una serie de elementos que influyen directamente en su comportamiento, como son la presión y la viscosidad. Así mismo, cada una de las partículas que componen el fluido, ha de tener una masa m, un volumen V y una velocidad $\vec{u}$.

La primera de las fuerzas que influyen en el comportamiento de un fluido es la presión, denotada por el vector $\vec{p}$, el cual representa la fuerza normal de cada una de las partículas del fluido. Sin embargo, lo que realmente importa es la fuerza neta en una partícula, pero si la presión es igual en todas direcciones la fuerza neta será cero. Por lo tanto, se considera únicamente el desequilibrio de presión en una partícula en donde a un lado de ella se encuentra un campo de mayor presión que en el otro. En otras palabras, $\vec{p}$ apunta a las zonas de baja presión y es contraria a aquellas de alta presión. Una manera de medir este desbalance en la presión es tomando el gradiente negativo de la presión $-\nabla\vec{p}$. Así podremos destacar aquellos puntos en donde la presión varía entre regiones de alta presión a regiones de baja presión. Las regiones de alta presión de un fluido han de desplazar a aquellas de menor presión. Para obtener la fuerza de presión es necesario multiplicar por el volumen V de cada partícula: $-V\nabla\vec{p}$.

Otra de las fuerzas que tienen que ver con el comportamiento del fluido es la viscosidad $\vec{v}$, la cual permite a un fluido resistirse a la deformación, en mayor o menor medida. Esta fuerza intenta hacer que el movimiento de cada partícula se acerque a la velocidad promedio de las partículas cercanas. El operador diferencial Laplaciano $\nabla \cdot \nabla$ medirá qué tanto dista la velocidad de una partícula en relación con el promedio. Al aplicarlo al coeficiente de viscosidad dinámico μ y multiplicarlo por el volumen de cada partícula obtenemos la fuerza de viscosidad $V\mu\nabla \cdot \nabla\vec{u}$.

De esta manera, podemos reescribir la ecuación de Newton de la siguiente manera:

$$F = ma = m\left(\frac{\partial \vec{u}}{\partial t}\right) = (m\vec{g} - V\nabla\vec{p} + V\mu\nabla \cdot \nabla\vec{u}) \qquad (2.3)$$

Si se toma un número muy pequeño de partículas la aproximación puede tener muchos errores. Por esta razón la cantidad de partículas tiende a infinito y el tamaño de ellas tiende a cero. Sin embargo, esto haría que el volumen y la masa de las partículas también tiendan a cero. Para evitar este problema se divide toda la ecuación por el volumen V. Tomando en cuenta que la densidad $\rho = \frac{m}{V}$ se tiene que:

$$\rho\left(\frac{\partial \vec{u}}{\partial t}\right) = (\rho\vec{g} - \nabla\vec{p} + \mu\nabla \cdot \nabla\vec{u}) \qquad (2.4)$$

Ahora se divide por la densidad y se reordenan los términos:

$$\left(\frac{\partial \vec{u}}{\partial t}\right) + \frac{1}{\rho}\nabla\vec{p} = \vec{g} + \frac{\mu}{\rho}\nabla \cdot \nabla\vec{u} \qquad (2.5)$$

Para simplificar esto se define la viscosidad cinemática como $v = \mu/\rho$ y se obtiene la ecuación del momentum:

$$\left(\frac{\partial \vec{u}}{\partial t}\right) + \frac{1}{\rho}\nabla\vec{p} = \vec{g} + v\nabla \cdot \nabla\vec{u} \qquad (2.6)$$

Sin embargo, para obtener la forma tradicional de la ecuación del momentum de Navier-Stokes se debe agregar el **material derivativo** a la ecuación. Para ello es necesario comprender los dos enfoques existentes en el estudio del comportamiento del fluido: el enfoque lagrangiano y el enfoque euleriano.

En el enfoque euleriano el fluido se estudia desde un punto fijo externo al fluido. Los elementos del fluido (como densidad, temperatura, velocidad, etc.) son medidos en aquellos puntos que van cambiando en el tiempo en ubicaciones fijas en el espacio. El enfoque euleriano considera al fluido como un todo, en lugar de estudiar el comportamiento aislado de cada partícula.

El enfoque lagrangiano estudia al fluido como un sistema de partículas, en donde cada molécula del fluido se trata como una partícula. Cada una de estas partículas tiene sus propios valores de densidad, temperatura, etc.

La clave para conectar ambos puntos de vista es el **material derivativo**. En el punto de vista lagrangiano, cada partícula tiene su posición $\vec{x}$ y su velocidad $\vec{u}$. Si queremos determinar el valor de algún elemento q de manera general (q puede ser temperatura, velocidad, densidad, etc.) simplemente evaluamos la función q en el tiempo y el espacio: $q(t, \vec{x})$, así podemos considerar a q una variable euleriana.

Por otro lado, cada partícula tiene un valor para q. Para saber que tan rápido cambia q en cada partícula tomamos la derivada de q:

$$\frac{d}{dt}q(t,\vec{x}) = \frac{\partial q}{\partial t} + \nabla q \cdot \frac{d\vec{x}}{dt} \qquad (2.7)$$

$$\frac{d}{dt}q(t,\vec{x}) = \frac{\partial q}{\partial t} + \nabla q \cdot \vec{u} \equiv \frac{Dq}{Dt} \qquad (2.8)$$

El primer término, $\partial q/\partial t$ indica que tan rápido cambia q en el tiempo (medida euleriana), mientras que el segundo término indica cuánto está cambiando de manera general. De esta manera, podemos ver como q se mueve en función de un campo de velocidad $\vec{u}$. Esto es conocido como **advección** o transporte.

Si evaluamos la ecuación de advección en la velocidad, veremos al vector $\vec{u}$ aparecer dos veces, como el campo de velocidad que está moviendo al fluido y como el valor q que está siendo movido. Tenemos así la advección de la velocidad $\vec{u} = (u, v, w)$.

$$\frac{D\vec{u}}{Dt} = \begin{bmatrix} Du/Dt \\ Dv/Dt \\ Dw/Dt \end{bmatrix} = \begin{bmatrix} \partial u/\partial t + \vec{u} \cdot \nabla u \\ \partial v/\partial t + \vec{u} \cdot \nabla v \\ \partial w/\partial t + \vec{u} \cdot \nabla w \end{bmatrix} = \frac{\partial \vec{u}}{\partial t} + \partial \vec{u} \cdot \nabla \vec{u} \qquad (2.9)$$

Se agrega el término de la advección de la velocidad que indica la cantidad de velocidad cambiante a la Ecuación 2.6 y así se obtiene la ecuación tradicional del momentum planteada por Navier-Stokes con el material derivativo, válida para ambos enfoques:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u}.\nabla\vec{u} + \frac{1}{\rho}\nabla p = \vec{g} + v\nabla.\nabla\vec{u} \qquad (2.10)$$

b) Incompresibilidad

Aún cuando ningún fluido es netamente incompresible, para efectos de animación y simulación se asume que todos lo son por razones de simplicidad.

Supongamos que tomamos un trozo de fluido en un instante de tiempo. Al volumen de este trozo le llamamos Ω y a su superficie límite $\partial\Omega$. Si medimos que tan rápido el volumen de este trozo está cambiando, integramos la componente normal de su velocidad a su alrededor:

$$\frac{d}{dt}Volume(\Omega) = \iint_{\partial\Omega} \vec{u}\cdot\hat{n} \qquad (2.11)$$

En un fluido incompresible, el volumen ha de permanecer constante, por lo tanto su tasa de cambio debe ser cero.

$$\iint_{\partial\Omega} \vec{u}\cdot\hat{n} = 0 \qquad (2.12)$$

Por el Teorema de Divergencia:

$$\iiint_{\Omega} \nabla\cdot\vec{u} = 0 \qquad (2.13)$$

Esta ecuación debe cumplirse para algún Ω que forme parte del fluido, y la integral de una constante es la constante misma. Tenemos así la condición de incompresibilidad de Navier-Stokes:

$$\nabla . \vec{u} = 0 \qquad (2.14)$$

3. Smoothed Particle Hydrodynamics

El método *Smoothed Particle Hydrodynamics* (SPH) es un método lagrangiano que, entre otras aplicaciones, es comúnmente utilizado para representar fluidos en movimientos. De esta forma, el mallado que representa el fluido no es una red fija o estática sino que se mueve de acuerdo al comportamiento del fluido.

El SPH trabaja dividiendo el fluido en un conjunto de elementos discretos llamados partículas. Esto es gobernado por una función kernel W, la cual incluye comúnmente la función Gaussiana y un spline cúbico (los kernels más habituales son funciones exponenciales o splines). Esta última función es exactamente igual a cero para las partículas más allá de dos longitudes de suavizado (a diferencia de la de Gauss, donde hay una pequeña contribución a cualquier distancia finita). Esto es una ventaja desde el punto de vista computacional, ya que la contribución de las partículas distantes no son tomadas en cuenta al considerarse irrelevantes.

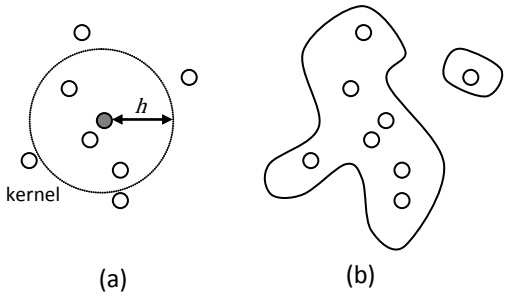


Figura 4: Modelo computacional Smoothed Particle Hydrodynamics (a). Algunas partículas y su núcleo de inferencia con el radio h (b).

Las contribuciones de cada partícula a una propiedad se ponderan de acuerdo a su distancia en relación a la partícula de interés, y su densidad. Estas partículas tienen una distancia espacial conocida como longitud de suavizado, representada típicamente en las ecuaciones con la variable h. Esto significa que la cantidad física de una partícula puede ser obtenida por la sumatoria de las propiedades relevantes de todas las partículas que se encuentran dentro del rango del kernel[8], tal como se indica en la Figura 4.

En analogía con la interpolación mediante elementos finitos, la aproximación espacial en el método SPH original puede escribirse en términos de "funciones de forma SPH". Esta aproximación se construye a partir del concepto de estimación "tipo kernel" (*kernel estimate*), inspirada por la siguiente propiedad de la función delta de Dirac [9]:

$$u(\boldsymbol{x}) = \int_{\boldsymbol{y}\in\Omega} u(\boldsymbol{y})\delta(\boldsymbol{x}-\boldsymbol{y})d\Omega \qquad (2.15)$$

La estimación kernel $\langle u(\boldsymbol{x})\rangle$ de una cierta función $u(\boldsymbol{x})$ se define como:

$$\langle u(\boldsymbol{x})\rangle = \int_{\boldsymbol{y}\in\Omega} u(\boldsymbol{y})W(\boldsymbol{x}-\boldsymbol{y},\boldsymbol{\rho})d\Omega \qquad (2.16)$$

y su correlato SPH discreto, $\hat{u}(\boldsymbol{x})$, es:

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{n} u_j W(x-x_j,h)V_j \qquad (2.17)$$

dónde Ω es el dominio del problema, discretizado en un conjunto de n nodos o partículas, $W(x-x_j,h)$ es un kernel o función de suavizado (*smoothing function*) centrado en la partícula j y V_j es el volumen asociado a la partícula j y ρ la densidad. Análogamente a lo que ocurre en el método de elementos finitos, la aproximación dada por *(2.17)* puede escribirse en términos de "funciones de forma SPH", de la forma [7]:

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{n} u_j N_j(\boldsymbol{x}) \qquad (2.18)$$

$$N_j(\boldsymbol{x}) = W\big(x - x_j, \rho\big)V_j \qquad (2.19)$$

El gradiente de $\hat{u}(\boldsymbol{x})$ puede evaluarse como:

$$\nabla_{\boldsymbol{x}}\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{n} u_j \nabla_{\boldsymbol{x}} N_j(\boldsymbol{x}) = \sum_{j=1}^{n} u_j \nabla_{\boldsymbol{x}} W_j(\boldsymbol{x})\, V_j \qquad (2.20)$$

En la práctica, suelen emplearse expresiones alternativas para calcular $\nabla_{\boldsymbol{x}}\hat{u}(\boldsymbol{x})$, imponiéndose determinadas propiedades de conservación en las ecuaciones discretas. Análogamente a lo expuesto más arriba se podrían obtener estimaciones de las derivadas de orden superior de la función. Nótese que los valores reconstruidos de la función $u(x)$ y sus derivadas en un determinado punto se obtienen empleando solamente los valores de dicha función en nodos vecinos y cierta información de "bajo nivel" acerca de la distribución de nodos (distancias entre nodos), sin referencia alguna a una estructura de datos tipo malla.

La ecuación para cualquier cantidad A en cualquier punto x está dada por:

$$A(x) = \sum_{j} m_j \frac{A_j}{\rho_j} W\big(|x - x_j|, h\big) \qquad (2.21)$$

donde m_j es la masa de la partícula j y ρ_j es su densidad, A_j es el valor de cantidad A en la partícula j, x es la posición de cada partícula y *W* es la función kernel. La resolución del método se puede ajustar fácilmente a otras variables como la densidad. Por ejemplo, la densidad de una partícula podemos obtenerla a través de:

$$\rho(x) = \sum_{j} \left(m_j \int W\big(|x - x_j|\big) \right) \qquad (2.22)$$

Así, la densidad ρ_i de una partícula i es simplemente $\rho_i = \rho(x_i)$ y puede ser expresada como:

$$\rho_i = \sum_j \left(m_j \frac{\rho_j}{\rho_j} W(|x - x_j|, h) \right) = \sum_j \left(m_j W(x - x_j, h) \right) \qquad (2.23)$$

Del mismo modo, la derivada espacial de una cantidad se puede obtener mediante la integración por partes del gradiente para la cantidad física de la función kernel. Una de las ventajas de esta formulación es que el gradiente de un kernel se puede calcular fácilmente mediante la sustitución del kernel por el gradiente del mismo:

$$\nabla A_s(x) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(|x - x_j|, h) \qquad (2.24)$$

El método SPH es muchas veces preferido antes que las tradicionales técnicas basadas en *malla* por las ventajas que tiene sobre estos. En principio, garantiza la conservación de masa sin la necesidad de hacer cálculos extras, debido a que las partículas representan la masa en sí mismas. Además, podemos calcular la presión a partir de la contribución en peso proveniente de las partículas vecinas sin necesidad de resolver un sistema lineal de ecuaciones. Finalmente, a diferencia de los métodos basados en malla que deben realizar un seguimiento del fluido de las fronteras, SPH crea una superficie libre de los fluidos que interactúan en dos fases directamente, las partículas que representan el fluido más denso (normalmente agua) y el espacio vacío que constituye el fluido más ligero (por lo general aire). Por esta razón es posible utilizar SPH para simular movimiento de fluidos en tiempo real.

Al igual que en los métodos basados en malla, SPH requiere de alguna técnica que permita generar una superficie geométrica, usando alguna técnica de poligonización como *Marching Cubes* [10], como se puede ver en la Figura 5.

En desventaja con los métodos basados en malla, SPH requiere de una gran cantidad de partículas para producir simulaciones equivalentes en resolución a las basadas en malla, de las cuales muchas no han de ser renderizadas. Por esta razón, el método SPH es preferido antes que las técnicas basadas en malla, en aquellas aplicaciones en donde la interactividad tiene más importancia que la exactitud.

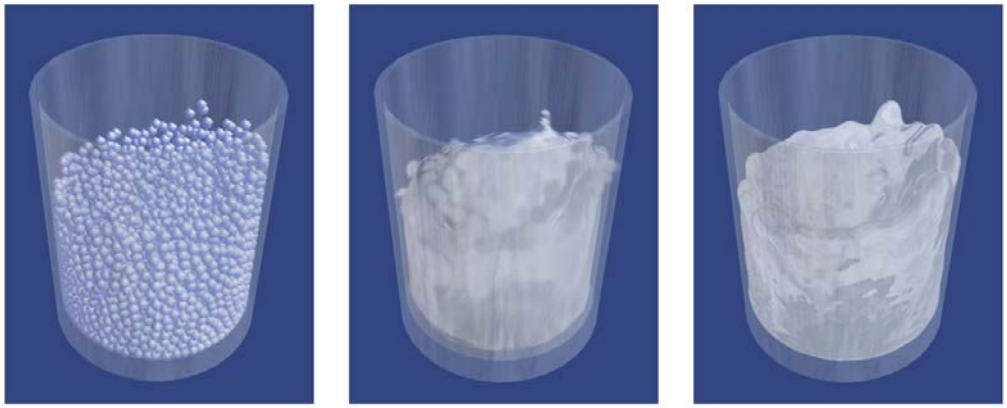


Figura 5: Remolino en un vaso producido por un campo de fuerza de rotación[8]. La imagen (a) muestra las partículas. La imagen (b) muestra la superficie renderizada con la técnica point splatting y la imagen (c) muestra la superficie triangulada con Marching Cubes.

Aunque el tamaño de la longitud de suavizado se puede fijar en el espacio y el tiempo, esto no se aprovecha al máximo en SPH. La resolución de la simulación puede ser adaptada automáticamente a sí misma dependiendo de las condiciones locales, mediante la asignación a cada partícula de su longitud de suavizado propia y parámetros de variación en el tiempo.

Por ejemplo, en una zona muy densa donde varias partículas están muy cerca una de otra la longitud de suavizado puede hacerse relativamente corta, obteniendo una alta resolución espacial. Por el contrario, en regiones de baja densidad donde las partículas individuales están bien apartadas y la resolución es baja, la longitud de suavizado puede incrementarse optimizando los cálculos en la región de interés. Conjuntamente con la ecuación de estado y un integrador, SPH puede simular flujos hidrodinámicos eficientemente. Sin embargo, la formulación de viscosidad artificial tradicionalmente usada en SPH tiende a disminuir los choques y la discontinuidad del contacto mucho más que los esquemas basados en malla.

CAPÍTULO III. Simulación de aguas de lluvia utilizando PhysX

En este capítulo explicaremos todo lo relacionado a la simulación del fluido con PhysX y otros detalles relevantes de la construcción de la escena. Así mismo se mencionarán aspectos generales de las precipitaciones que han de ser tomados en cuenta para simular aguas de lluvia.

1. Lluvias

En el área de la meteorología, la precipitación es cualquier forma de hidrometeoro que cae del cielo en forma de lluvia, llovizna, nieve o granizo y llega a la tierra. La cantidad de precipitación sobre un punto de la superficie terrestre es llamada pluviosidad, o monto pluviométrico[11]. Según la definición oficial de la Organización Meteorológica Mundial[12] la lluvia es la precipitación de partículas líquidas de agua, de diámetro mayor de 0,5 mm o de gotas menores, pero muy dispersas.

Cuando el agua se condensa, producto de la evaporación, se acumulan en forma de nubes de vapor de agua que, al saturarse caen en forma de precipitación. La precipitación se mide en milímetros de agua, o litros caídos por unidad de superficie (m^2), lo que es equivalente al agua que se acumularía en una superficie horizontal e impermeable de 1 metro cuadrado durante el tiempo que dure la precipitación.

Nótese que hay dos elementos primordiales en la medición de la pluviosidad: La intensidad y la duración de la lluvia. Ambos están asociados por el hecho de que durante un mismo período de tiempo, al aumentarse la duración de la lluvia disminuye su intensidad media. Dicho de otra forma, si estudiamos dos precipitaciones diferentes pero de la misma magnitud y ocurridas en el mismo tiempo de retorno, la intensidad media aumentará cuanto menor sea la duración de dicha precipitación. Dicha formulación se conoce como Curvas IDF (Intensidad-Duración-Frecuencia), las cuales son determinadas, para cada sitio particular, con procedimientos estadísticos basados en observaciones de larga duración [13].

Las precipitaciones en Venezuela varían de acuerdo a la distribución espacial (hay regiones del país en donde la intensidad de precipitación es mayor que en otras) y al período del año (período de lluvias o de sequía). Incluso pueden verse afectadas por fenómenos atmosféricos que alteran la intensidad de precipitación normal con respecto a la época del año.

En período lluvioso (Junio y Julio) las precipitaciones pueden superar incluso los 600mm en la cuenca alta del Caroní, mientras que en la temporada de sequía (Febrero y Marzo) la mayor parte del país presenta una media de precipitación menor a los 50mm [14].

Para este trabajo se tomaran en cuenta precipitaciones desde los 50mm hasta los 200mm, lo equivalente a la media de precipitación a nivel nacional durante el tiempo de transición entre los períodos de lluvia y sequía.

2. Simulación de lluvia con PhysX

PhysX surgió como un chip capaz de elaborar simulaciones físicas que por su complejidad no pueden ser realizadas por un CPU en tiempo real. Este fue creado por la compañía AGEIA Technologies, la cual posteriormente fue comprada por Nvidia Corporation, y es cuando se le da realmente interés a la aceleración de física por hardware. Esta conocida empresa estadounidense fabricante de procesadores gráficos, incluyó esta tecnología en sus tarjetas gráficas de la serie GeForce 8000 en adelante. El procesador PhysX puede ser programado para la mayoría de cálculos de la física clásica a través del PhysX SDK, el cual incluye: gravedad, colisiones, presión, tensión, deformaciones, etc.

El API de PhysX ofrece tecnología de simulación para un número arbitrario de escenas de la física. En particular con la física de fluidos tenemos dos alternativas:

- Creación volumétrica de fluidos.
- PhysX FX, que utiliza sistemas de partículas inteligentes que permiten simular el fuego, el humo y la niebla de una manera natural, como el humo encerrado en una habitación que se eleva hasta el techo, llenándola por completo hasta que se filtre por las ventanas.

Para utilizar PhysX se emplea el PhysX SDK (*Software Development Kit*), el cual está implementado en C++, e internamente está organizado como una jerarquía de clases, ver [15]. Cada clase tiene una clase base abstracta, desde donde el usuario puede usar una serie de funcionalidades de acceso, implementadas en el descriptor (la clase concreta). Además, algunas funciones de utilidad se exportan como funciones de C. Las clases de interfaz han de seguir algunas convenciones de codificación:

- Todas las clases tienen un archivo de cabecera con el mismo nombre de archivo del nombre de la clase
- Los tipos y las clases comienzan con una letra mayúscula
- Las clases de interfaz siempre empiezan con “Nx”
- Los métodos y variables comienzan con letras minúsculas
- Los valores de retorno y parámetros en lugares donde el valor es NULL, son aceptables si el código utiliza sintaxis de apuntadores (*)
- Los valores de retorno y parámetros en lugares donde el valor es NULL, son inaceptables si el código utiliza sintaxis de referencia (&)
- Si alguna funcionalidad definida por el usuario depende de la ejecución, es necesario implementar una interfaz para este propósito (por ejemplo el manejo de memoria).

La clase NxPhysicsSDK es una clase *abstract singleton factory* usada para instanciar objetos y acceder variables globales que sean afectadas en diferentes clases. Para tener acceso a esta clase debe invocarse el método NxCreatePhysicsSDK() de la siguiente forma:

```
NxPhysicsSDK * myWorld = NxCreatePhysicsSDK(NX_PHYSICS_SDK_VERSION);
```

Para cada simulación es necesario crear una escena, es decir, una instancia de la clase NxScene(). Una escena es una colección de cuerpos, fuerzas y efectos que interactúan entre sí. La escena simula el comportamiento de los objetos presentes en ella en el tiempo. Pueden existir varias escenas al mismo tiempo pero cada cuerpo, fuerza o efecto es específico a una escena, no puede ser compartido.

PhysX Fluids permite la simulación de líquidos y gases utilizando un sistema de partículas y emisores, utilizando el método SPH (*Smoothed Particle Hydrodynamics*) o sistemas de partículas simples. En este último caso no se le da al usuario la posibilidad de controlar la simulación, es decir, la simulación se rige por un comportamiento físico y no ha de ser una simulación guiada para animación [1]. Para este trabajo utilizamos el método SPH para la simulación puesto que lo que queremos es el comportamiento físico del agua.

Una escena puede tener varios tipos objetos, entre ellos los fluidos como se muestra en la Figura 6.

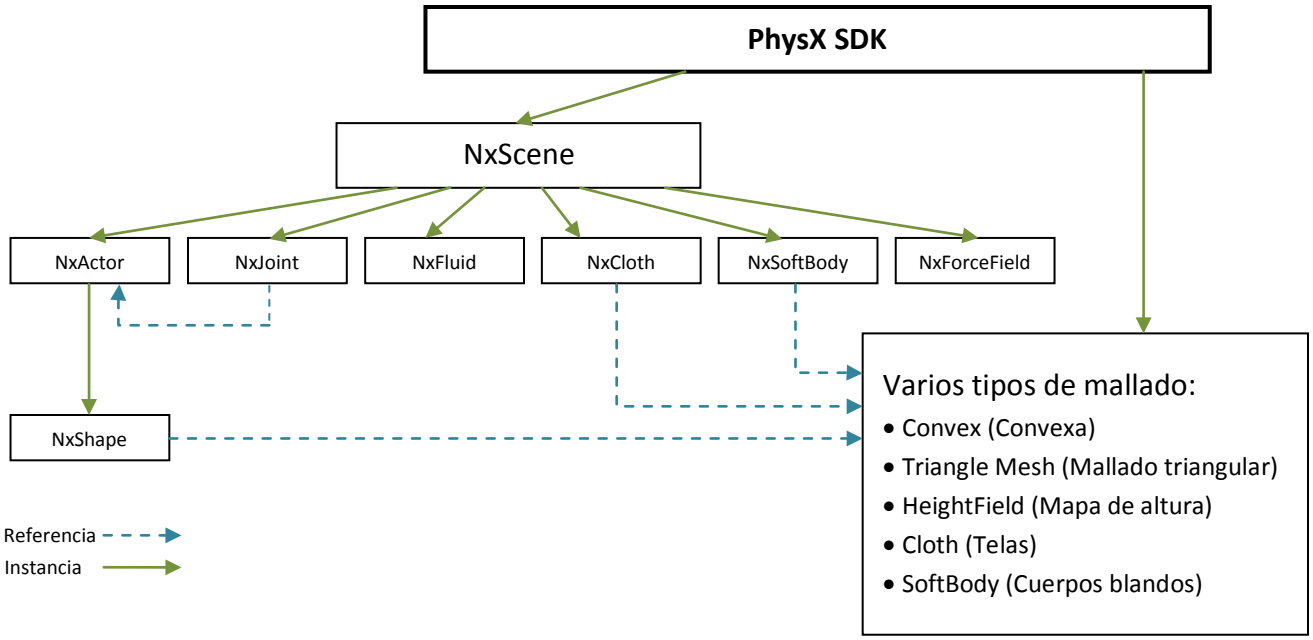


Figura 6: Diagrama de la arquitectura de PhysX SDK

a) Creación del fluido

Existen dos maneras de crear fluidos. Es posible generar un sistema de partículas para simular el fluido a través de un emisor (*emitter*) o simplemente instanciar las partículas con sus parámetros inicializados. Para eliminar las partículas es posible absorber el fluido restante a través de un drenaje (*drain*) o darles un tiempo de vida a cada partícula. La Figura 7 ilustra el ciclo de vida de un fluido generado por el emisor y absorbido por un drenaje.

Figura 7: Comportamiento de un fluido utilizando emisores y drenajes.

En este trabajo la creación de los fluidos se hace a partir de emisores para simular con más precisión el origen de las precipitaciones a partir de la condensación y saturación de vapor de agua en una nube. Además el emisor permite la emisión constante de partículas del fluido, manteniendo el mismo número de partículas por cada *frame* de la simulación. Esta tasa puede

ser ajustada dinámicamente, modificando en tiempo de ejecución el estado del atributo *NxFluidEmitterDesc::rate.*

El *emitter* está asociado directamente a un *shape*, al cual se le adjunta un *emitter*, es decir, tendrá las propiedades y el comportamiento de un *emitter*. El *emitter* tendrá un *frameShape*, un apuntador al actor *NxShape* al cual el *emitter* está asociado. En este caso el *shape* asociado es un *NxBoxShape* cuya dimensión será un poco más grande que el área del techo, la idea es que la "nube" simulada por el *emitter* cubra la estructura en su totalidad.

Los *drains* no son objetos propiamente dichos, son *shapes* con la bandera *NX_SF_FLUID_DRAIN* aplicada al atributo *NxShapeDesc::shapeFlags*, causando que las partículas colisionen con ellos para ser removidas de la simulación. Un *drain* puede tomar la forma de cualquier primitiva de PhysX.

Los *drains* son un método importante para mantener el conteo de las partículas y su difusión bajo control. Los *drains* han de situarse en la zona alrededor de la cual las partículas del fluido deben detenerse, para evitar que sigan expandiéndose de manera innecesaria contrarrestando el rendimiento.

Idealmente el agua de lluvia colectada por un sistema de drenes y canaletas debe ser dirigida a un sistema de drenaje de aguas de lluvias a través de canales separados de las aguas servidas. Su disposición final debe ser una planta potabilizadora o cualquier curso natural de aguas no contaminado [16]. El conocimiento de las lluvias intensas, de corta duración, es muy importante para dimensionar el drenaje urbano, y así evitar inundaciones en los centros poblados. Además si el agua de lluvia no es drenada correctamente, más allá de lo que ocurre en los techos y de las inundaciones que puedan suscitarse, el agua será absorbida por el suelo, dañando las capas asfálticas o erosionando el suelo donde están los cimientos de las edificaciones y provocar hundimientos o derrumbes.

Sin embargo, este trabajo no contempla el sistema de drenaje de las aguas de lluvia más allá de lo pertinente a una edificación. Solo se está considerando el drenado del agua de lluvia de un techo a través de un sistema de canaletas o drenes.

3. Actores de PhysX

PhysX cuenta con una clase para crear los diferentes elementos que serán parte de la escena. Estos son los actores, instancias de la clase *NxActor*, el objeto principal en la simulación de *Physics SDK*, los cuales son instanciados y están contenidos por la escena como puede apreciarse en la Figura 6. Los actores (*NxActors*) no son más que geometrías a la cuales se le añaden ciertas propiedades para que puedan interactuar de acuerdo a las leyes físicas implementadas por el motor de PhysX.

Un actor puede opcionalmente encapsular el comportamiento de un cuerpo rígido dinámico al agregarle un cuerpo al descriptor del actor en el momento en que es creado, de lo contrario el actor será un objeto estático, fijo en el espacio. Para este trabajo se utilizan solo objetos estáticos.

Cada actor está asociado a una forma (*NxShape*) a partir de la cual se hace la detección de colisiones, lo que hace PhysX es asegurarse de que una forma no intercepte a otra. PhysX SDK simula las formas de acuerdo a leyes físicas que sostienen que, un cuerpo puede ser representado perfectamente por una tensión de inercia y un punto de masa localizado en el centro de masa del objeto.

a) Creación del Actor

Para crear un actor se hace una llamada a NxScene::createActor(), al cual se le asignará luego un descriptor *NxActorDesc* que salva el estado del actor, luego se crea un *NxActorDescBase* sobre quien se configuran todas las propiedades relativas del actor. Algunos de los atributos importantes en *NxActorDescBase* son:

- **globalPose:** Indica la posición del actor en coordenadas de mundo.
- **body:** Es un atributo de tipo *NxBodyDesc* que determina el tipo de actor (estático o dinámico). Si el actor es estático su valor será nulo.
- **density:** Densidad del actor utilizada en el cálculo masa/inercia.
- **flags:** Conjunto de banderas que pueden ser activada a través de un *NxActorFlags* para activar algunas propiedades de los actores.

Dependiendo del tipo de NxShape asociado al actor será necesario configurar algunos otros parámetros relativos a su forma. Para eliminar un actor se llama a NxScene::releaseActor().

CAPÍTULO IV. Fundamentos Arquitectónicos para Sistemas de Techos y Canaletas

En este capítulo describiremos, de manera general, los aspectos más resaltantes en la construcción de un sistema de techos y algunos de los elementos que permiten drenar el agua de lluvia.

El sistema de techo funciona como el elemento primario de resguardo para los espacios interiores de un edificio. La forma y la inclinación del techo deben ser compatibles con el tipo de techumbre que se usa para desviar el agua pluvial hacia un sistema de canaletas y desagües.

1. Terminología de Techos

- **Cumbrera o brocal de techo:** Es la línea horizontal de intersección de la parte superior de dos planos inclinados de un techo.
- **Voladizo:** Es la parte de techo que sobresale a una pared externa o pared maestra en una estructura o edificación.
- **Alero:** Es el borde inferior en voladizo de un techo. También existen aleros inclinados, que corresponden a la orilla inclinada, generalmente proyectante, de un techo inclinado.

2. Detalles Arquitectónicos de Implementación

Los techos pueden ser clasificados según su morfología de la siguiente manera:

- **Techos planos:** Requieren un material de techumbre continuo de membrana y no llevan revestimiento. Los techos planos deben tener una inclinación hacia los drenes del techo que se ubican en los puntos inferiores y que pueden conectarse al sistema de drenaje pluvial del edificio. La pendiente mínima recomendada es de 1:50, lo cual sería equivalente a un ángulo de 1,15° con respecto a los muros.

- **Techos Inclinados:** Descargan fácilmente el agua pluvial a las canaletas de los aleros. La altura y el área de un techo inclinado aumentan con sus dimensiones horizontales, lo que permite clasificarlos en techos de baja pendiente (hasta 3:12) o techos de pendiente media – alta (4:12 a 12:12), de acuerdo a una proporción elevación – longitud.

  Para este trabajo solo se tomarán en cuenta las escalas indicadas en la Tabla 2:

Tipo de Techo Inclinado	Proporción	Equivalente en ángulos
Baja Pendiente	3:12	14°
Pendiente media a alta	4:12	18°
	7:12	30°
	12:12	45°

Tabla 2: Pendientes estándar en techos inclinados [17].

  La pendiente del techo afecta la elección del material con que ha de hacerse el mismo. Las pendientes medias o altas pueden llevar revestimiento de tejas de madera o baldosas, o materiales en hoja para techumbre corrugada, como láminas de zinc, losacero, etc.

  En el caso de las pendientes bajas es necesario un techado prearmado o preparado de membrana continua. Se pueden usar tejas de madera o materiales en hoja para pendientes de 3:12.

- **Techos a dos aguas:** El techo se divide en dos partes iguales que se inclinan hacia abajo a partir de una cumbrera central o brocal de techo.
- **Techos a cuatro aguas:** Tienen extremos inclinados y lados que se intersecan en un ángulo inclinado que se proyecta.

Existen otras variedades en la morfología de los sistemas de techos, sin embargo, las antes mencionadas son las más sencillas y comunes. Para este trabajo solo se considerarán techos planos, techos inclinados y techos a dos aguas.

3. *Métodos de cálculo para aguas de lluvia*

La cantidad de lluvia que un techo y su sistema de drenaje deben manejar es una función de: el área de techado que conduce a los drenes o a las canaletas del techo y la frecuencia y la intensidad de precipitación de acuerdo a la región.

La precipitación se mide en milímetros de agua, o litros caídos por unidad de superficie (m^2), es decir, la altura de la lámina de agua recogida en una superficie plana es medida en mm o l/m^2. Nótese que un milímetro de agua de lluvia equivale a 1 l de agua por m^2.

La cantidad de lluvia que cae en un lugar se mide con un instrumento llamado pluviómetro y se expresa en milímetros de agua. La intensidad de precipitación puede ser entendida como la velocidad del caudal, siendo que la velocidad se expresa como $V = {}^{x}/_{t}$ podríamos decir que, la cantidad de agua que se acumularía en una superficie horizontal e impermeable A de 1 m^2 por 100mm de altura x, durante el tiempo t que dure la precipitación. En otras palabras, si tenemos una precipitación de 100 mm/hora, sería equivalente a llenar en una hora una caja de 1 m^2 por 100mm de altura.

El cálculo de la cantidad de lluvia depende de la duración y la intensidad de la precipitación pluvial y del área de drenaje. Los datos de diseño para estimar el gasto Q son la intensidad de lluvia Vexpresada en mm/hora y el área servida A en metros cuadrados:

$$Q = A \times V \qquad (4.1)$$

Para el cálculo de la velocidad del agua en canales abiertos se utiliza la fórmula de Chézy $V = C\sqrt{RS}$ en dónde S es la pendiente y C es un coeficiente que según la fórmula de Manning[18] se expresa de la siguiente manera:

$$C = \frac{1}{\eta}\sqrt[6]{R} \qquad (4.2)$$

En dónde η es un coeficiente de rugosidad, para el cual se han definido algunos estándares expresados en la Tabla 3, que la mayoría de los autores aceptan.

Material	Coeficiente de Rugosidad η
Asbesto y PVC	0,010
Metal Liso	0,011
Concreto $\phi < 55$ cms.	0,015
Concreto $\phi > 55$ cms.	0,013
Canales de concreto	0,015
Canales de tierra	0,025

Tabla 3: Coeficientes de rugosidad para distintos materiales [16].

La velocidad V expresada en m/seg sería expresada en este caso de la forma:

$$V = \frac{1}{\eta} \sqrt[3]{R} \times \sqrt[2]{S} \qquad (4.3)$$

Para evitar la acumulación de sedimentos dentro de una tubería, la velocidad mínima del agua ha de ser de 0,60 m/seg, mientras que la velocidad máxima contemplada para evitar el ruido generado por el flujo del agua dentro de las tuberías, oscila entre 3,00 y 9,5 m/seg dependiendo del material de la superficie por donde se desplaza el agua.

Así mismo, la cantidad de agua Q puede ser calculada a partir de:

$$Q = \frac{A}{\eta} \sqrt{R} \times \sqrt[2]{S} \qquad (4.4)$$

Siendo R es el radio hidráulico, expresado por:

$$R = \frac{A}{P} \qquad (4.5)$$

En donde A es el área del colector en m^2 y P es el perímetro mojado en m.

Todos estos cálculos pueden resultar bastante laboriosos por lo que el problema de selección de diámetros, pendiente y sección en el caso de canales, se aproximan mediante la utilización de tablas, que permiten además adaptar los resultados a ciertas medidas estándar que coincidan con los productos ofrecidos en el mercado, para el caso de tuberías y canaletas. En la

Tabla 4 podemos apreciar una tabla tabulada para el cálculo de tuberías, con una duración de 10 minutos, 5 años de frecuencia y una intensidad asumida de 150 mm/hora.

Diámetro del bajante pluvial		**Intensidad de Precipitación (mm/h)**					
		50	75	100	125	150	200
Centímetros	**Pulgadas**	**Áreas Máximas de proyección horizontal drenadas (m^2)**					
5,08	2	140	90	65	50	45	30
6,35	2 ½''	240	160	120	100	80	60
7,62	3''	400	270	200	160	135	100
10,16	4''	850	570	425	340	285	210
12,70	5''	1600	1070	800	640	535	400
15,24	6''	2510	1670	1250	1000	835	630
20,32	8''	5390	3590	2690	2155	1759	1350

Tabla 4: Áreas máximas de proyección horizontal en metros cuadrados que pueden ser drenadas por bajantes de aguas de lluvia de diferentes diámetros para varias intensidades de lluvia [16].

4. Sistema de Canaletas

El agua pluvial recolectada por los techos inclinados debe ser desalojada por un sistema de canaletas para evitar la erosión del suelo y el deterioro de muros y paredes de una edificación. Cuando las precipitaciones son escasas o las áreas de techos son pequeñas, se puede prescindir del sistema de canaletas utilizando voladizos adecuados y un lecho de grava o un listón de mampostería en el terreno debajo de la línea del alero [17].

El diámetro o sección de los colectores de aguas de lluvia provenientes de techos y azoteas se determina de acuerdo al área que debe ser desaguada, de su pendiente y la intensidad de lluvia registrada en la zona. Si no se conoce la intensidad de lluvia en la región se asume el valor por defecto: 100 litros por metro cuadrado (100mm/hora) [16].

El sistema de canaletas está compuesto por varias partes que pueden ser apreciadas en la Figura 8.

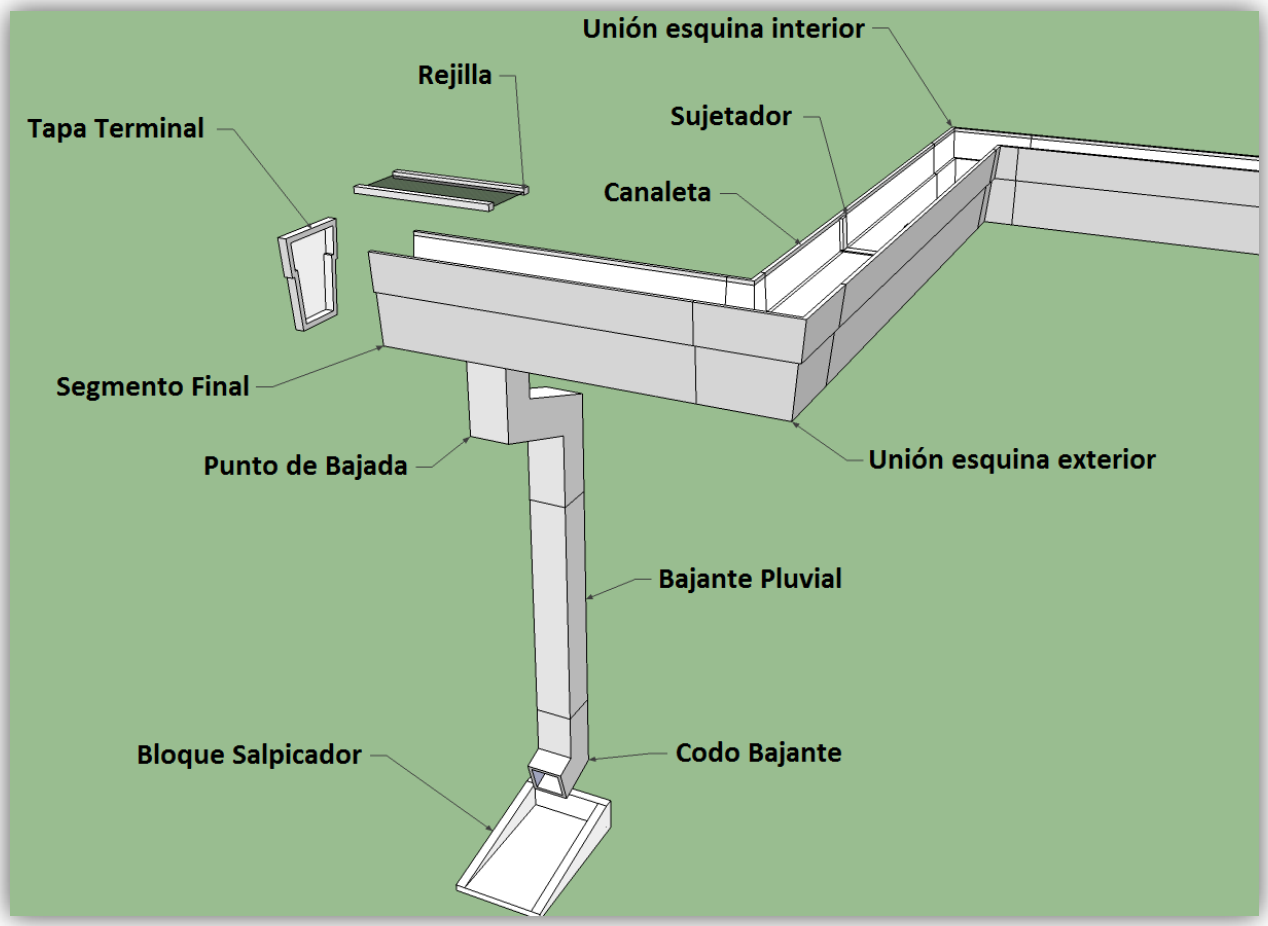


Figura 8: Diagrama de los elementos básicos de un sistema de canaletas.

- **Canaleta o canal:** Es el conducto por donde se desplaza el agua que viene del techo hasta llegar al punto de bajada. Para que el agua fluya a través de la canaleta en lugar de quedarse empozada, esta debe tener una pendiente mínima en dirección rectilínea es de $^{1}/_{16}$ '' por pie (1:200), lo que es equivalente a 0,29°. Para unir las canaletas entre sí y con el segmento final se debe traslapar y soldar o sellar las juntas con macilla. Se deben colocar juntas de dilatación para tramos de más de 12 m (40') de longitud.
- **Tapa terminal:** Se coloca al extremo de una canaleta para cerrar el paso de agua

- **Segmento final:** Sección terminal con salida. Los modelos 3d realizados para esta pieza incluyen la tapa terminal y no se utiliza la unión del bajante pluvial por razones de simplicidad.
- **Rejilla:** Es una tela de alambre que se adhiere en la parte superior de la canaleta para protegerla de las hojas. No se utiliza en el trabajo porque no se está considerando la caída de hojas o cualquier otro elemento que pueda ser arrastrado por advección junto con el agua a las canaletas.
- **Sujetadores:** Permiten colgar la canaleta del alero o de la pared. No se considera en el trabajo puesto que las canaletas se mantienen estáticos en la posición adecuada.
- **Punto de bajada:** Conduce el agua recogida por la canaleta al alcantarillado pluvial. Su área dependerá de la pendiente. En este trabajo se están considerando solo las tres pendientes indicadas en la Tabla 5.

Proporción	Equivalente en Ángulos
12:12	45°
7:12	30°
5:12	22,6°

Tabla 5: Pendientes para el punto de bajada

- **Bajante Pluvial**: Tubo de descenso vertical del agua que se adhiere al segmento final y descarga en el codo bajante.
- **Codo Bajante**: Recibe el agua drenada y la descarga a un bloque salpicador o se conecta a un sistema de drenado pluvial.

Los sistemas de tuberías plásticas adoptaron en un principio los estándares de medición de las tuberías metálicas, puesto que fueron las primeras tuberías industriales existentes. Hoy día la mayoría de las tuberías plásticas tienen dimensiones que concuerdan con estándares como IPS (*Iron Pipe Size*) o CTS (*Cooper Tube Size*), sin embargo hay dos sistemas básicos para el diseño de tuberías: *Schedule Number* (Cédula *Schedule*) o RDE (Relación Diámetro Espesor). Para el modelado del sistema de canaletas se han tomado en cuenta las dimensiones que habitualmente se encuentran en el mercado bajo los estándares antes nombrados. Como referencia se tiene la hoja técnica del manual de PAVCO [19] en el que se especifican diversas características de cada pieza que integra el sistema de canaletas.

CAPÍTULO V. Render

Para poder entender mejor lo que ocurre físicamente entre el fluido y los demás elementos existentes en la escena, es conveniente emplear técnicas que permitan visualizar al fluido lo más parecido posible al agua de lluvia, y los elementos rígidos con quienes colisiona, como todas aquellas estructuras y elementos que forman parte del sistema de drenaje y protección contra la humedad en una edificación.

En este capítulo se explican una serie de técnicas que fueron utilizadas para mejorar la visualización de la simulación. Algunas de ellas han sido realizadas previamente al despliegue, como las texturas, y otras se realizan en tiempo real utilizando varios *shaders* con diferentes técnicas visuales aplicadas a la escena.

1. Modelos 3D

En una escena se pueden encontrar diversos elementos que pueden interactuar entre sí o permanecer estáticos. Algunas librerías permiten crear primitivas geométricas (esferas, cubos, toroides, cápsulas, etc.) que formarán parte de la escena. Sin embargo, muchas veces necesitaremos crear elementos más complejos que no pueden ser representados a partir de una primitiva geométrica y deben ser creados en alguna otra aplicación para el diseño de modelos 3D.

PhysX permite la creación de algunas primitivas como *NxBoxShape* (cubos) o *NxPlaneShape* (planos) que se utilizan en este trabajo para representar techos de superficie lisa o el suelo, respectivamente.

Hay elementos un poco más complejos como la estructura de la base de la edificación que no puede ser representado con una primitiva pura sino con una primitiva modificada, para lo cual se utiliza una clase que provee PhysX: *NxConvexShape* [1].

Sin embargo hay elementos más complejos aún que necesitan ser modelados con más detalle como es el caso de los techos corrugados, los techos planos, las tejas y el sistema de canaletas en su totalidad. Para ello se utilizó el programa de modelado 3ds Max y se exportó cada modelo

3D como un archivo .obj de manera independiente. Estos modelos estaban basados en triángulos y ya incluían la configuración de normales y coordenadas de textura.

2. Texturizado

Para lograr un efecto más realista en los objetos de una escena con frecuencia se aplican a los modelos técnicas de texturizado. Esto constituye una práctica común para añadir detalles a una superficie.

El patrón de texturas puede estar dado por una matriz de valores de color o por un procedimiento que modifique los colores del objeto. En este trabajo se utilizan ambos enfoques en varios patrones de textura superficial, para lo cual se definen matrices bidimensionales trabajando con dos índices (s,t).

Como C++ no tiene soporte para el manejo de archivos de imagen, se utiliza la clase "QImage" de QT, con la cual podremos cargar imágenes con formato: bmp, gif, jpg, jpeg, png, etc. La clase "QImage" proporciona una representación de imagen independiente del hardware que permite el acceso directo a los datos de píxeles, y se puede utilizar como un dispositivo de pintura [2].

La configuración de las coordenadas de texturas para los modelos utilizados en este proyecto ha sido preestablecida desde 3ds Max. En algunos casos se utilizó un mapeado planar o cúbico, y en otros el mapeado fue hecho utilizando una técnica de mapeado por atlas de textura (*unwrap* de 3ds Max).

3. Modelo de Iluminación

Aplicar modelos de iluminación a la escena permite dar una sensación más realista y ayuda a comprender mejor la morfología de los elementos y la posición en la que están ubicados.

En este trabajo se puede alternar entre dos métodos de iluminación local, el modelo de iluminación Gouraud y Phong. También se utilizan algunas aproximaciones basadas en imágenes para la iluminación global.

La iluminación basada en imágenes es un tipo de IBR (*Image-based rendering*), que son técnicas para el despliegue basado en imágenes. Este conjunto técnicas tratan de resolver el problema del alto costo computacional que acarrea la ejecución en tiempo real de un algoritmo de visualización fotorrealista, mediante la sustitución de un objeto por alguna entidad que conserve las características visuales importantes de manera eficiente y que además simplifique las tareas de despliegue valiéndose del uso de imágenes como primitiva de despliegue. En el caso de la iluminación basada en imágenes, es precisamente el cálculo de la iluminación lo que es aproximado por medio de la utilización de imágenes.

En este trabajo se utiliza la técnica de mapeo de entorno, una aproximación para simular las reflexiones presentes en la escena, en lugar de utilizar el método clásico de *Ray Casting* con el cual se obtiene más exactitud en reflexiones pero a un alto costo computacional.

Del mismo modo se utiliza el método de oclusión ambiental para producir sombras suaves en los objetos de la escena, producto de la iluminación ambiental como una aproximación, evitando una vez más el *Ray Casting.*

a) Oclusión ambiental (Ambiental Occlusion)

La oclusión ambiental es un método de iluminación global que se utiliza para producir sombras rápidas, suaves y difusas en espacios abiertos mediante el "trazado de rayos" (*Ray Tracing*) o con alguna aproximación a través de otros métodos menos costosos. A través de esta técnica se puede añadir realismo a una escena, tomando en cuenta la atenuación de la luz en algunas partes de un modelo 3D debido a la oclusión de otros objetos o de ciertas partes del mismo [20].

Cuando se habla de oclusión en técnicas de iluminación y sombreado, se refiere a la capacidad que tienen ciertos objetos de una escena para obstruir el paso de la luz parcial o totalmente. En este sentido, es posible determinar áreas claras y oscuras en función de las demás geometrías en la escena y la proximidad de las superficies.

Esta técnica permite tener una mejor percepción de la forma que tiene cada uno de los objetos 3D desplegados en la escena. Así puede entenderse mejor la geometría de cada uno de los techos y los elementos que conforman el sistema de canaletas.

Para calcular la oclusión ambiental se emiten varios rayos a partir de un punto de luz incidental, los cuales al rebotar en la superficie de las geometrías hacen una contribución de sombreado en ese punto. Cada rayo emitido puede entenderse como un fotón que, a mayor cantidad de rebotes, mayor será la oscuridad y a menor cantidad de rebotes, más clara podrá percibirse la superficie [21].

Hay dos formas de realizar esta técnica: a través de una textura o en tiempo real. Cuando la escena tiene luz uniforme y objetos estáticos es más habitual utilizar texturas para calcular la oclusión ambiental, de lo contrario es preferible hacer el cálculo en tiempo real, en cuyo caso los tiempos de renderización serán mayores pero los resultados más exactos.

En este trabajo solo se tiene un punto de luz incidental estático y los modelos también permanecen estáticos, por lo cual se utilizan texturas donde ya ha sido calculada previamente la iluminación global con oclusión ambiental.

4. Render del Agua

Como el método SPH utilizado por PhysX para la simulación del fluido es un método basado en partículas, debe utilizarse una técnica de extracción de superficie acorde a este enfoque para dar una apariencia realista. En este caso, técnicas de extracción de superficie como *marching cubes* no son las más adecuadas, al contrario, sería favorable utilizar técnicas no basadas en poligonización. Por lo tanto, conviene usar técnicas basadas en primitivas de punto (*point rendering*).

Una buena opción sería utilizar *splatting*. Sin embargo, por la naturaleza esférica del splatting se producen artefactos que le dan una apariencia gelatinosa a las superficies, haciendo que su aspecto final sea irreal y que el fluido no se perciba como un continuo. Por esto debe utilizarse algún filtro de suavizado que aligere los cambios bruscos de curvatura entre las partículas y cree una superficie continua y lisa.

Finalmente es necesario acoplar otras técnicas que le den un mejor acabado al fluido. Para lograr que un fluido se vea real es necesario agregar ciertos efectos ópticos como la reflexión, refracción y cáustica. A continuación se describen brevemente las técnicas utilizadas en el despliegue del agua.

a) Splatting

Bagar[22] presenta un enfoque para el despliegue directo de fluidos basado en partículas utilizando splatting en lugar de despliegues poligonales y permitiendo así la reducción de los artefactos propios de los enfoques basados en mallado.

De acuerdo a Zwicker et al. [23], el despliegue de superficies utilizando splatting surgió como una alternativa de despliegue diferente a las técnicas comunes basadas en mallados de polígonos.

Con esta técnica se realiza el despliegue basado en primitivas de punto, con lo que se podrían desplegar imágenes de alta calidad de objetos geométricos a partir de un conjunto suficientemente denso de puntos que constituyan la superficie de los mismos. La idea es aproximar regiones locales de puntos en elipses planares en el espacio objeto y con la acumulación y mezcla de estas elipses crear la superficie en el espacio imagen[24].

Cada punto o *splat* es asociado con un vector normal que permitirá más adelante ser utilizado para calcular la iluminación local, entre otras cosas.

b) Adaptative Curvature Flow Filtering

El método de filtrado adaptativo para la curvatura del flujo en espacio de pantalla (*screen space curvature flow filtering*) planteado por Wladimir van der Laan et al. [25], es utilizado para disimular la geometría esférica de las partículas e impedir que el fluido tenga esa apariencia gelatinosa característica del splatting. Todo el procesamiento, despliegue y pasos de sombreado se hacen directamente en la tarjeta gráfica y el método logra un alto rendimiento en tiempo real.

Este filtro trabaja con un mapa de profundidades a nivel de *shaders*. La idea general consiste en desplazar cada valor de profundidad a partir de su vector normal en una posición de curvatura media con respecto a sus vecinos. Este proceso se repite por varias iteraciones hasta obtener resultados más refinados.

Una curvatura media es definida como la divergencia existente entre el vector normal de cada *splat* y el vector normal de la superficie del fluido. El proceso de suavizado se hace de forma iterativa, de forma tal que en cada iteración el valor de profundidad en el mapa de profundidad del agua será desplazado proporcionalmente de acuerdo al valor de curvatura media. Esto lo podemos expresar en la Ecuación 5.1.

$$H = \frac{\partial z}{\partial t} \tag{5.1}$$

donde z es el valor de profundidad, t es el número de iteración y H es el valor de la curvatura media. Para una superficie en el espacio 3D, el valor de curvatura media es definido de acuerdo a la Ecuación 5.2:

$$2H = \nabla \cdot \hat{n} \tag{5.2}$$

donde $\hat{n}$ es el vector unitario de la superficie. La normal es calculada a partir del producto cruz entre las derivadas de la posición P en el espacio, vista en la dirección (x, y), lo cual es expresado en la Ecuación 5.3:

$$\hat{n}(x,y) = \frac{n(x,y)}{|n(x,y)|} = \frac{(-C_y \frac{\partial z}{\partial x}, -C_x \frac{\partial z}{\partial y}, -C_y z)^T}{\sqrt{D}} \tag{5.3}$$

donde D, se ve expresada en la ecuación 5.4 como:

$$D = C_y^2 \left(\frac{\partial z}{\partial x}\right)^2 + C_x^2 \left(\frac{\partial z}{\partial y}\right)^2 + C_x^2 C_y^2 z^2 \tag{5.4}$$

La diferencia finita es usada para calcular la derivada espacial, C_x y C_y constituyen las coordenadas del punto de vista, mientras que x e y se refieren a la dirección. Esto se calcula para obtener las dimensiones del área de visión (*viewport*), expresadas en las Ecuaciones 5.5 y 5.6 como el FOV.

$$C_x = \frac{2}{\tan\left(\frac{FOV}{2}\right) * V_x} \tag{5.5}$$

$$C_y = \frac{2}{\tan\left(\frac{FOV}{2}\right) * V_y} \tag{5.6}$$

El vector unitario $\hat{n}$ de la Ecuación 5.3 es sustituido en la Ecuación 5.2 de forma tal que H pueda ser derivada, lo que conduce a:

$$2H = \frac{\partial \hat{n}_{\mathrm{x}}}{\partial \mathrm{x}} + \frac{\partial \hat{n}_{\mathrm{y}}}{\partial \mathrm{y}} = \frac{C_y E_x + C_x E_y}{D^{\frac{2}{3}}} \tag{5.7}$$

en la cual E_x y E_y quedan definidas en las Ecuaciones 5.8 y 5.9, respectivamente:

$$E_x = \frac{1}{2}\frac{\partial \mathrm{z}}{\partial \mathrm{x}}\frac{\partial D}{\partial \mathrm{x}} - \frac{\partial^2 \mathrm{z}}{\partial \mathrm{x}^2}\mathrm{D} \tag{5.8}$$

$$E_y = \frac{1}{2}\frac{\partial \mathrm{z}}{\partial \mathrm{y}}\frac{\partial D}{\partial \mathrm{y}} - \frac{\partial^2 \mathrm{z}}{\partial \mathrm{y}^2}\mathrm{D} \tag{5.9}$$

En resumen, la Ecuación 5.1 es una simple integración euleriana utilizada para modificar el valor de profundidad en cada iteración. La derivada espacial de z es calculada usando diferencias finitas.

La superficie puede ser discontinua por el hecho de que puedan percibirse varias siluetas de fluido en el espacio de pantalla. Para evitar unir estos parches de superficie de fluido que están realmente separados, es importante asegurarse de que las condiciones de borde sean las adecuadas. Podría definirse un umbral que descarte los grandes cambios de profundidad entre píxeles contiguos. En estos casos, así como en los que se encuentran fuera de la pantalla, al valor de la derivada espacial se le asigna el valor de cero de manera arbitraria, para impedir que se haga el suavizado en esas áreas.

El número de iteraciones elegidas dependerá del valor de suavizado final esperado. Mientras más iteraciones se hagan, la superficie quedará más suavizada pero el costo computacional será más alto.

c) SkyBox

Para mejorar la apariencia y la percepción realista de los objetos presentes en una escena, podemos aplicar diversas técnicas que alteren la forma, el relieve y la profundidad, entre otras cosas, para este fin. Aún cuando la escena sea dinámica e interactiva, se puede hacer uso de texturas, técnicas de iluminación, shaders, etc.; que estilicen el despliegue final. Sin embargo, todo esto tiene un alto costo computacional, por lo que todas estas técnicas deben ser usadas con moderación para que solo se ejecuten cuando realmente han de ser visualmente percibidas.

Cuando los elementos se encuentran muy distantes del punto de visión difícilmente se percibe algún cambio visual al desplazarse en la escena y es posible que jamás se interactúe con dichos elementos, por esta razón se puede impedir que se realicen cálculos para la mejora del despliegue de los mismos.

Así mismo, es posible crear la ilusión de tener una serie de objetos tridimensionales muy distantes del centro de interés con buenos efectos visuales pero que en realidad no existen. Esto permite dar una sensación de amplitud y grandeza a una escena, o simplemente mostrar el entorno que la rodea, en lugar de mantener un fondo unicolor que de una sensación de vacío o de abismo.

Con este propósito surge la técnica conocida como *SkyBox*, la cual no es más que encerrar la escena en un cubo lo suficientemente grande, con las normales invertidas para poder mapear dentro de él o bien seis texturas (una por cada cara), o una única textura que contenga el entorno de forma tal que pueda utilizarse un mapeado cúbico. Un *Skybox* contiene típicamente elementos que nos den la sensación de encontrarnos en un espacio abierto, tales como: el cielo, el sol, el suelo, algunas montañas o construcciones lejanas, etc. Sin embargo también puede utilizarse para ilustrar un espacio cerrado, como las paredes de una habitación, un galpón o una discoteca.

Para que un *SkyBox* luzca bien la resolución de la textura debe ser lo suficientemente grande manteniendo una proporción de un texel por píxel de pantalla [21]. La fórmula para calcular la resolución aproximada podemos expresarla a través de la Ecuación 5.10:

$$resolución\ de\ textura = \frac{resolución\ de\ pantalla}{\tan(FOV/2)} \qquad (5.10)$$

dónde el FOV representa el campo de visión. En el caso de que se utilicen seis texturas se ha de tener cuidado que las texturas estén compaginadas.

Es posible tener diferentes texturas para el *SkyBox* en el caso de que se estén utilizando diferentes escenas, como por ejemplo en los videojuegos que tienen diferentes escenarios, mapas o niveles. Para este trabajo se utilizó un *SkyBox* estático, es decir, un *SkyBox* en el que la textura siempre es la misma puesto que la escena nunca cambia.

Otro detalle importante acerca del *SkyBox* es el desplazamiento que este hace junto con la cámara en escenas con navegación. Esto se hace para evitar que el usuario atraviese el *SkyBox* al desplazarse por la escena y mantener la sensación de lejanía de los objetos estáticos simulados por el *SkyBox*.

d) Mapeo de Entorno Estático

Algunos de los efectos visuales presentes en una escena pueden realizarse mediante una técnica de mapeo de entorno (*Environment Mapping*) que permite precisamente mapear en una textura uno o más efectos visuales relacionados con el entorno que rodea al objeto.

La reflexión es uno de los efectos visuales que pueden ser aplicados con esta técnica, que en este caso se trataría específicamente de mapeo de reflexión (*Reflection Mapping*). Estos términos en ocasiones son usados indistintamente, sin embargo el mapeo de reflexión tiene un significado específico. Cuando las propiedades del material de la superficie son usadas para modificar un mapa de entorno existente, se genera una textura que es ahora un mapa de reflexión.

Se denomina Reflexión al fenómeno en el cual ondas de cualquier tipo inciden sobre una barrera plana generando nuevas ondas que se mueven en otra dirección, alejándose de la barrera [5]. Esta barrera puede entenderse como un límite entre dos medios distintos, como el aire y un vidrio, el agua, o cualquier otro tipo de superficie.

Debido a que el cálculo de la ecuación de reflectancia completa en tiempo real es muy compleja se tiende a usar aproximaciones más simples, como:

$$L_0(v) = R_F(\theta_0)L_i(r) \qquad (5.11)$$

donde R_F representa el término de Fresnel, r es el vector de reflexión, L_i el vector de incidencia y θ_i el ángulo formado entre el vector de incidencia y el vector normal de la superficie.

Si consideramos únicamente la dirección de la luz en el cálculo de la radiancia, esto puede ser almacenado en un espacio bidimensional a través de una tabla. Esto permitirá la iluminación eficiente en una superficie reflectante de cualquier forma con una distribución arbitraria de energía lumínica, con sólo calcular r en cualquier punto de la tabla de radiancia.

A esta tabla la conocemos como mapa de entorno y su uso durante el despliegue se conoce como mapeo de entorno, tal como ilustra la Figura 9. Para que L_i sólo dependa de la dirección es necesario que el punto de la luz venga de muy lejos y que el reflector no se refleje a sí mismo.

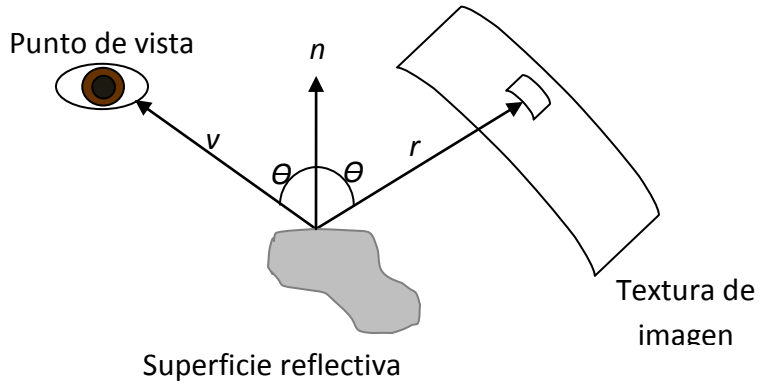


Figura 9: Función de proyección para convertir el vector de vista *(x,y,z)* reflejado en una textura *(u,v)* de la imagen creada a partir de la proyección.

Los pasos que constituyen el algoritmo para el mapeo de entorno son:

- Generar o cargar una imagen en dos dimensiones que represente el entorno de una escena.
- Para cada píxel contenido en un objeto reflectante se debe calcular la normal en la ubicación en la superficie del objeto.
- Calcular el vector de reflexión con el vector vista y el vector normal.

- Usar el vector de reflexión para calcular un índice en el mapa de entorno que representa el valor de energía lumínica o radiancia entrante en la dirección del vector de reflexión.
- Usar los datos de téxel para correlacionarlo con un valor del mapa de entorno de acuerdo al índice antes calculado.

Los mapas de entorno pueden almacenar otras propiedades además de la reflexión, por esta razón tienden a ocupar más espacio que una textura normal. A partir de técnicas como el *normal mapping* o el *bump mapping* puede ser calculado un mapa de entorno con resultados visuales muy ricos.

Cuando en una escena los elementos permanecen estáticos el cálculo de la reflexión a partir del mapeo de entorno puede hacerse una sola vez debido a que este entorno no ha de cambiar. Como los elementos de la escena en este trabajo permanecen siempre estáticos, se utiliza un mapeo de entorno estático (*Static Environment Mapping*) para el cálculo de la reflexión.

e) Espesor y coeficiente de Fresnel

Además de las técnicas antes mencionadas es conveniente configurar algunos otros parámetros de visualización que mejoren la apariencia final del fluido. Este es el caso del espesor (*thickness*) y el coeficiente de Fresnel.

El espesor se utiliza para calcular correctamente los atributos visuales como la atenuación del color, la transparencia y la refracción. Para ello las partículas del fluido son consideradas como esferas de tamaño constante en el espacio y se despliegan de manera similar a como se hace en el método de filtrado adaptativo para la curvatura del flujo, la diferencia es que en lugar de almacenar valores de profundidad se guardan los valores de espesor en el píxel proyectado [22]. El espesor del fluido será calculado de acuerdo a lo expresado en la Ecuación 5.12:

$$T(x,y) = \sum_{i=0}^{n} d\left(\frac{x - x_i}{\sigma_i}, \frac{y - y_i}{\sigma_i}\right) \qquad (5.12)$$

donde d representa la profundidad de la función kernel, x_i e y_i son las componentes de la posición de proyección de la partícula, x e y las coordenadas de pantalla y finalmente σ_i es el tamaño de proyección. Esta sumatoria utiliza mezclado aditivo para ir escribiendo el color del espesor del fluido en lugar de la profundidad, y las partículas son desplegadas con el test de

profundidad habilitado y la escritura de profundidades deshabilitada para asegurar una correcta visibilidad de la geometría presente detrás del fluido.

El coeficiente de Fresnel indica qué cantidad de luz se refracta y cuánto se refleja a partir de las conocidas ecuaciones de Fresnel, las cuales describen el comportamiento de la luz al atravesar dos medios con diferentes índices de refracción. Estas ecuaciones constituyen un conjunto de relaciones matemáticas entre la amplitud de la onda reflejada, la onda refractada y la onda incidente. Benjamin Peters y Martin Schäf [26] expresan el coeficiente de Fresnel con la Ecuación 5.13 y obtienen un valor entre 0 y 1:

$$F = \frac{1}{2}\left(\frac{\sin^2(\theta_i - \theta_t)}{\sin^2(\theta_i + \theta_t)}\right) + \tan^2(\theta_i - \theta_t) * \tan^2(\theta_i + \theta_t) \qquad (5.13)$$

donde θ_i es el ángulo del rayo incidente y θ_i el ángulo del rayo refractado, ambos con respecto a la normal. También se puede obtener el valor de contribución del coeficiente de Fresnel en el reflejo especular de la luz con la Ecuación 5.14, conocida como aproximación de Schlick:

$$F(\theta) = F_0 + \big((1 - \cos\theta)(1 - F_0)\big)^5 \qquad (5.14)$$

donde θ constituye la mitad del ángulo entre la dirección de luz entrante y saliente y F_0 es la cantidad de reflexión generada, lo cual expresamos de acuerdo a la Ecuación 5.15:

$$F_0 = \frac{(r-1)^2}{(r+1)^2} \qquad (5.15)$$

en la cual r constituye el índice de refracción del medio. Finalmente, las propiedades ópticas del fluido estarán basadas en la ecuación de Fresnel y el valor especular de Phong, lo cual se indica en la Ecuación 5.16 [22]:

$$C = a\big(1 - F(\vec{n} \cdot \vec{v})\big) + bF(\vec{n} \cdot \vec{v}) + k_s\big(\vec{n} \cdot \vec{h}\big)^{\delta} \qquad (5.16)$$

donde a es el color del fluido, que incluye además el color de fondo refractado, b es el color de reflexión obtenido por medio del mapa de entorno, k_s y δ son constantes para el valor especular, F es el coeficiente de Fresnel, $\vec{n}$ es la normal de la superficie en el espacio de pantalla, $\vec{h}$ la mitad del ángulo entre la cámara y la luz y finalmente $\vec{v}$ es el vector director de la cámara.

CAPÍTULO VI. Diseño e Implementación

En este capítulo se explican detenidamente cada uno de los detalles de implementación a considerar. Las especificaciones con respecto al tipo de techo, estructura, dimensiones, fluido, simulación, técnicas y efectos visuales implementados, etc. Así como las clases que conforman la aplicación y como están relacionadas entre sí, los elementos que conforman la interfaz gráfica y la navegación de la escena utilizando mouse y teclado.

1. Detalles de implementación

A continuación se explican en detalle cada uno de los aspectos considerados en la simulación y en la creación de la escena. Entre ellos, todo lo referente a los detalles arquitectónicos de implementación, simulación de aguas de lluvias utilizando PhysX y los efectos visuales que le dan una apariencia más realista al fluido.

Las dimensiones del techo están disponibles en un rango limitado. Se diseñó una interfaz que permita la configuración del largo y ancho del techo, desde el tamaño más pequeño 6m, incrementando en 2m hasta el tamaño más grande de 20m. En el caso de los techos de tejas, el ancho y largo más grande es de 16m por la complejidad del mallado.

Dado que las dimensiones del techo son conocidas y que el cálculo de tuberías puede ser discretizado en tablas (ver Tabla 4) es posible tabular las dimensiones del sistema de canaletas de acuerdo a las especificaciones del proyecto. En este caso, al conocer todos los tamaños configurables de techo, se pueden calcular de antemano todas las áreas de techo, y obtener a partir de allí el diámetro del bajante pluvial.

A continuación se muestra la Tabla 6, en función de la intensidad de precipitación y algunos valores de área máxima de proyección horizontal existentes en la Tabla 4, acordes al rango de áreas posibles en este trabajo.

Tanto los modelos de techos como todas las partes que conforman el sistema de canaletas, han sido hechos, morfológicamente a escalas reales y de proporciones idénticas entre sí. Lo único que no se ha mantenido a escala real son los tamaños en los que generalmente son

comercializados, esto para reducir la cantidad de modelos y simplificar el algoritmo por el cual cada pieza habría de encajar en un edificio, es decir, armar la estructura final basada en todas las variantes elegidas por el usuario.

Áreas de proyección horizontal drenadas (m^2)	Intensidad de Precipitación (mm/h)					
	200	150	125	100	75	50
45	6,35	5,08	5,08	5,08	5,08	5,08
50	6,35	6,35	5,08	5,08	5,08	5,08
60	6,35	6,35	6,35	5,08	5,08	5,08
65	7,62	6,35	6,35	5,08	5,08	5,08
80	7,62	6,35	6,35	6,35	5,08	5,08
90	7,62	7,62	6,35	6,35	5,08	5,08
100	7,62	7,62	6,35	6,35	6,35	5,08
120	10,16	7,62	7,62	6,35	6,35	5,08
135	10,16	7,62	7,62	7,62	6,35	5,08
140	10,16	10,16	7,62	7,62	6,35	5,08
160	10,16	10,16	7,62	7,62	6,35	6,35
200	10,16	10,16	10,16	7,62	7,62	6,35
210	10,16	10,16	10,16	10,16	7,62	6,35
240	12,70	10,16	10,16	10,16	7,62	6,35
270	12,70	10,16	10,16	10,16	7,62	7,62
285	12,70	10,16	10,16	10,16	10,16	7,62
340	12,70	12,70	10,16	10,16	10,16	7,62
400	12,70	12,70	12,70	10,16	10,16	7,62

Tabla 6: Diámetro del bajante pluvial (cm) de acuerdo a las áreas de proyección horizontal generadas y a diferentes valores de intensidad de lluvia.

Por esta razón las canaletas siempre tienen una tapa terminal, bien sean segmentos finales o simplemente canaletas. Ambas partes se trasladarán de acuerdo al tamaño del techo y la canaleta se escalará a lo largo para completar al segmento final, que no puede ser escalado para preservar su morfología.

Del mismo modo, las techumbres corrugadas no tendrán el tamaño convencional de las láminas en que son vendidas (los largos estándar son de 1,83m; 2,44m; 3,05m y 3,66m; los anchos estándar oscilan entre los 0,8m y 0,9m), sino el ancho y largo elegido por el usuario, correspondiente al tamaño final del techo. Igualmente, los techos de tejas corresponderían a

un mallado de cómo se vería un tejado completo, para evitar armar un tejado cargando varios modelos de tejas de 255×485 mm, necesarias para revestir un techo de 36m^2, por ejemplo.

Es necesario destacar que ninguna de estas dos diferencias altera la morfología real de los modelos, puesto que solo se estaría generando un mallado de lo que sería colocar perfectamente varias láminas de techumbre corrugada sobre un techo, hacer un tejado o colocar varias canaletas hasta completar la longitud necesaria.

Los modelos son instanciados en las coordenadas (0,0,0). Armar la estructura es posible haciendo una serie de escalamientos, traslaciones y rotaciones, dependientes de las características elegidas por el usuario y de los modelos en sí mismos.

a) Transformaciones afines a los techos

Los techos son cargados de antemano en el tamaño elegido previamente por el usuario, por lo cual no es necesaria una operación de escalamiento, que podría deformar el modelo.

En el caso de los techos inclinados y techos a dos aguas, siempre se hará una rotación. Los techos planos, tienen también una ligera pendiente, pero al funcionar su sistema de drenaje de una manera distinta a como lo hacen los antes mencionados, esa pendiente viene incluida ya en el mallado. El ángulo de rotación es determinado por la pendiente elegida por el usuario a través de la interfaz (véase la Tabla 3).

En el caso de los techos a dos aguas, se cargan dos techos del mismo tipo y las mismas dimensiones. En este caso, la rotación se hace en uno de los techos en α y en el otro en $-\alpha$. Todas estas rotaciones se hacen en el eje x.

Ambos techos han de ser desplazados la mitad de su longitud en el eje z, de esta forma el brocal del techo se extenderá exactamente sobre el eje x. Por el contrario los techos planos e inclinados se mantienen en la coordenada en que han sido instanciados (0,0,0), por lo cual no es necesario hacer traslación alguna.

b) Transformaciones afines al sistema de canaletas

La pendiente mínima en dirección rectilínea es de $\alpha = 0{,}29°$, lo equivalente a una relación por pie (1:200). Este ángulo de rotación aplica tanto para las canaletas como para los segmentos finales, sin embargo, la diferencia de longitud no permite que coincidan en altura, como se aprecia en la Figura 10, por lo cual es necesario desplazar un poco hacia abajo el segmento final.

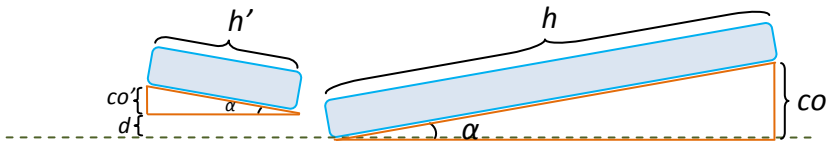


Figura 10: Rotación de las canaletas en un ángulo de pendiente mínima.

El cálculo del desplazamiento del segmento final sobre el eje y para corregir el defecto producto de la rotación, se hace en base a la Ecuación 6.1:

$$d = co - co' = sin(\alpha) * (h - h') \qquad (6.1)$$

donde $h = 3m.$ corresponde a la longitud de la canaleta mientras que $h' = 0{,}3m.$ corresponde a la longitud del segmento final. Los segmentos derechos rotarán en α y los izquierdos en $-\alpha$.

Las canaletas han de tener un diámetro específico de acuerdo a la cantidad de agua servida que deben drenar. Esto dependerá del área del techo y la intensidad de precipitación, valores que serán indicados por el usuario a través de la interfaz, y de los cuales ya se conoce el diámetro de canaleta correspondiente (véase la Tabla 6). Para ello se utiliza una "matriz esparcida", en donde las diferentes áreas de techo posible están indicadas por las filas y las intensidades máximas de precipitación consideradas son representadas por las columnas.

Como la canaleta ya tiene un valor ancho-alto que viene con el modelo, el escalamiento no será exactamente el indicado por la Tabla 6. Para cada canaleta, se ha calculado previamente el valor de escalamiento que ha de tener de acuerdo a su dimensión original (dimensión del modelo) y a las cinco dimensiones estándar (diámetros de canaletas ofrecidas en el mercado) que están siendo consideradas de acuerdo a las áreas de techos posibles de obtener.

Estos valores de escalamiento están almacenados en una matriz de flotantes. A su vez, la intersección fila-columna en la matriz esparcida indica el índice que en la matriz de flotantes contiene la dimensión de escalamiento de la canaleta correspondiente a dicha combinación. Este escalamiento será aplicado a los ejes y y z de las canaletas y los segmentos finales. Al punto de bajada y el bajante pluvial, se le escalará en los ejes x y z.

Los segmentos finales no se escalan en el eje x para evitar deformar el modelo y que pierda la correspondencia con las demás piezas que conforman el sistema de canaletas (de lo contrario perdería la dimensión estándar del orificio que ha de encajar con el punto de bajada para drenar el agua). Sólo se escalará la canaleta para cubrir la longitud total del alero, correspondiente al ancho del techo. Este escalamiento se calcula a partir de la Ecuación 6.2:

$$\frac{anchoTecho \; - \; longitudSegmentoFinal}{longitudCanaleta} \qquad (6.2)$$

donde la longitud del segmento final y la longitud de la canaleta son constantes con valor 0.3mts y 3mts, respectivamente.

Cuando el segmento de desagüe esté en el centro del alero, el escalamiento de las canaletas se calculará de acuerdo a la Ecuación 6.3:

$$\frac{(anchoTecho/2) \; - \; (longitudSegmentoFinal/2) - \; longitudCanaleta}{longitudCanaleta} \qquad (6.3)$$

Con respecto a la traslación, todos los elementos que constituyen el sistema de canaletas deben ubicarse a lo largo de los aleros. El sistema de canaletas sólo estará presente en los

techos inclinados o a dos aguas, por esta razón se conoce ya que el techo siempre ha de estar rotado.

Para determinar el desplazamiento en z y en y, es necesario deducir una ecuación a partir de una vista transversal de la estructura. La Figura 11 representa el techo inclinado visto lateralmente, en donde P representa al techo, L es su longitud y α el ángulo de inclinación elegido por el usuario.

De esta manera, desplazaremos $\frac{L}{2}\cos\alpha + \frac{d}{2}$ la(s) canaleta(s) y al segmento final en el eje z, donde d es el diámetro de la canaleta. En el eje y el desplazamiento será de $\frac{L}{2}\sin\alpha + \frac{h}{2}$, donde h es la altura de la canaleta.

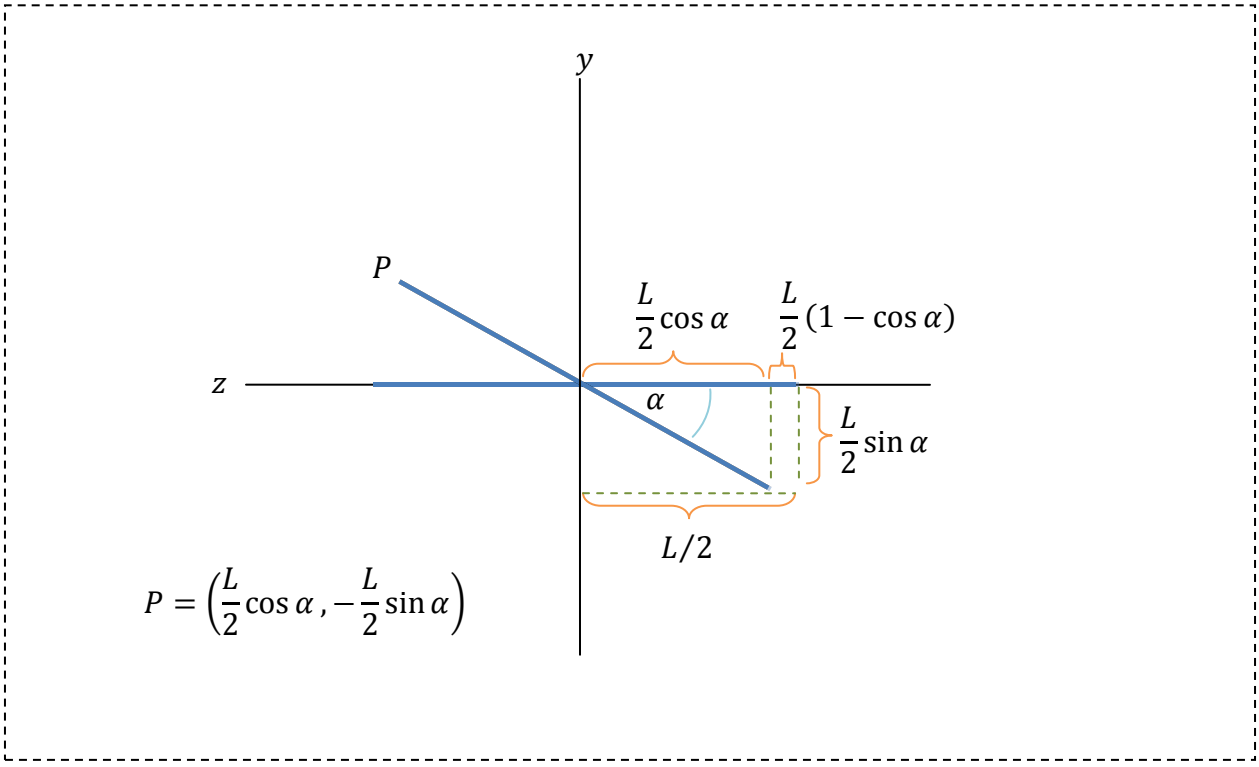


Figura 11: Vista lateral de un techo inclinado

El desplazamiento en x es opuesto en los elementos izquierdos y derechos. Es decir, para las piezas izquierdas, el desplazamiento será de $l/2 - a/2$, mientras que para las derechas el desplazamiento será de $a/2 - l/2$, donde a es el ancho del techo y l la longitud de la canaleta.

Cuando el segmento de desagüe está en el centro del alero, es necesario colocar canaletas tanto del lado derecho como del lado izquierdo. En este caso el desplazamiento será el antes descrito para ambas canaletas. Sin embargo, el segmento de desagüe no se desplazará en x.

En el caso del punto de bajada, el desplazamiento en z será de $\frac{L}{2}\cos\alpha + c$, en donde c es una constante con valor 0,009m que representa la distancia entre el borde de la canaleta y el orificio de desagüe. En el eje y el desplazamiento será de $-\frac{L}{2}\sin\alpha - h$.

El bajante pluvial será desplazado $h_{canaleta} - h_{punto\ de\ bajada} - \frac{L}{2}\sin\alpha - s$ en el eje y, donde s es una constante de valor 0,375m, que representa la mitad de la dimensión del bajante pluvial. En el eje z el desplazamiento será de $\frac{L}{2}\cos\alpha + c - e/2 + b$, donde e es el ancho del punto de bajada y b el diámetro o el ancho del bajante pluvial, que corresponde a 0,21m.

Para ambos, el desplazamiento en x será el mismo de las canaletas, restando la distancia existente entre el borde del segmento de desagüe y el orificio del mismo, una constante de 0,295m.

c) Estructura del Edificio

Al construir una edificación de cualquier tipo es importante la existencia del voladizo del techo para evitar la erosión del suelo y el deterioro de las paredes. En este trabajo, tanto las dimensiones de la estructura como del voladizo están determinadas por el tamaño del techo y el sistema de canaletas.

Para determinar las dimensiones y la posición de la estructura del edificio es necesario tomar varios factores en cuenta, como el ángulo de inclinación del punto de bajada y las dimensiones del techo y su inclinación.

Una de las razones por las que el punto de bajada necesita un codo inclinado, es para que el bajante pluvial pueda aferrarse a la pared. Dependiendo de la distancia que hay entre el techo y las paredes se elige la longitud e inclinación de dicho codo. Para este trabajo se utilizan solo 6 modelos predeterminados, 3 circulares y 3 rectangulares, en ambos casos se cuenta con 3

ángulos de inclinación: 22,6°, 30° y 45° cuyas longitudes son también una constante. Por esta razón, la distancia que ha de haber entre el alero y el techo se calcula en función del punto de bajada elegido por el usuario

Como la estructura esta siempre centrada en el origen, lo que se hace es restar al largo del techo la distancia que ha de haber entre el borde del techo inclinado (previamente calculada para ubicar las canaletas) y el ancho del punto de bajada. Esa dimensión será exactamente la mitad de la estructura, en los casos de techo inclinado, tanto en ancho como en largo. Para los techos a dos aguas, la dimensión a lo largo se duplicará.

Para crear la estructura se utiliza un *convexMesh* de PhysX de ocho vértices. El convexMesh permite modificar una primitiva, en este caso se editan los valores de los vértices de un box, una geometría básica de PhysX. La idea es modificar la cara superior del box de forma tal que se ajuste a la inclinación del techo, para ello se le resta una constante a dos de sus vértices superiores y a los otros dos se les suma la misma constante como se indica en la Figura 12.

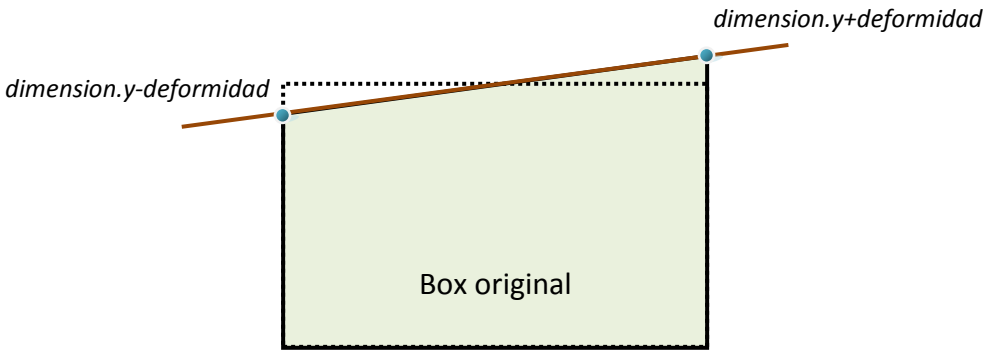


Figura 12: Vista lateral de la estructura modificada a partir de un *convexMesh*.

De esta forma se tienen dos variables: la dimensión del box y la constante de deformidad. Para calcular la dimensión se resta al tamaño del techo el largo del punto de bajada con la Ecuación 6.4:

$$\Delta zPB - escala * e/2 \qquad (6.4)$$

donde ΔzPB representa el desplazamiento en el eje z del punto de bajada y e corresponde al ancho del punto de bajada, que al multiplicar por la escala de diámetro correspondiente de acuerdo al cálculo de tuberías se obtiene el tamaño real del mismo. Por estar todos los modelos ubicados por defecto en el origen, es necesario desplazar siempre la mitad de su dimensión en cada uno de sus posicionamientos. Este desplazamiento inicial del punto de bajada ya había sido incluido en ΔzPB, por lo que para anularlo se divide entre dos la constante e. Este cálculo define la dimensión en los ejes x y z, el valor para el eje y es una constante de 3mts.

Ahora es necesario deformar el box para que la cara superior del mismo se adecue a la inclinación del techo. Para calcular el valor de deformidad se hace uso de los criterios de semejanza de triángulos rectángulos [27]. En la Figura 13 vemos como, en una vista transversal, el plano sobre el que se extiende el techo, el plano de la cara superior del box (sobre el eje z) y el plano que pasa por el eje y forman un triángulo rectángulo semejante al triángulo conformado por la distancia que había entre la mitad del techo inclinado hasta el alero. Estas distancias ya son conocidas (posZ,posY), puesto que han sido previamente calculadas a partir del ángulo α y la longitud del techo, para aplicar sobre el sistema de canaletas las transformaciones afines antes mencionadas.

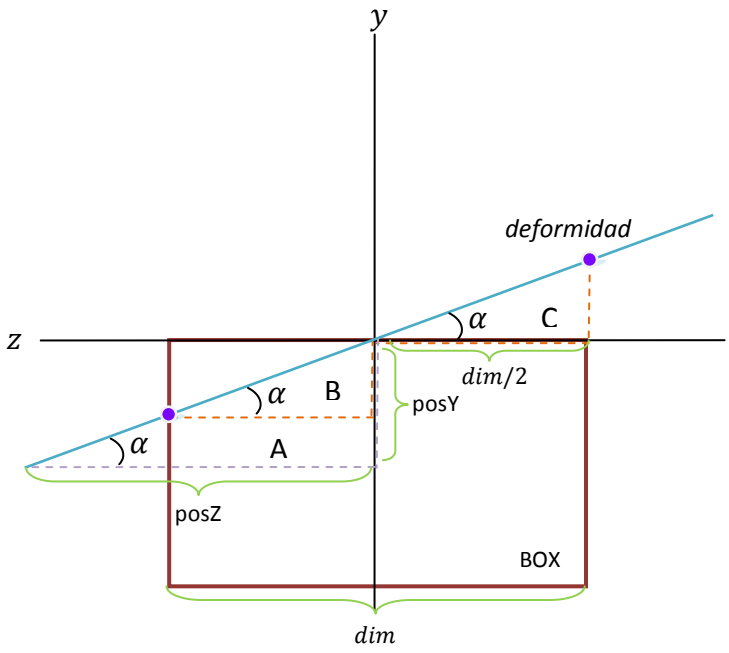


Figura 13: Principio de triángulos semejantes para deducir el valor de deformidad.

Por el principio de triángulos semejantes [27], podemos afirmar que el ángulo α es el mismo para los triángulos A, B y C. Además conocemos la dimensión del box, por lo que podemos calcular el valor de deformidad a través de la Ecuación 6.5:

$$dim/2 * \tan \alpha \qquad (6.5)$$

Para crear la estructura de los techos a dos aguas se utiliza un convexMesh de diez vértices. En este caso los vértices superiores serán seis y los inferiores solo cuatro (los cuatro que constituyen la base de la estructura). A los cuatro vértices superiores que se encuentran en los extremos (los más alejados y los más cercanos en el eje z) se les restará el valor de deformidad, mientras que a los dos vértices superiores centrales (ubicados en cero en el eje z) se les sumará dicho valor de deformidad. En la Figura 14 se pueden apreciar las transformaciones aplicadas sobre los vértices superiores del convexMesh.

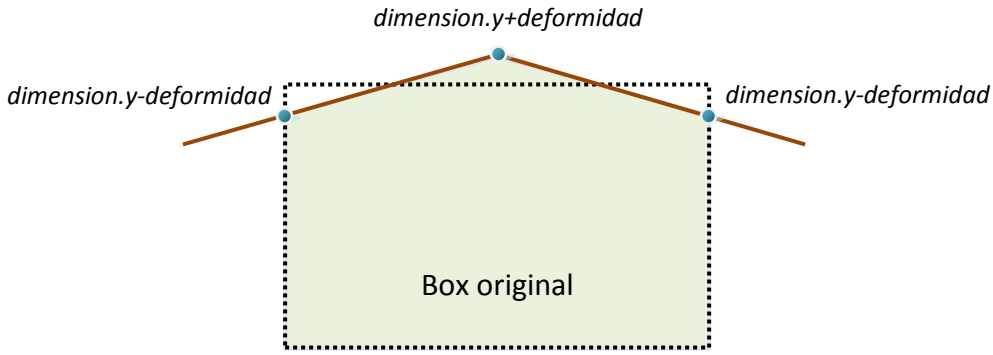


Figura 14: Vista lateral de la estructura para techo a dos aguas modificada a partir de un *convexMesh*.

El cálculo de la deformidad se hace exactamente igual al caso del techo inclinado, con la diferencia que el valor del cateto opuesto ya no será $dim/2$ sino dim puesto que la dimensión en este caso se duplica.

A nivel de código, lo que hacemos es crear los vértices de la estructura con estas modificaciones ya incluidas para ser encolados posteriormente en un vector de vértices. Se crearán ocho o diez vértices, según sea el caso, de acuerdo a lo anteriormente explicado. En la Figura 15 podemos observar el código referente a la configuración de los vértices para la creación de la estructura.

```
// Estructura del edificio
if(morfologia==2){
	verts.pushBack(NxVec3(dimension.x,  -dimension.y,   dimension.z));
	verts.pushBack(NxVec3(-dimension.x, -dimension.y,   dimension.z));
	verts.pushBack(NxVec3(dimension.x,  -dimension.y,  -dimension.z));
	verts.pushBack(NxVec3(-dimension.x, -dimension.y,  -dimension.z));
	verts.pushBack(NxVec3(dimension.x,  dimension.y+deformidad,  0.0f));
	verts.pushBack(NxVec3(-dimension.x, dimension.y+deformidad,  0.0f));
	verts.pushBack(NxVec3(dimension.x,  dimension.y-deformidad,  -dimension.z));
	verts.pushBack(NxVec3(-dimension.x, dimension.y-deformidad,  -dimension.z));
	verts.pushBack(NxVec3(dimension.x,  dimension.y-deformidad,  dimension.z));
	verts.pushBack(NxVec3(-dimension.x, dimension.y-deformidad,  dimension.z));
}else{
	verts.pushBack(NxVec3(dimension.x,  -dimension.y,   dimension.z));
	verts.pushBack(NxVec3(-dimension.x, -dimension.y,   dimension.z));
	verts.pushBack(NxVec3(dimension.x,  -dimension.y,  -dimension.z));
	verts.pushBack(NxVec3(-dimension.x, -dimension.y,  -dimension.z));
	verts.pushBack(NxVec3(dimension.x,  dimension.y+deformidad, -dimension.z));
	verts.pushBack(NxVec3(-dimension.x, dimension.y+deformidad, -dimension.z));
	verts.pushBack(NxVec3(dimension.x,  dimension.y-deformidad,  dimension.z));
	verts.pushBack(NxVec3(-dimension.x, dimension.y-deformidad,  dimension.z));
}
```

Figura 15: Creación de los vértices de la estructura a nivel de código.

2. Clases de la aplicación

A continuación se explicarán en detalle cada una de las clases que conforman la aplicación y como están relacionadas entre sí.

La aplicación contiene una serie de clases que controlan la interfaz, la creación y manipulación de elementos de PhysX, la navegación, manejo de la escena, control del programa, y la utilización de shaders que mejoran el render final de la escena. En la Figura 16 podemos ver el diagrama de clases de dicha aplicación.

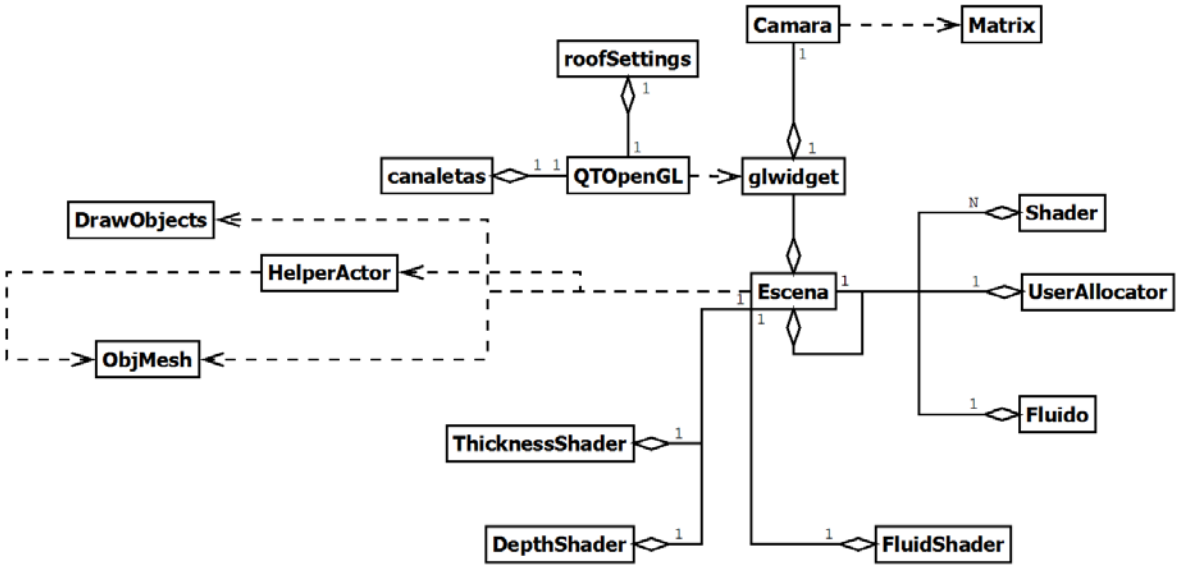


Figura 16: Diagrama de clases de la aplicación.

Las clases que controlan la interfaz gráfica y el control de la aplicación son:

a) Camara

Controla la cámara de la aplicación y captura los eventos de teclado y mouse para la navegación en la escena. Es posible desplazarse hacia adelante o hacia atrás con las flechas de arriba y abajo en el teclado, así mismo es posible desplazarse hacia los lados con las flechas de izquierda y derecha. Con el mouse es posible cambiar el punto de vista de la cámara.

La cámara está hecha con OpenGL y la captura de eventos se hace con las clases QEvent de QT. En la Figura 17 se pueden ver en detalle los elementos que conforman esta clase.

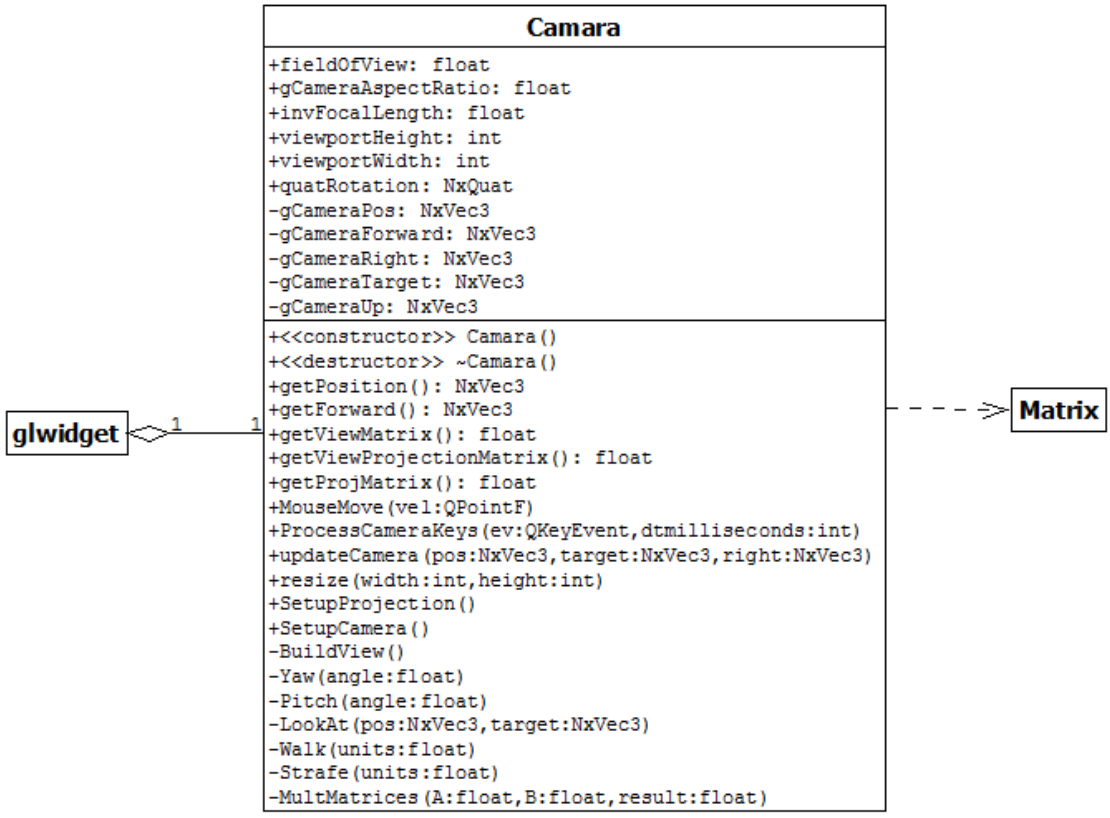


Figura 17: Diagrama de clases de la clase Camara.

b) canaletas

Controla la ventana “Insertar canaleta”, ver Figura 19. Permite la selección de un único tipo de canaleta (Biselada, Semicircular, Estilo K y Pecho Paloma) por medio de botones. A través del comboBox “Puntera de desagüe” se puede elegir hacia qué lado va a quedar el segmento final (derecha, izquierda o central). Se puede elegir el ángulo de inclinación del punto de bajada (45°, 30° o 22,6°).

La clase hereda de “QMainWindow” quien gestiona la captura de eventos de interfaz. Es instanciado por la clase “QTOpenGL” como se aprecia en la Figura 18.

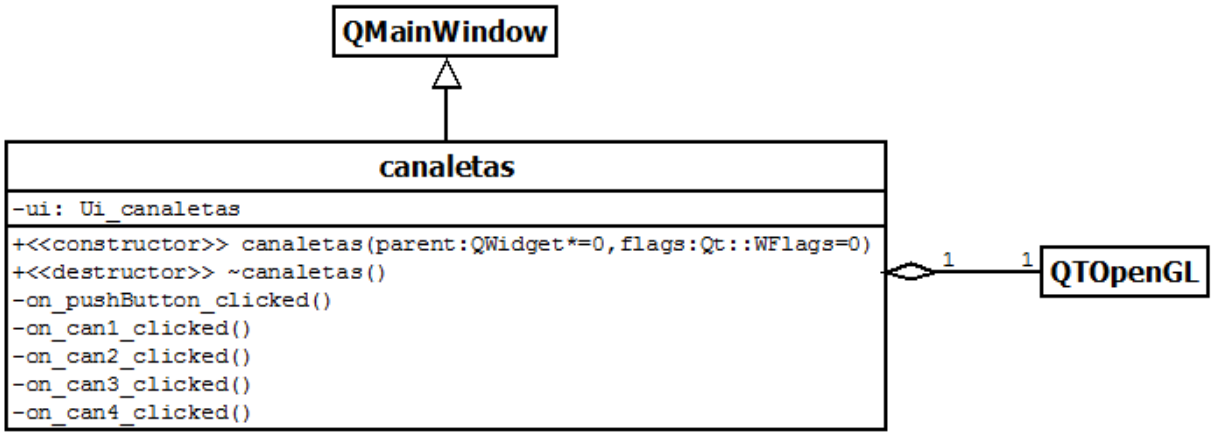


Figura 18: Diagrama de clases de la clase canaletas.

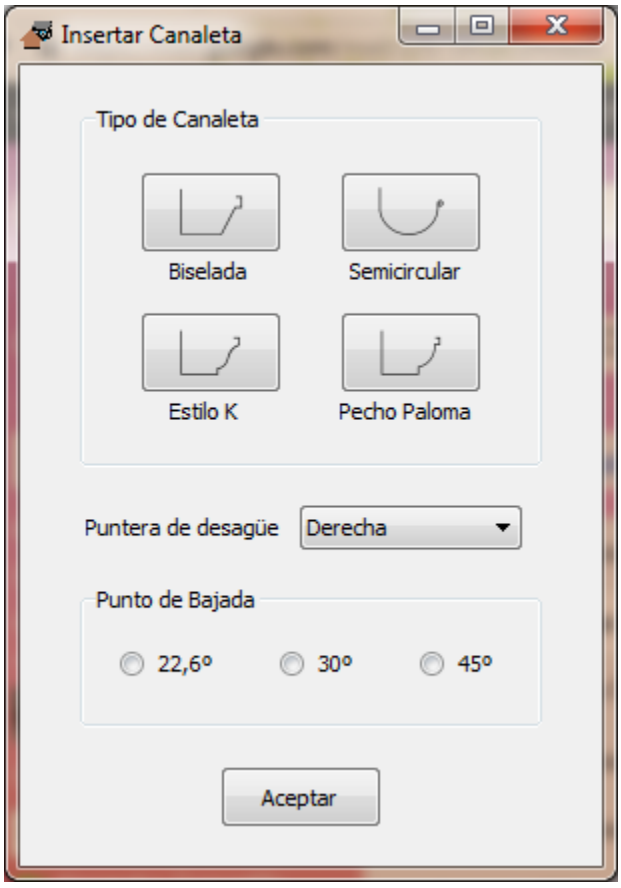


Figura 19: Ventana de interfaz para configurar el sistema de canaletas.

c) glwidget

Establece todas las configuraciones básicas de OpenGL. Esta clase hereda de “QGLWidget” una clase de la librería QT que permite crear un lienzo de OpenGL. Además es en esta clase en donde se crea una instancia del singleton “Escena” y se crea la cámara (instancia de la clase Camara) , como se aprecia en la Figura 20.

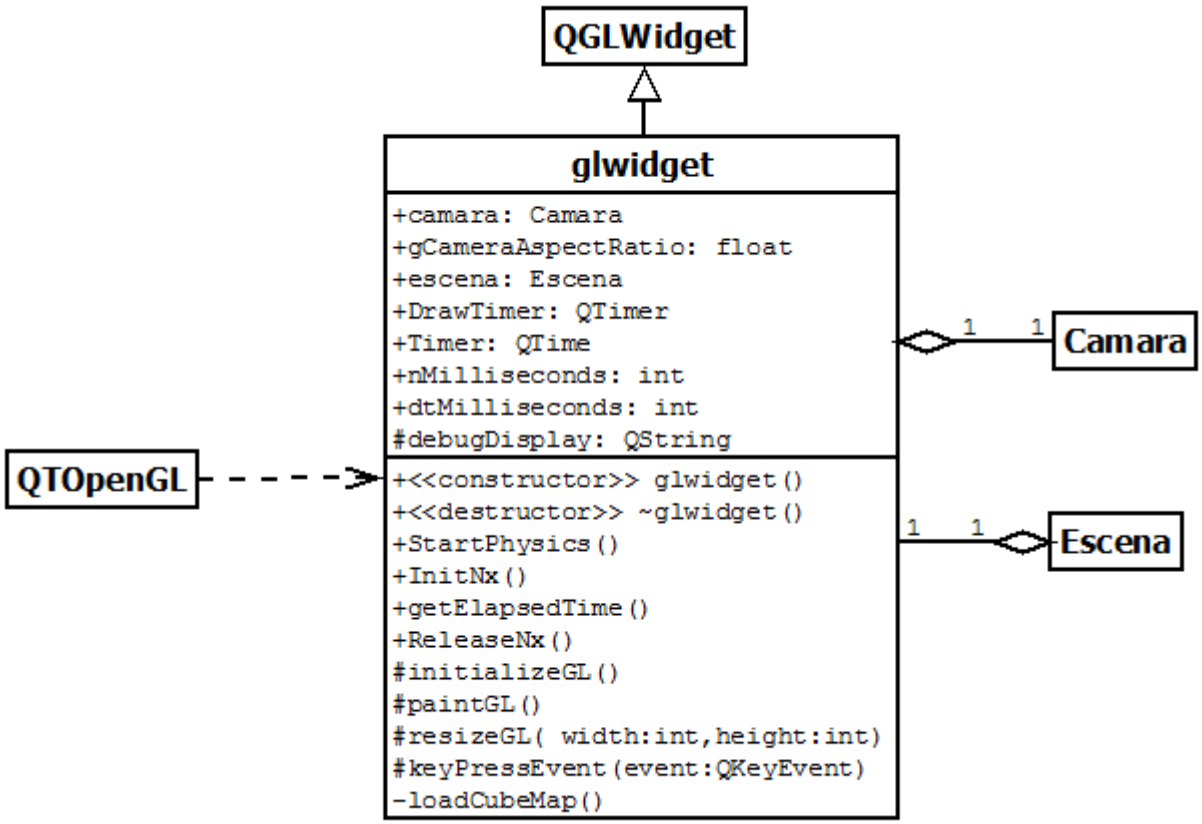


Figura 20: Diagrama de clases de la clase glwidget.

d) Librerias

Incluye todas las librerías de PhysX y los Helpers de Nvidia que han de ser utilizados. Es un archivo únicamente de cabecera que se incluye en todas las clases de la aplicación.

e) qtopengl

Es una clase de control que permite capturar todos los eventos de la interfaz gráfica y a partir de ello ejecutar acciones o darle valor a algunos atributos de control. Esta clase hereda de "QMainWindow" que es la clase principal en la librería de QT para el manejo de aplicaciones de ventana, tal como se aprecia en la Figura 21.

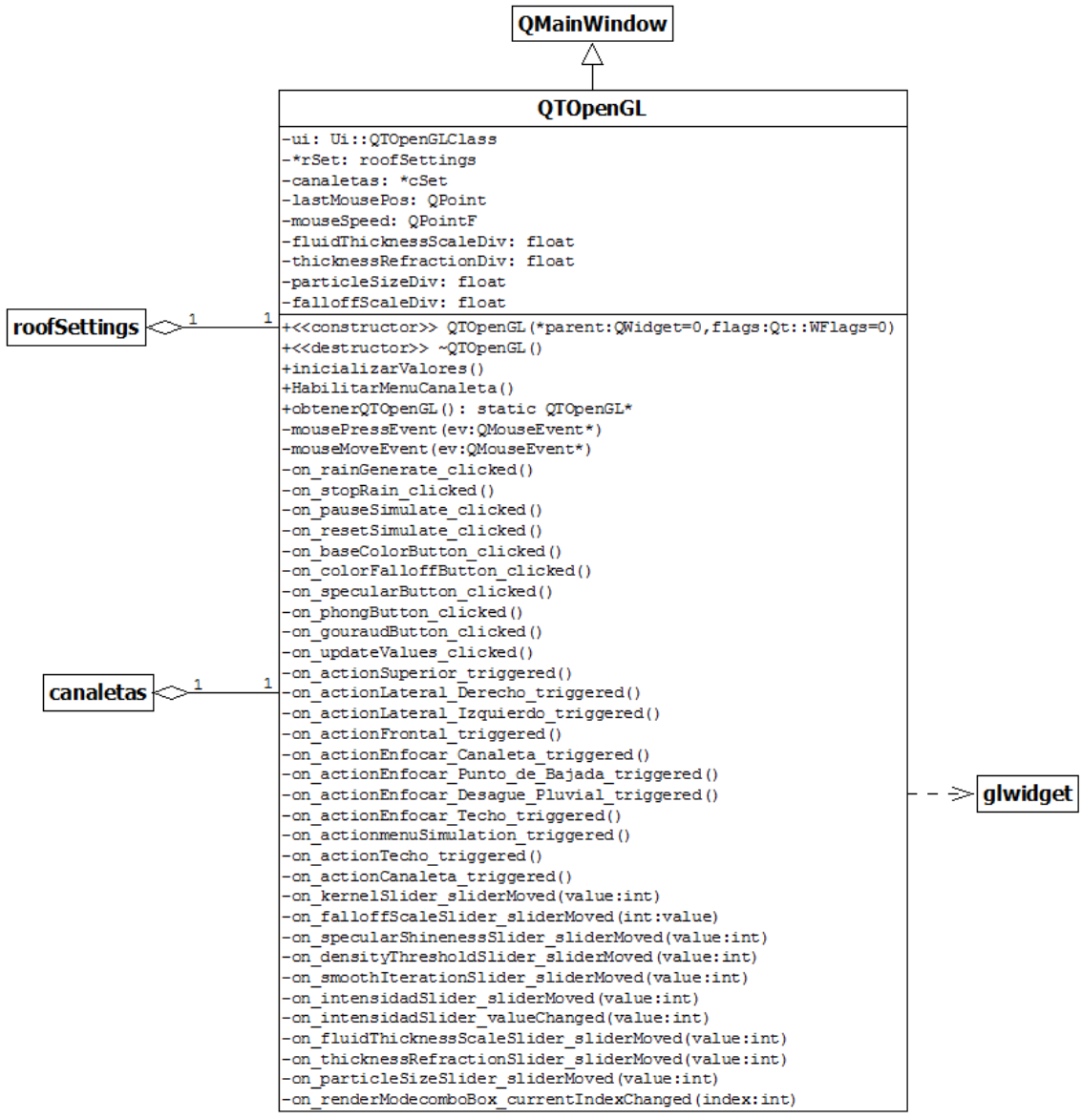


Figura 21: Diagrama de clases de la clase QTOpenGL.

La Figura 22 muestra una captura de la aplicación en tiempo de ejecución. Allí pueden apreciarse los elementos que forman parte de la interfaz principal: el menú “Simulación”, desde donde se crean los techos y el sistema de canaletas; el menú “Vistas” que permite mover la cámara a posiciones predeterminadas. Los botones que controlan la iluminación con el modelo de sombreado Phong o Gouraud, según sea el caso. Una serie de *sliders* y botones para la edición de todos los parámetros que intervienen en el proceso de despliegue y visualización del agua, así como la posibilidad de elegir el tipo de despliegue que se hará (*ComboBox* “Modo de Despliegue”) y de reiniciar los parámetros a su valor por defecto. También hay un slide que nos permite editar la intensidad de precipitación y finalmente se encuentran unos botones que inician el proceso de generación de la lluvia o lo detiene. Se puede pausar la simulación o simplemente reiniciarla.

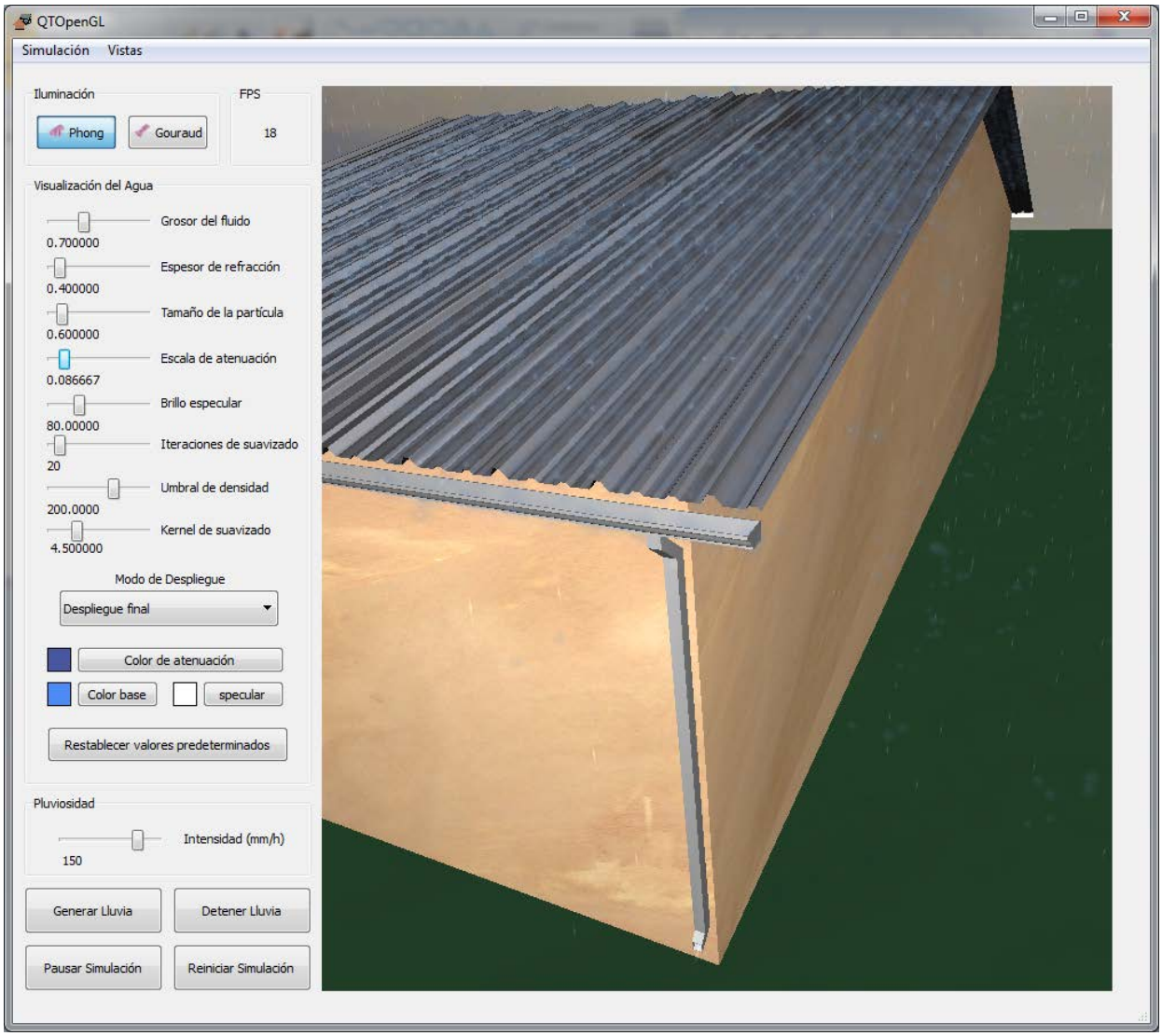


Figura 22: Captura de la aplicación con la interfaz principal.

f) roofSettings

Controla la interfaz de la ventana “Insertar Techo”, ver Figura 24. Desde allí es posible configurar el largo y ancho del techo, expresado en metros; la morfología del techo (techo plano, techo inclinado o techo a dos aguas) y el tipo de techo (superficie lisa, revestimiento de tejas o techo corrugado). Si en tipo de techo se elige la opción de techo corrugado, se habilitarán cuatro botones que permitirán seleccionar el tipo de techo corrugado. Finalmente es posible elegir la proporción *(alto:ancho)* que determina la inclinación del techo si se ha elegido la opción de techo inclinado o a dos aguas.

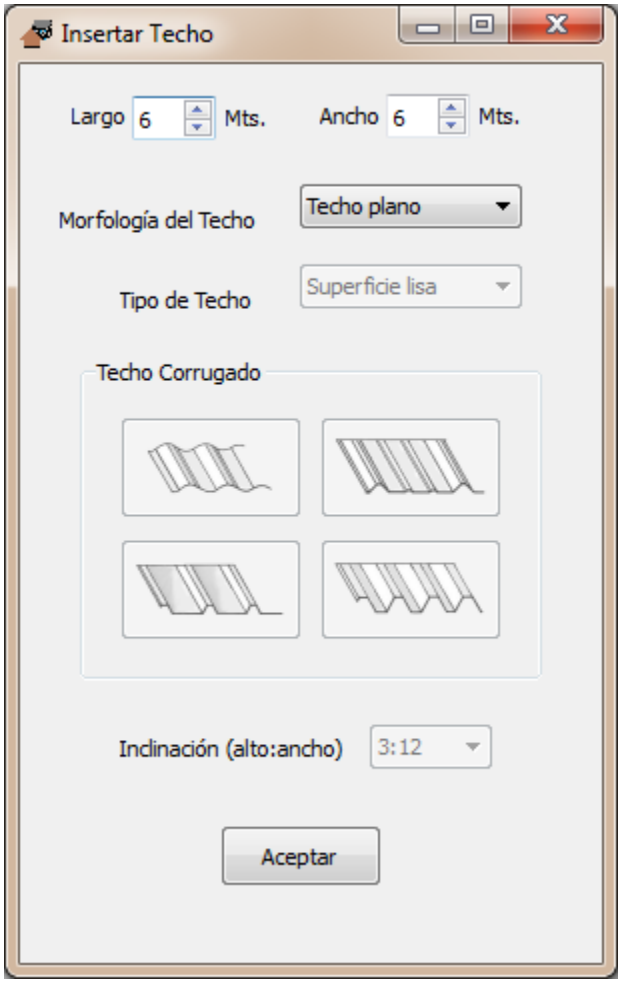


Figura 23: Ventana de interfaz para configurar el sistema de techos.

Esta clase hereda de “QMainWindow” quien le permite la captura de eventos tal como se observa en la Figura 23. Al igual que en el caso de la clase “canaleta”, es la clase “QTOpenGL” quien crea la instancia de esta clase.

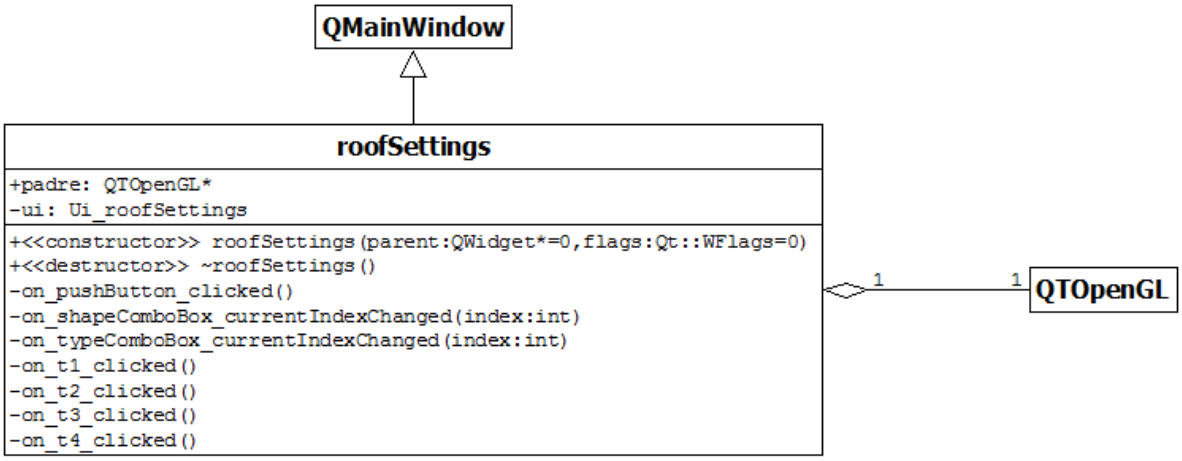


Figura 24: Diagrama de clases de la clase roofSettings.

La clase base de toda la aplicación es la clase “Escena”. Sin embargo hay otros elementos relacionados a la creación y manejo de los elementos de PhysX que deben ser controlados por otras clases diferentes a la Escena. Las clases que controlan los elementos de PhysX son:

g) HelperActor

Esta clase controla la creación de actores en la escena utilizando diferentes shapes. Es posible crear los shapes nativos: box, capsule y plane y también un actor basado en un convexMesh o un *NxTriangleMesh*. Para la creación de los techos inclinados o techos a dos aguas de superficie lisa se utilizan elementos de tipo box. Para la creación del suelo, que además es un *drain*, la clase fluido utiliza a la clase “HelperActor” para crear un actor con forma *NxPlaneShape*. La estructura de la edificación se crea utilizando un *NxConvexMesh* y los techos corrugados, tejas y canaletas se cargan como un *NxTriangleMesh* a partir de un archivo .obj.

Para la creación de estos elementos, la clase “HelperActor” cuenta con los métodos: “CreateBox”, “CreateCapsule”, “CreateGroundPlane”, “crearTriangleMesh” y finalmente “crearConvexMesh”; tal como se observa en la Figura 25.

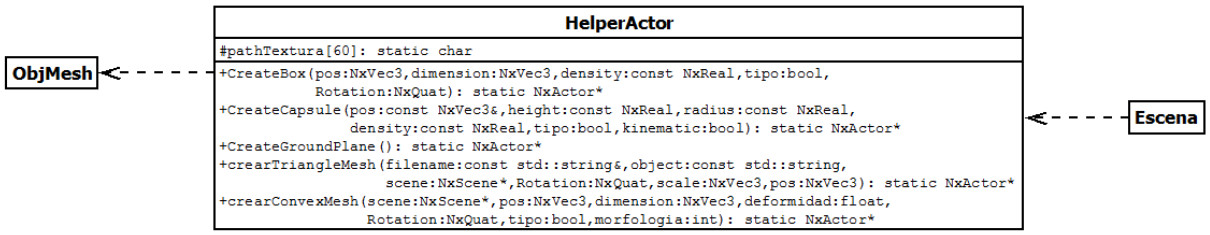


Figura 25: Diagrama de clases de la clase HelperActor.

h) DrawObjects

Despliega los actores en forma de mallado o de geometría. Todos los actores, a excepción de los *NxTriangleMesh* son desplegados desde la escena por medio de esta clase. La Figura 26 muestra los atributos y métodos más importantes que forman parte de esta clase.

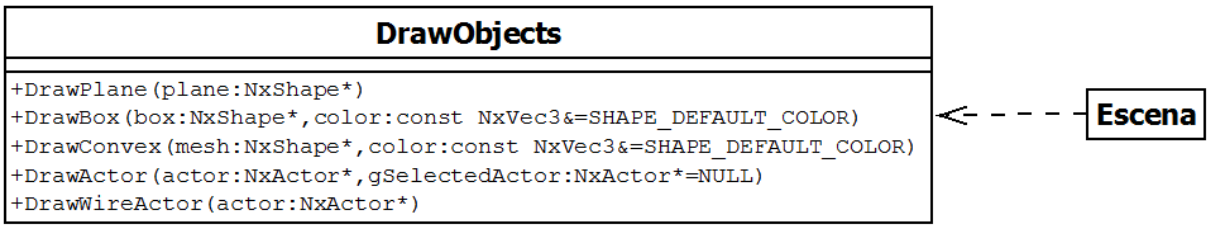


Figura 26: Diagrama de clases de la clase DrawObjetcs.

i) Fluido

Para la gestión del fluido se instancia desde la clase “Escena” una clase llamada “Fluido”, que se encargará de controlar la creación y eliminación de dicho fluido. Esta clase tiene seis métodos: “CrearFluidEmitter”, “CrearFluido”, “CrearDrain”, “bindVBO”, “unbindVBO” y “UpdateVBO”, como se aprecia en la Figura 27.

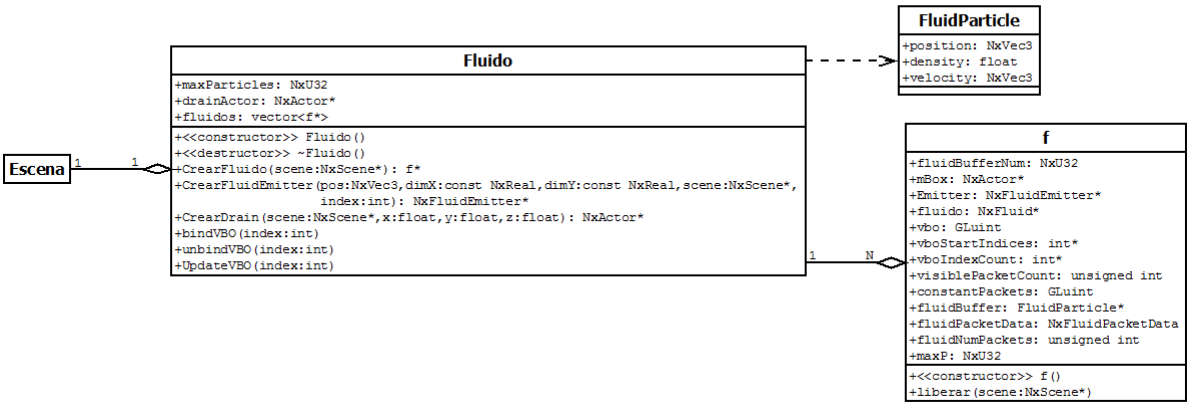


Figura 27: Diagrama de clases de la clase Fluido.

El método **"**CrearFluidEmitter" crea una instancia de la clase *NxFluidEmitterDesc* e inicializa algunos de sus atributos. Desde allí se crea también el fluido invocando al método "CrearFluido". Este método recibe como parámetro la posición y la dimensión del emisor, así como una referencia a la escena. El actor del emisor es creado por la clase "HelperActor" como un *NxActor* estático, al cual se le adjunta un *emitter*, es decir, tendrá las propiedades y el comportamiento de un *emitter*.

Algunos atributos importantes del *NxFluidEmitterDesc* creado por el método "CrearFluidEmitter" son:

- **rate:** Cantidad de partículas que son emitidas por segundo. La tasa sólo es considerada durante la simulación si es del tipo *NX_FE_CONSTANT_FLOW_RATE.* Este valor se calcula dividiendo la cantidad máxima de partículas entre la constante 300.
- **randomAngle:** Indica la desviación aleatoria en la que han de ser generadas las partículas desde el emitter. Este ángulo es relativo a la orientación del emitter con respecto al eje *z*. Este valor ha sido fijado en 0,1.
- **randomPos:** Es la variación aleatoria de la posición de las partículas relativa a la posición del *emitter* con respecto al eje *z*. El valor que recibe este parámetro es del tipo *NxVec3*, en este caso ha sido un vector de valor 0 en todas sus coordenadas.
- **fluidVelocityMagnitude:** Magnitud de la velocidad de las partículas del fluido generadas por el emitter. Note que la velocidad máxima de partículas está limitada por el atributo *NxFluid::motionLimitMultiplier*. En este caso las partículas inician con velocidad 0, puesto que caen por gravedad.

- **repulsionCoefficient:** Define un factor de transferencia del impulso con que son despedidas las partículas desde el emisor. Como las partículas de lluvia se precipitan en forma de caída libre sin ninguna fuerza o velocidad inicial, este valor ha sido inicializado en cero.
- **maxParticles:** Determina la cantidad máxima de partículas que han de ser generadas por el *emitter*. El *emitter* detendrá la creación de partículas cuando este límite sea alcanzado y comenzará de nuevo una vez que el número de partículas del fluido esté por debajo del límite. Si se establece en 0, el número de partículas emitidas es libre (hasta el máximo para fluidos). En la Tabla 7 podemos apreciar la cantidad de partículas utilizadas para la simulación de diferentes intensidades de precipitación en este trabajo.

Nro. de Partículas	Intensidad media (mm/h)
50000	50
100000	75
150000	100
250000	125
350000	150
600000	200

Tabla 7: Relación Partículas – Intensidad media de precipitación generadas en la simulación.

- **particleLifetime:** Tiempo de vida en segundos de las partículas que genera el emitter. Si se coloca en cero, las partículas vivirán hasta llegar a un *drain*. En este caso las partículas han sido configuradas con un tiempo de vida largo de 50 segundos, para evitar que desaparezcan antes de llegar al suelo, pero garantizando su eliminación en caso de quedar atascadas en algún lugar que les impida llegar al *drain*.
- **type:** Tipo de emisor que puede ser constante de presión (*NX_FE_CONSTANT_PRESSURE*) o constante de tasa de flujo (*NX_FE_CONSTANT_FLOW_RATE*), siendo este último el elegido en este trabajo.
- **shape**: Forma del *emitter*. Existen dos opciones: Rectangular (*NX_FE_RECTANGULAR*) o Elíptica (*NX_FE_ELLIPSE*). Para este trabajo se utiliza un emisor rectangular.
- **flags**: |= NX_FEF_FORCE_ON_BODY;

El método **"**CrearFluido" invoca al método *NxScene::createFluid()* con su descriptor apropiado para crear el fluido. Luego se asignan todos los parámetros al fluido:

- **kernelRadiusMultiplier:** Controla el radio de la esfera de influencia para la interacción de partículas. El valor utilizado es 2,0.
- **restDensity:** Densidad esperada para el fluido. Para el agua la densidad aproximada es 1000.
- **restParticlesPerMeter:** Este parámetro está relacionado a la densidad del fluido, se refiere a la cantidad de partículas por metro lineal, medido cuando el fluido está en su estado de reposo. El valor utilizado es 0,7. Cuando se simula un fluido, la masa de cada partícula es constante, sin embargo la densidad varía a través del tiempo debido a la compresibilidad del fluido. Por lo tanto, el cambio de densidad es logrado a través del movimiento de partículas. Un área del fluido con alta densidad será producto de un conjunto de partículas que se encuentran muy cercanas las unas de las otras en un momento dado.
- **stiffness**: Especifica la rigidez de las partículas, relacionadas con la presión. En este caso el valor asignado es 1,0.
- **viscosity:** Define el comportamiento viscoso del fluido. Se toma como valor 1, lo correspondiente a la viscosidad del agua a 20°C, que es un estimado de la media de la temperatura cuando llueve.
- **damping:** Constante global de velocidad de amortiguación para todas las partículas. El valor utilizado es 0,2. Este parámetro ayuda a simular la resistencia de las partículas al caer o moverse en otras direcciones, debido a la fuerza de la gravedad o al viento.
- **surfaceTension:** La tensión superficial es modelada a través de una fuerza de atracción entre partículas. En el interior del fluido, las fuerzas de tensión superficial se equilibran entre sí. Sin embargo, en la superficie del fluido ellos actúan en la dirección opuesta a la normal de la superficie, la cual tiende a reducir la curvatura de la superficie del fluido.
- **motionLimitMultiplier:** Distancia máxima en la que una particular podría desplazarse en un instante de tiempo. Este valor está configurado en 16.
- **packetSizeMultiplier:** Permite paralelizar el fluido en diferentes paquetes, en este caso la cantidad de paquetes ha sido limitada a 8.
- **collisionDistanceMultiplier:** Define la distancia de la colisión entre las partículas y la superficie de la geometría, este valor está configurado en 0.5 para que al interceptarse las partículas con la superficie de la geometría se simule un poco la forma en la que una gota de agua se esparce al colisionar con un cuerpo rígido con permeabilidad nula o poca.

- **dynamicFrictionForStaticShapes:** Coeficiente de fricción dinámico con respecto a cuerpos estáticos. Su valor es 0.
- **dynamicFrictionForDynamicShapes:** Coeficiente de fricción dinámico con respecto a cuerpos dinámicos. Su valor es 0.
- **staticFrictionForDynamicShapes:** Coeficiente de fricción estático con respecto a cuerpos dinámicos. Su valor es 0,5.
- **staticFrictionForStaticShapes:** Coeficiente de fricción estático con respecto a cuerpos estáticos. Su valor es 0,5.
- **restitutionForStaticShapes:** Define el coeficiente de restitución usado par alas colisiones de las partículas del fluido con formas estáticas. En este caso el valor asignado es cero.
- **collisionResponseCoefficient:** Define el factor de transferencia del impulso cuándo las partículas del fluido colisionan con un cuerpo rígido, este valor está configurado en 0,2.
- **collisionMethod:** Es una bandera que indica si la colisión con el entorno, bien sea estática o dinámica, ha de realizarse. En este caso se activan ambas asignado los valores NX_F_STATIC y NX_F_DYNAMIC.
- **flags:** Bandera que permite definir con exactitud el conjunto de propiedades activas en el fluido. En este caso se activa la colisión de las partículas con otros actores y viceversa por medio del valor NX_FF_COLLISION_TWOWAY.

El método “CrearDrain” recibe como parámetro la escena y la posición donde ha de ser creado el *drain*. La primitiva de PhysX utilizada como forma para este *drain*, es un *NxPlaneShape* que viene a ser el suelo donde se sostiene la estructura. Al elegir un plano que simule el suelo, no solo se está creando un *drain* que mantenga bajo control la cantidad de partículas sino que además simula lo que realmente ocurre, que el agua es absorbida por el suelo hasta desaparecer por efecto de la percolación.

El método “bindVBO” verifica la dimensión de los paquetes del fluido para copiar los bloques de memoria de PhysX a OpenGL. Luego estos bloques de memoria serán desplegados haciendo uso del *Vertex Buffer Object* (VBO) de OpenGL.

Cuando el *Vertex Buffer Object* de OpenGL no va a ser utilizado nuevamente, el método “unbindVBO” lo desenlaza. Este método desvincula todos los arreglos inherentes a las propiedades de las partículas (posición, densidad, velocidad) del VBO.

El método “UpdateVBO” establece los arreglos de posición, densidad y velocidad perteneciente al VBO para luego desplegar con “glMultiDrawArrays”. Dicho de otra forma, reserva el espacio

de memoria e indica los apuntadores a dichos bloques donde el método “bindVBO” ha de copiar la información de OpenGL para que el VBO pueda manejarla y desplegarla.

j) Escena

Es la clase principal del proyecto, casi todos los elementos son instanciados por esta clase como se aprecia en la Figura 28. Esta clase es un *singleton*, desde ella se crean todos los objetos de PhysX y se hace el despliegue final.

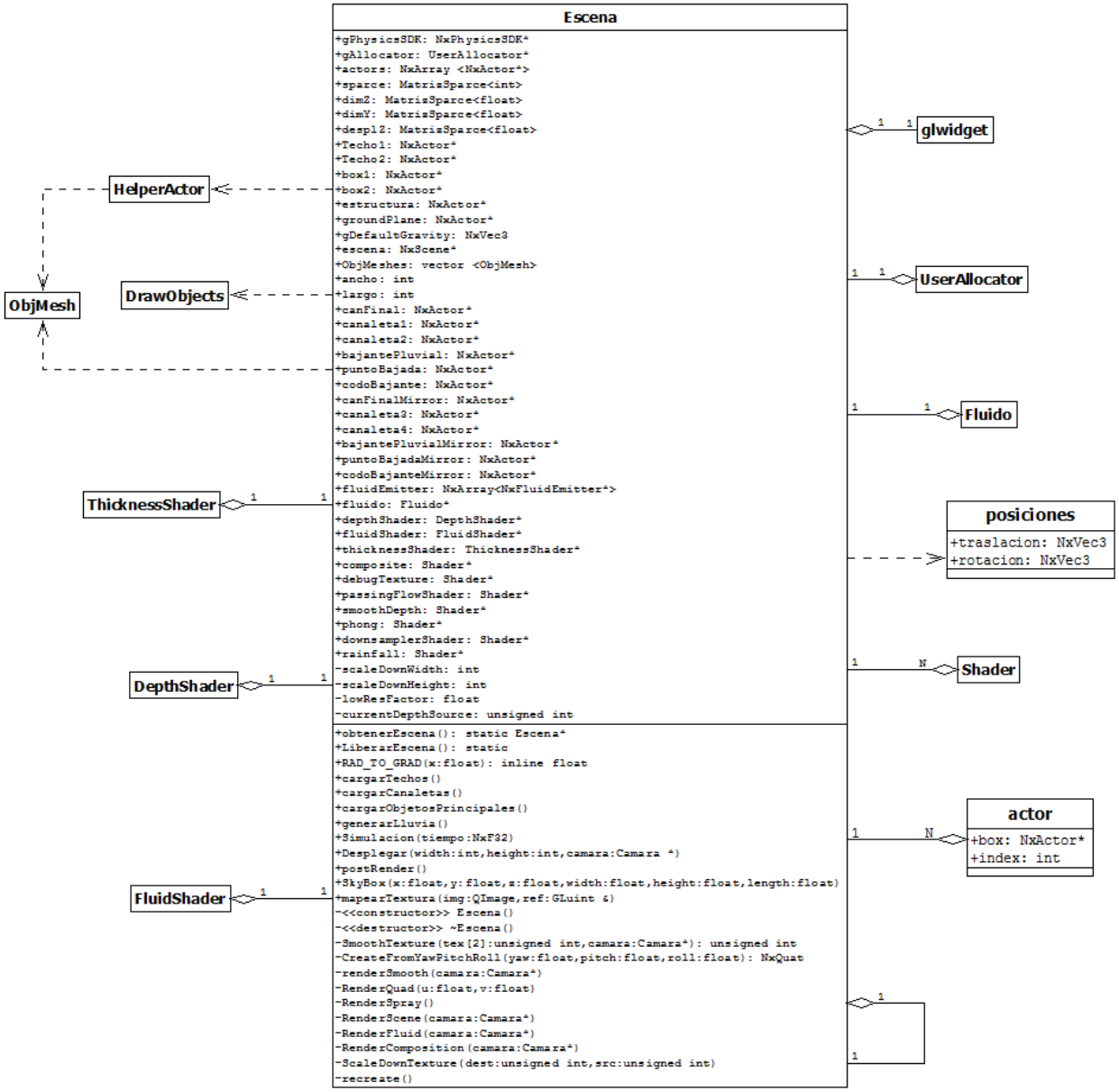


Figura 28: Diagrama de clases de la clase Escena.

Los métodos de esta clase son:

- **Escena:** Es en el constructor de la escena donde se crea el *NxPhysicsSDK*, clase base para trabajar con PhysX, tal como se explicó en el CAPÍTULO III. Se instancian algunos parámetros del *NxPhysicsSDK* y se crea el *NxSceneDesc*. A partir de allí se pueden crear los diferentes actores de *PhysX*, incluyendo al fluido (ver Figura 6). Se instancian otros atributos de la escena como la gravedad, los actores y las variables de control con un valor por defecto. Así mismo, se llenan las tres matrices esparcidas que contienen la relación intensidad-área de techo y las dimensiones de los puntos de bajada con relación a la forma de los mismos (bajante circular o rectangular).

- **obtenerEscena:** Retorna una instancia de Escena si ya existe, sino la crea por primera y única vez (*singleton*).

- **cargarTechos:** Funciona como un método *Factory*. Utiliza varias variables de control para saber qué tipo de techo ha de instanciar y sus detalles de configuración. La variable "morfología" indica si el techo es plano, inclinado o a dos aguas. Cuando el techo es a dos aguas se crean dos *NxActor* para el techo y se trasladan de forma tal que se unan en el plano que contiene el eje y como si fuera un espejo. La variable "tipo" indica que tipo de techo se va a cargar (superficie lisa, tejas o techo corrugado) y la variable "techoCorrugado", el tipo de techo corrugado. Ambas variables determinan el archivo que la clase *HelperActor* ha de cargar como un *triangleMesh*. Si la superficie es lisa, se le indicará a *HelperActor* que debe crear un *NxActor* a partir de un box.

- **cargarCanaletas:** Al igual que en el método anterior utiliza variables de control para verificar las configuraciones del sistema de canaletas que ha de cargar. Cuando la variable "morfología" indica que el techo es a dos aguas, se deberá duplicar todo el sistema de canaletas para adjuntar uno a cada alero del techo. La variable de control "tipoCanaleta" indicará si el sistema de canaletas tendrá forma Biselada, Pecho Paloma, Semicircular o Estilo K. La variable de control "desagueCanaleta" indicará si el segmento final ha de ubicarse a la derecha, en el centro o del lado izquierdo. La variable de control "elbow" permite saber el ángulo de inclinación del punto de bajada. Además de verificar las configuraciones pertinentes a la creación del sistema de canaletas, este método realiza todas las transformaciones afines a cada uno de los elementos de la escena para ubicarlos, rotarlos y escalarlos de manera adecuada.

La clase Escena tiene un "*struct* posiciones" que almacena un valor de traslación y rotación. A través de un *map* se vincula una etiqueta con un elemento de este *struct* por cada actor creado. Tanto el método "cargarTechos" como "cargarCanaletas" se encargarán de etiquetar a cada actor utilizando esta estructura. Esto servirá para que más adelante el método "RenderScene" pueda acceder a esta información y aplicar las transformaciones adecuadas, ya no a nivel de actor sino a nivel de despliegue.

- **cargarObjetosPrincipales:** Carga los objetos que han de aparecer en la escena por defecto, en este caso solo el "groundPlane" y le aplica una textura llamando a la clase "mapearTextura".

- **generarLluvia:** Crea una instancia de la clase Fluido y llama al método crearFluidEmitter.

- **Simulacion:** Crea la simulación de *NxScene*.

- **postRender:** Verifica si la simulación ha sido reiniciada a través de un booleano antes de liberar el *NxPhysicsSDK*, la escena y todos los elementos que la contienen. Lo primero que elimina es el fluido en caso de que haya sido creado, los emisores y vacía los arreglos que contienen las referencias a los fluidos y a los emisores. Luego elimina los actores (sistema de techos y canaletas, estructura y el "groundPlane") siguiendo un proceso similar a lo ocurrido con los fluidos. Lo último en liberarse es la escena y el *NxPhysicsSDK.* Finalmente se crea de nuevo una escena vacía y se cargan los objetos principales.

- **SkyBox:** Despliega el skyBox previamente cargado por la clase "glwidget".

- **mapearTextura:** Habilita el mapeo de texturas de OpenGL y asigna la imagen y las coordenadas de textura.

- **Desplegar:** Organiza la secuencia de despliegue de la aplicación. La Figura 29 ilustra la secuencia que ha de seguir la aplicación para el despliegue gráfico, tanto los métodos que son llamados directamente por el método desplegar como los que constituyen una llamada indirecta, la instancia de otras clases o la aplicación de shaders.

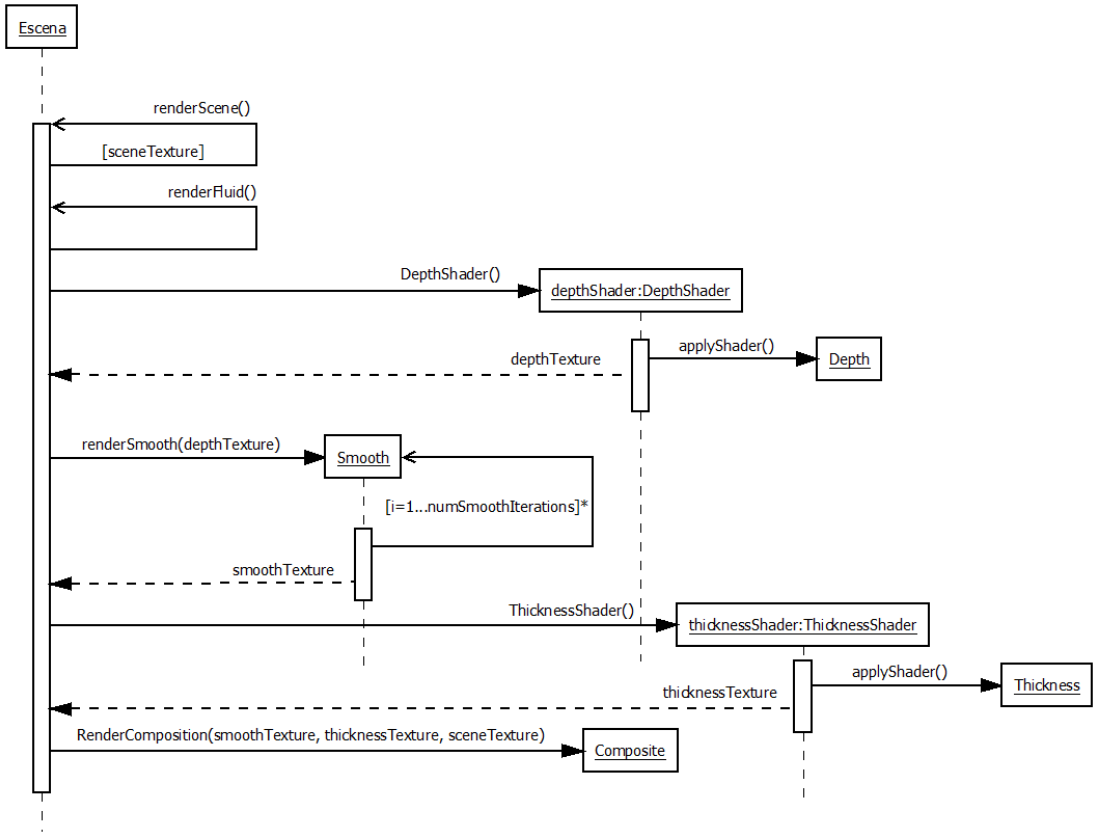


Figura 29: Diagrama de secuencia del proceso de despliegue.

- **RenderScene:** Se encarga del despliegue de todos los elementos de la escena a excepción del fluido. En este método, se relacionan todos los elementos a ser renderizados con la textura que le corresponde. El primer elemento que se despliega es el Skybox puesto que es lo que se encuentra más distante y debe ser renderizado antes de activar el Z-buffer para el despliegue de los demás elementos. Para ello se hace una llamada a la función *SkyBox*. Luego se renderizan todos los modelos 3D, utilizando la etiqueta del elemento como identificador, lo cual permitirá saber qué transformaciones afines y qué texturas serán aplicadas a dicho objeto. Finalmente se despliegan todos los actores basados en primitivas de PhysX, esto incluye el suelo, un *NxPlaneShape; la estructura, un NxConvexShape;* y los techos inclinados de superficie lisa que utilizan como primitiva un *NxBoxShape***.** Todo esto es plasmado en una única textura que será utilizada posteriormente por la función "RenderComposition" para el despliegue final como se aprecia en la Figura 30.

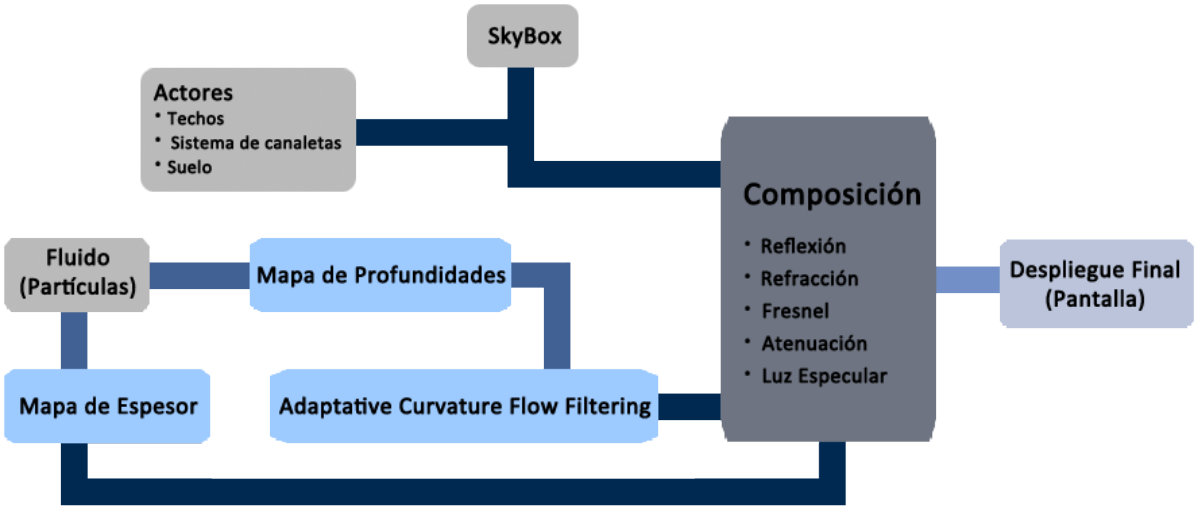


Figura 30: *Pipeline* de despliegue.

- **renderSmooth:** Es invocado por el método "RenderFluid". Carga el shader Smooth y retorna a RenderFluid una textura con el mapa de profundidades de las partículas suavizado (*Adaptative Curvature Flow Filtering*).
- **RenderQuad:** Dibuja un recuadro que cubre toda la pantalla para hacer las veces de lienzo, en donde las funciones "RenderComposition" y "renderSmooth" van a plasmar todo lo que ha de desplegarse o alguna textura intermedia. Las posiciones de los vértices del Quad están en coordenadas de pantalla y están normalizadas entre -1 y 1. Así mismo las coordenadas de textura se encuentran normalizadas entre 0 y el tamaño de pantalla para preservar el tamaño de los píxeles.
- **RenderSpray:** Crea las configuraciones de lluvia para que el shader "Rain" pueda desplegar algunas partículas como lluvia. Lo primero que hace es asignar el color de las partículas, luego verifica que el fluido exista antes de comenzar a configurar los valores. Una vez comprobada la existencia del fluido habilita el test de profundidad, la transparencia a partir de mezcla aditiva, y llama al shader "Rain" para que haga los cálculos correpondientes.
- **RenderComposition:** Llama al shader Composite para unir todas las texturas y generar la composición de ellas para el despliegue final.
- **RenderFluid:** Establece las configuraciones de despliegue necesarias para ejecutar los shaders inherentes al render del fluido. Luego de ello invoca a dichos shaders.

k) ObjMesh

Maneja toda la estructura de los modelos 3D para cargar un modelo a partir de un archivo .obj y a partir de los datos del modelo generar el mallado, exportar las coordenadas de textura, las normales, el material, etc. La Figura 31 muestra los principales métodos y atributos que forman parte de esta clase, así como las estructuras que utiliza y las clases con quienes se relaciona.

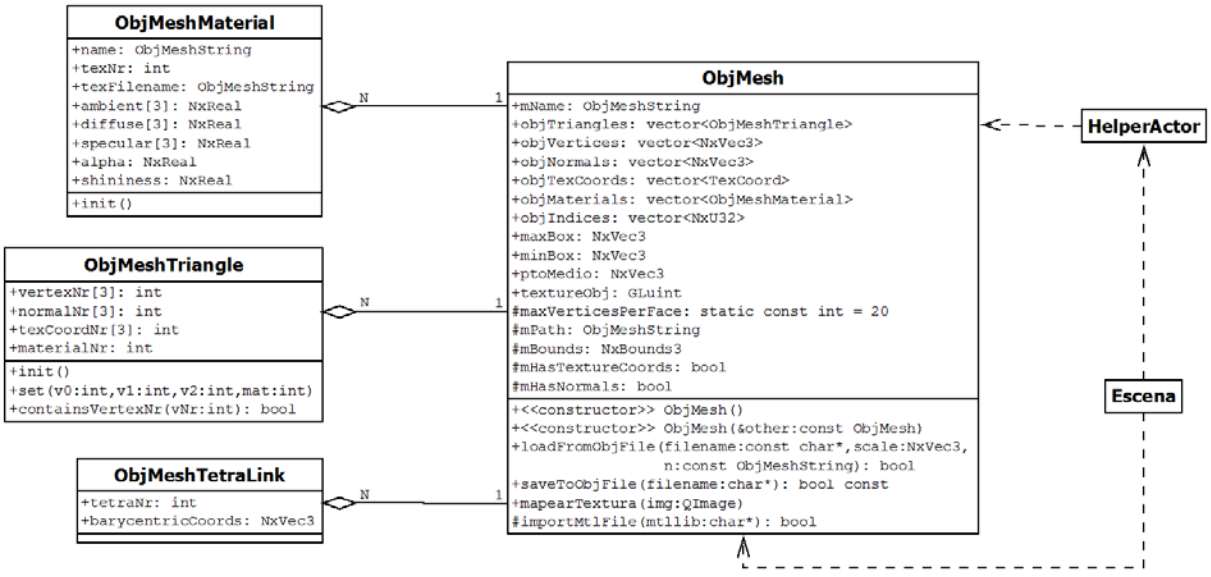


Figura 31: Diagrama de clases de la clase ObjMesh.

l) Shader

Dado el nombre de un archivo de fragmento y/o de vértice, esta clase carga el *shader*, lo compila y lo vincula a OpenGL. Además esta clase es la encargada de activar y/o desactivar el shader y de pasar por parámetro los atributos y texturas a los diferentes *shaders*.

La Figura 32 muestra en un diagrama los elementos que forman parte de él.

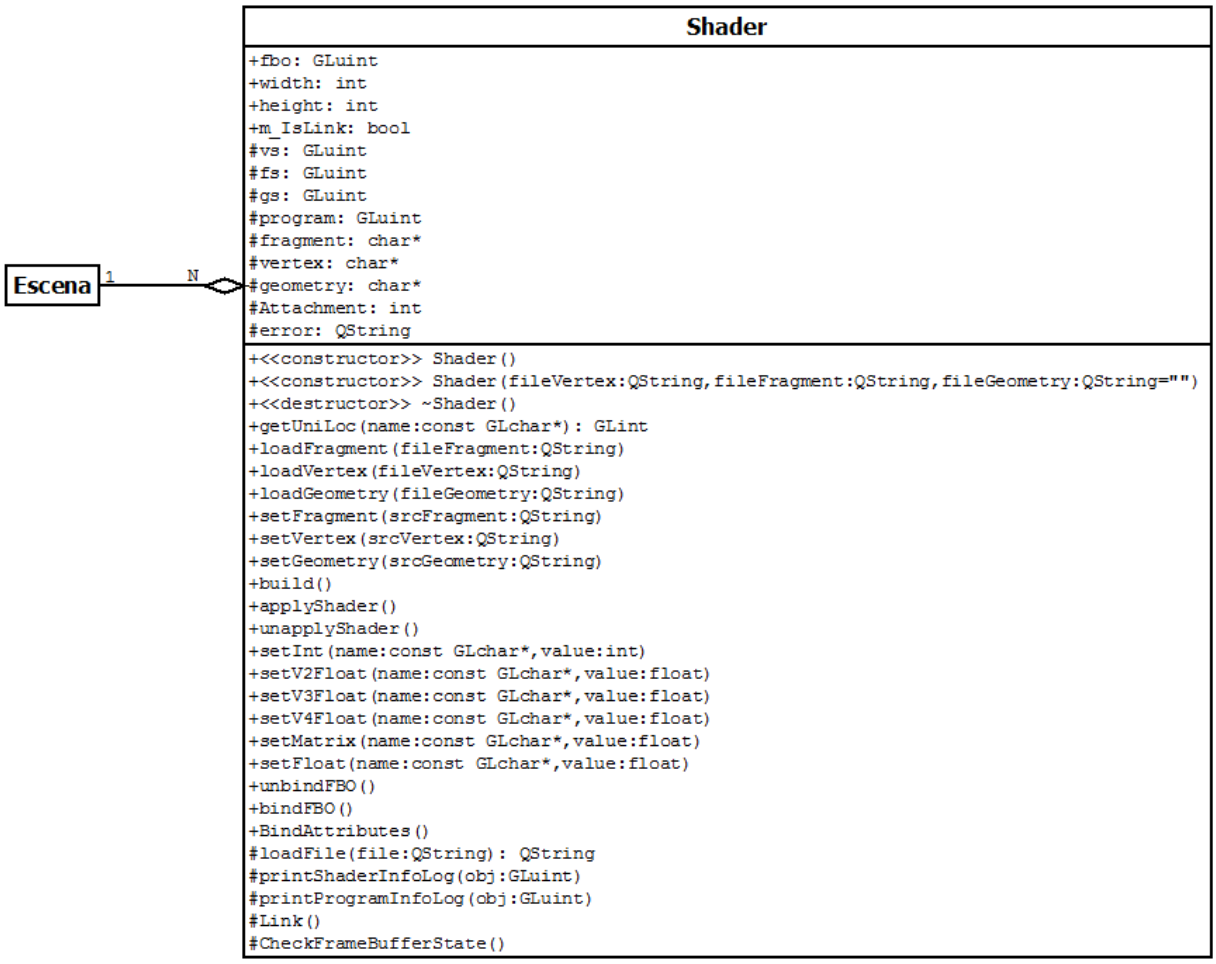


Figura 32: Diagrama de clases de la clase Shader.

m) Matrix

Contiene la estructura y un conjunto de operaciones sobre dicha estructura, necesarias para el cálculo de la rotación de la cámara. La Figura 33 muestra el detalle de la clase “Matrix”.

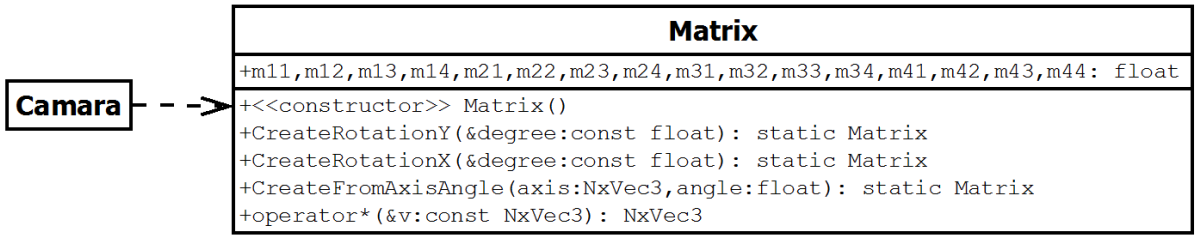


Figura 33: Diagrama de clases de la clase Matrix.

n) UserAllocator

Hereda de la clase NxUserAllocator para gestionar el proceso de reserva de memoria que necesita PhysX SDK para instanciar cualquier elemento del API. Los elementos que intervienen en esta clase están ilustrados en la Figura 34.

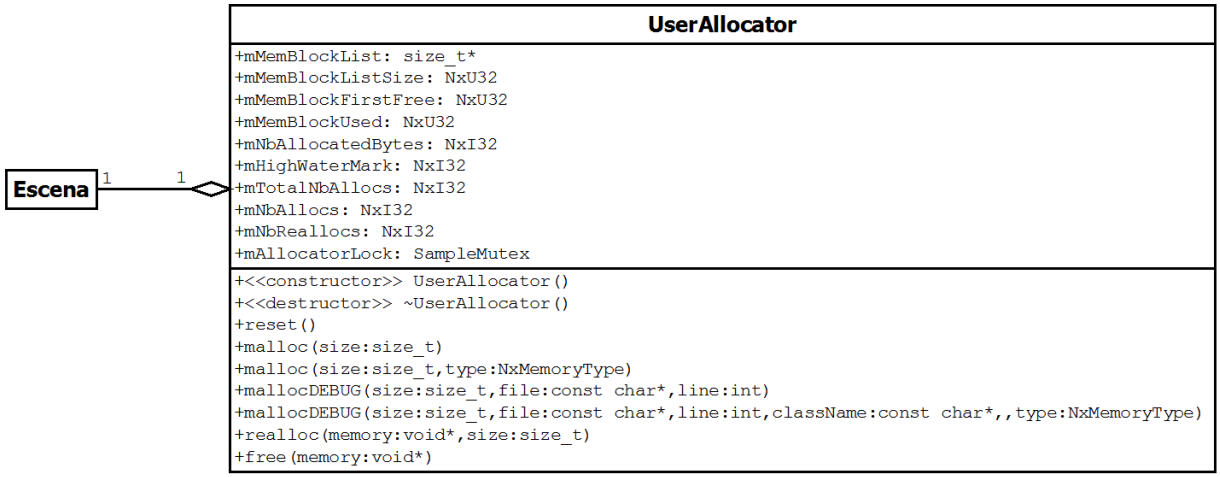


Figura 34: Diagrama de clases de la clase UserAllocator.

o) DepthShader

Llama al Shader “Depth” para que haga el cálculo del mapa de profundidades, ver Figura 35.

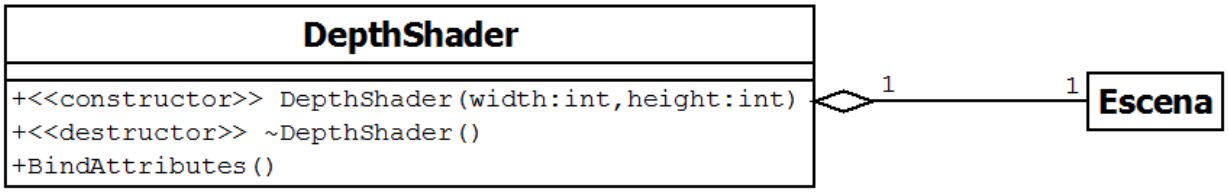


Figura 35: Diagrama de clases de la clase DepthShader.

p) FluidShader

Esta clase gestiona todas las texturas intermedias que van a ser utilizadas durante el proceso de despliegue, además es quien guarda los parámetros de despliegue del fluido, tales como: el grosor, la refracción, coeficiente de atenuación, tamaño de las partículas, etc. Todos los atributos y métodos que forman parte de esta clase pueden apreciarse en la Figura 36.

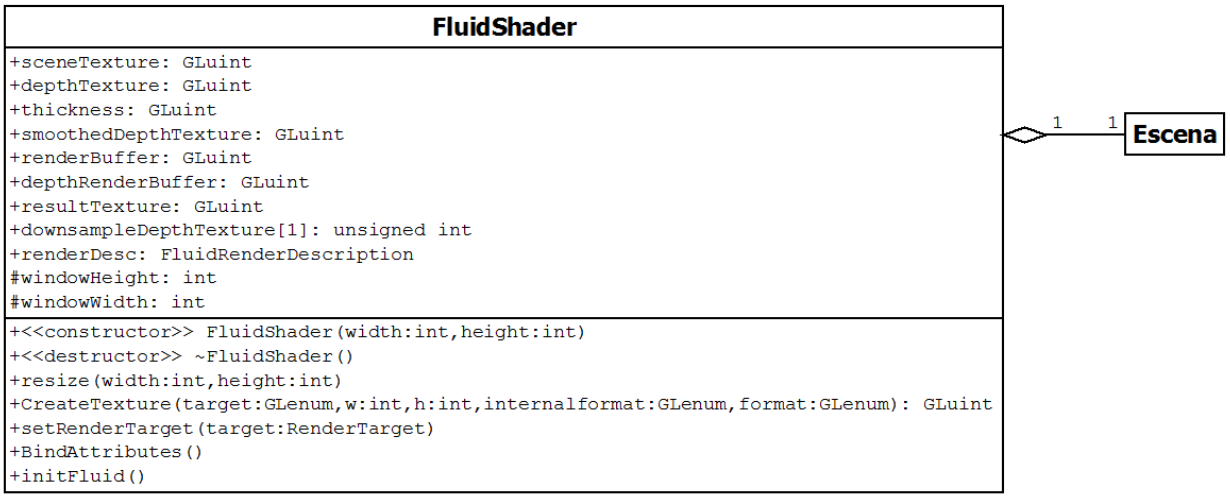


Figura 36: Diagrama de clases de la clase FluidShader.

q) ThicknessShader

Llama al Shader “Thickness” para que haga el cálculo del mapa de profundidades, ver Figura 37.

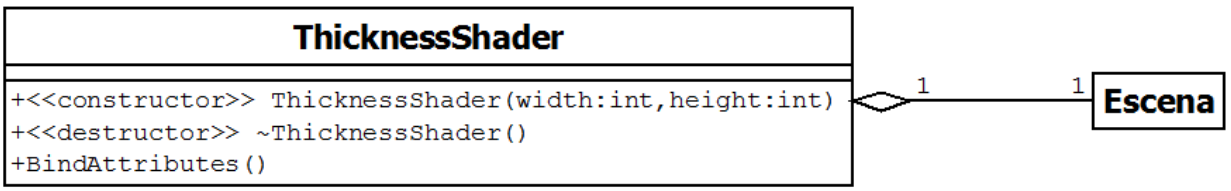


Figura 37: Diagrama de clases de la clase ThicknessShader.

3. Shaders

A continuación se describen todas las funcionalidades de los shaders utilizados en este trabajo. El Phong utilizado para calcular la iluminación y todos los demás son empleados en el render del fluido.

a) Phong

Este shader se encarga de calcular la iluminación con el método de sombreado Phong en la escena. En el *vertex shader* se calculan los vectores V y L a partir de la posición y la dirección del rayo de luz, con lo cual el *fragment shader* ha de calcular la luz especular y difusa respectivamente. Además se almacena en el vector N el valor de la normal transformado a coordenadas de normales.

La función "reflect" de GLSL permite saber la cantidad de energía luminosa especular que incide en la superficie, la cual se almacena en el vector R. La luz ambiental se calcula a partir del color del material y la cantidad de luz ambiental que recibe la superficie. Del mismo modo ocurre con la luz difusa, con la diferencia de que estas se multiplican por el producto punto entre N y L.

El cálculo de la luz especular se obtiene a partir del valor de luz especular multiplicado por la luz incidente en la superficie. Para obtener esto último se calcula el producto punto entre R y V y se eleva por el valor de brillo del material (*shininess*). Para descartar la luz que incide por la parte de atrás de los polígonos no se toma en cuenta los valores negativos, así se evita calcular la iluminación en las caras traseras.

b) Depth

Este shader se encarga básicamente de calcular el mapa de profundidades de las partículas aplicando splatting y además modifica el Z-buffer para que el sprite parezca una esfera.

En principio se debe saber que las partículas del fluido son desplegadas como un sprite de OpenGL por el método “RenderFluid” de la clase “Escena”, la cual crea el shader “Depth” y le pasa como parámetro todos estos sprites. A partir del sprite, cuya forma es un cuadrado el fragment shader descarta los fragmentos que no estén dentro de la esfera utilizando la fórmula de la circunferencia.

El único dato que se almacena del sprite es su centro. El sprite va de -1 a 1 por lo que el radio de una esfera contenida dentro de este debería medir 1. Si la distancia desde el centro hasta un fragmento cualquiera es mayor que uno quiere decir que se encuentra fuera de la circunferencia, con lo cual es descartado. La Figura 38 ilustra esto gráficamente.

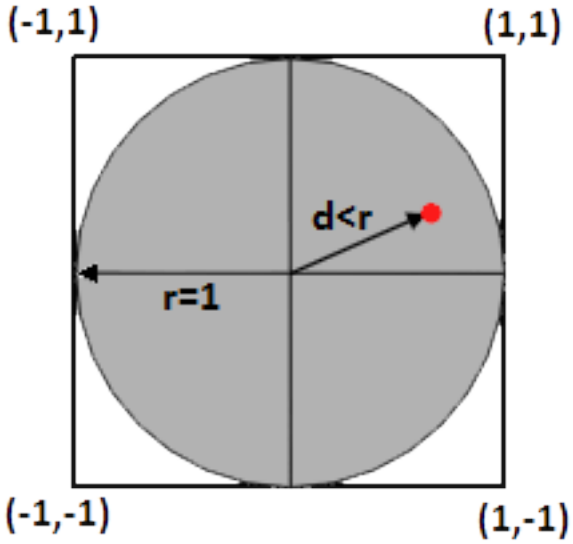


Figura 38: Textura *Sprite* para el despliegue del fluido.

Una vez que se han convertido los sprites en circunferencias se procede a aplicar el *splatting*, para ello se determina cuáles son las caras delanteras de la superficie del fluido desde la posición de la cámara. De este modo, se renderizan todas las partículas como esferas sobre un mapa de profundidades. Al utilizar este mapa de profundidades se asegura que sólo los píxeles más cercanos al viewport sean almacenados.

Antes de activar el shader el mapa de profundidad es inicializado con el valor -10000 para todos sus téxeles, lo cual indica un valor infinito relativo. Además, la mezcla de colores es deshabilitada y el test de profundidad por hardware es habilitado para asegurar que los valores más cercanos de cada píxel sean conservados.

c) Downsample

Constituye una optimización para el shader “Smooth” que sirve para reducir el tamaño del buffer de profundidad a la mitad antes de aplicar la técnica *Adaptative Curvature Flow Filtering.* Esto supone sacrificar un poco la calidad de imagen para obtener una mejora en el rendimiento del algoritmo.

d) Smooth

Este shader está basado en la técnica de *Adaptative Curvature Flow Filtering* y es ejecutado después de obtener el mapa de profundidades del fluido. Esta técnica realiza una serie de iteraciones en donde cada una de ellas resuelve una integración euleriana.

Dado un fragmento se debe evaluar si su profundidad no corresponde al infinito relativo, en ese caso se procede a determinar el valor de las profundidades de dicho fragmento, calculando las diferencias finitas de dicho valor en las vecindades del fragmento. Luego se calcula la derivada direccional a partir de la cual se obtiene el valor de la curvatura media.

Es importante destacar que antes de calcular el valor de la curvatura media se debe evaluar los valores resultantes del cálculo de la derivada direccional para determinar si pertenecen al mismo parche de superficie de fluido. Esto se consigue comparando las derivadas direccionales con el valor del coeficiente de atenuación *“blurDepthFalloff”*, que viene a hacer las veces de umbral.

La Ecuación 5.4 se resuelve multiplicando las derivadas finitas por unas constantes calculadas a partir de algunos parámetros del punto de vista. Con esto se calcula la curvatura media expresada en la Ecuación 5.7 y finalmente se obtiene el color del fragmento, el cual usa el canal rojo para almacenar la profundidad desplazada.

e) Passing

Dado que no es posible leer y escribir una textura al mismo tiempo, se requieren dos texturas para utilizar la técnica *Adaptative Curvature Flow Filtering*, de modo que mientras el shader “Smooth” se encarga de leer la textura, el shader “Passing” se encargará de guardar los valores recibidos de “Smooth”.

Esto servirá para intercambiar las texturas en donde se desplegarán las profundidades desplazadas. Por cada iteración de suavizado calculada en el shader “Smooth” se almacenará la textura resultante para poder preservar el valor de profundidad anterior correspondiente a ese fragmento antes de descartalo.

f) Thickness

Este shader se encarga básicamente de calcular el grosor o espesor (*Thickness*) de las partículas aplicando nuevamente splatting. En principio se despliegan el conjunto de partículas del fluido de manera similar al shader “Depth”, con la diferencia de que en este shader se utiliza mezcla aditiva de colores para acumular la contribución de cada partícula y que el Z-buffer obtenido al desplegar la escena es utilizado para comprobar que el grosor del fluido no esté siendo ocluido por alguna geometría de la escena, por esta razón se deshabilita la escritura del Z-buffer.

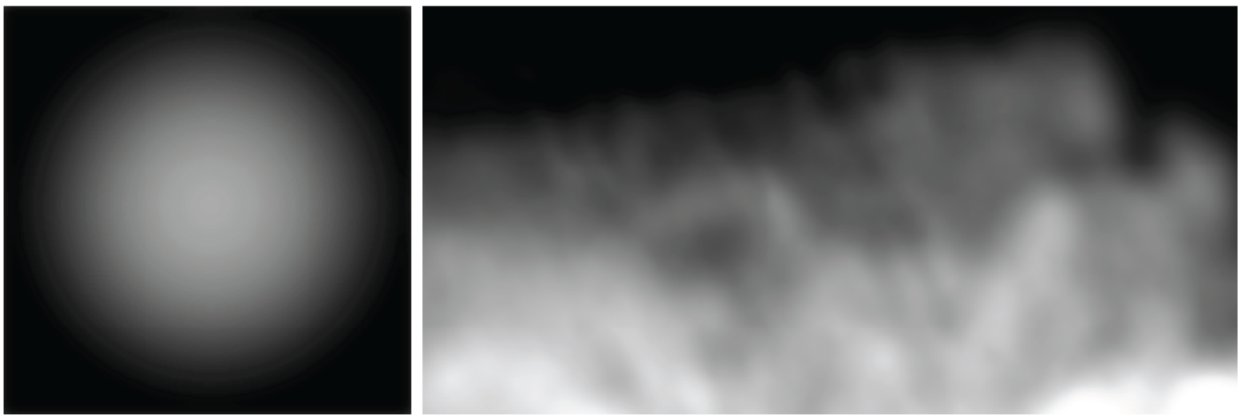


Figura 39: Kernel de *Splatting* utilizado para simular el grosor del agua. A la izquierda se muestra una sola partícula. A la derecha una captura de una capa de fluido con el cálculo de espesor.

El cálculo del grosor está basado en una función exponencial en la cual, las partículas que son acumuladas no presentan discontinuidades o cambios bruscos en el grosor. En la Figura 39 se aprecian los resultados del cálculo del grosor o el espesor del fluido utilizando la técnica de *Splatting*.

g) Composite

Este shader se encarga de componer la imagen final mediante los cálculos realizados en los pasos anteriores. Este proceso es ilustrado brevemente en la Figura 40.

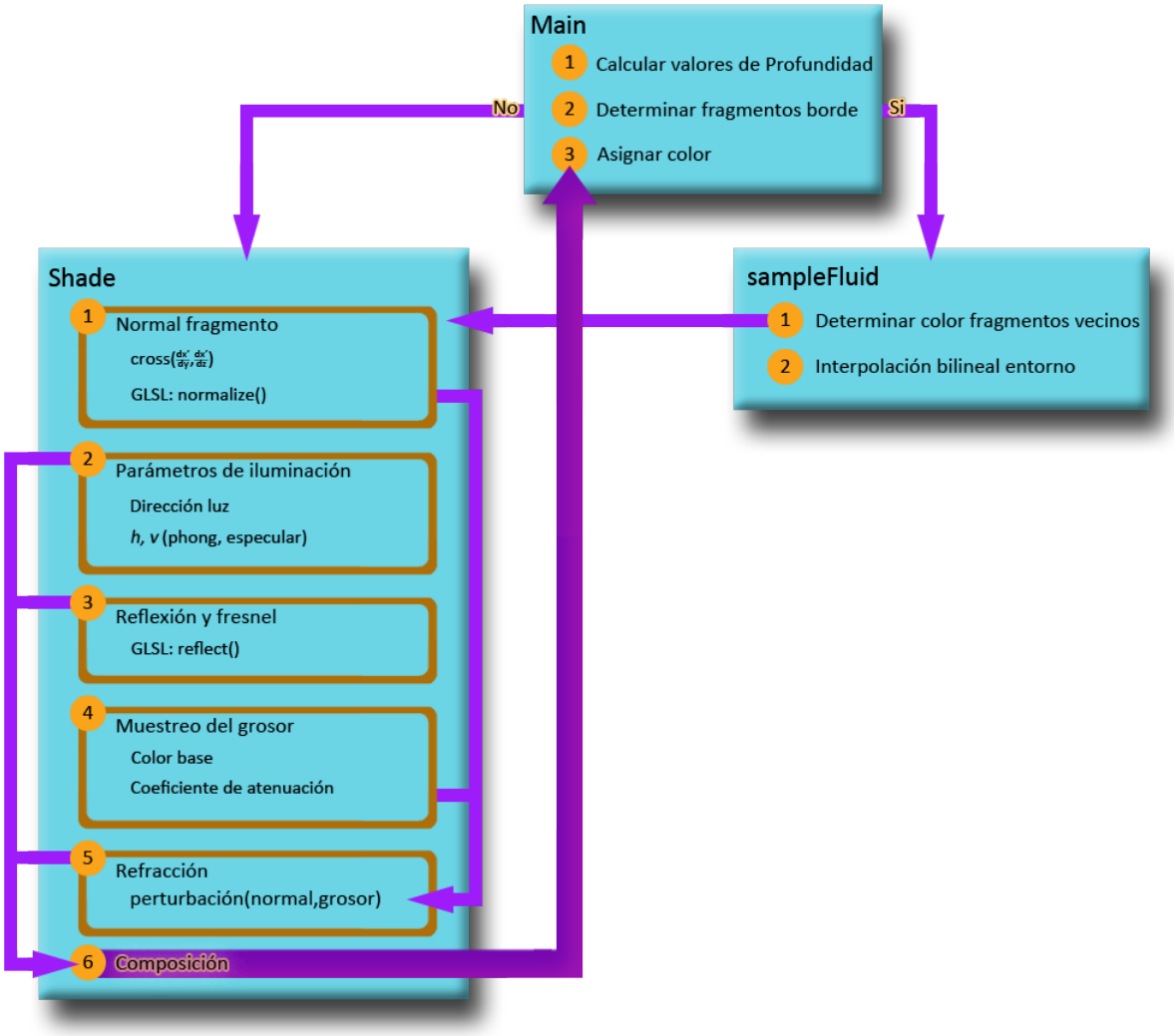


Figura 40: Diagrama de proceso del shader "Composite".

Las funciones que llevan a cabo este proceso son:

- **Main:** Se calcula la profundidad promedio en cada fragmento a partir de los valores de profundidad que estén por debajo del infinito relativo en los vecinos de dicho fragmento. Luego se verifica si el fragmento forma parte del borde de la superficie del

fluido, si es así se realiza sobre él una interpolación bilineal con el entorno para suavizar el límite de la silueta y obtener así el color del fragmento, de lo contrario se hace una llamada a la función “shade” para determinar dicho color. Finalmente se asigna el color calculado al fragmento.

- **ipnormal2:** Realiza una interpolación lineal para el cálculo de la normal del fragmento. Recibe por parámetro el valor de la coordenada de textura y la dirección (x, y) hacia donde se va a realizar la interpolación. Con estos valores se calcula una muestra interpolada del fragmento actual y de dos vecinos haciendo uso de la función texture2DRect de GLSL. Luego se obtiene la profundidad interpolada llamando a la función “ipdepth” y finalmente se calcula una aproximación de la normal interpolada por medio de la función “uvToEyeD”.

- **uvToEyeD:** Recibe por parámetro la dirección del vector, la profundidad interpolada en la dirección dada y la profundidad del fragmento actual, y con estos valores calcula una aproximación de la derivada parcial del fragmento mediante la diferencia central.

- **ipdepth:** Recibe por parámetro tres valores de muestreo del fragmento actual y dos de sus vecinos y con ellos calcula la interpolación de las profundidades haciendo uso de la función “mix” de GLSL.

- **sampleFluid:** Es usado para calcular la composición a los fragmentos que son bordes del fluido. Se determina el valor de la posición del fragmento en coordenadas de vista y con ello se calculan las derivadas parciales, necesarias para determinar el gradiente y con ello obtener los fragmentos que constituyen el borde del fluido. Luego se calcula el color del fragmento llamando a la función “shade”.

- **shade:** Determina el color final de cada uno de los fragmentos que han de ser desplegados. Recibe por parámetro las derivadas de x e y en z, la coordenada de textura del fragmento y la posición del fragmento en coordenadas de vista. Lo primero que se hace es calcular la normal del fragmento, para ello se calcula el producto cruz de las derivadas de x e y en z, y luego se normaliza el valor con la función “normalize” de GLSL. Luego se configuran algunos variables relacionadas con la iluminación de la superficie del fluido: la dirección de la luz, el vector v y h inherentes al cálculo del sombreado Phong y la especular. En los píxeles que no forman parte de la superficie del fluido se deshabilita el valor especular.

El siguiente paso en el cálculo del color real del píxel es el cálculo de la reflexión del entorno, en este caso las reflexiones estáticas basadas en mapeo de entorno cúbico y un término de Fresnel el cual depende del ángulo de visión usado sobre la superficie del agua. El vector de reflexión es fácilmente calculado con la función de reflexión de GLSL basada en la dirección del vector vista y la normal de la superficie. Nótese que el vector reflexión resultante ha sido transformado a espacio mundo porque este es el espacio nativo del mapeo de entorno.

A continuación se hace un muestreo con coordenadas enteras de la textura de grosor obtenida del shader "Thickness" y se agrega el valor de algunos otros parámetros del fluido como el color base del fluido y el coeficiente de atenuación.

Luego se calcula el color con la refracción en cada fragmento. La refracción se calcula perturbando las coordenadas de texturas del fragmento por la normal del fluido, con una magnitud correspondiente al grosor antes calculado. Este valor se compone luego con el color del fluido, el término de fresnel y la luz especular que también han sido calculadas con anterioridad.

Finalmente se determina el tipo de despliegue que ha de hacerse de acuerdo a lo elegido desde la interfaz (refracción, especular, coeficiente de fresnel, normales, alpha, grosor del fluido o el despliegue completo).

- **getEyeSpacePos:** Dadas las coordenadas de textura del fragmento se calcula la posición del fragmento en coordenadas de vista, haciendo uso de la función "uvToEye".

- **uvToEye:** Recibe un valor de profundidad en coordenadas de pantalla y lo transforma a una posición aproximada en coordenadas de vista para el fragmento, utilizando la longitud focal de la cámara y las coordenadas de textura *(u,v)* del fragmento.

h) Rain

Verifica si una partícula se encuentra por debajo del umbral de densidad y dado el caso modifica la partícula de forma tal que pueda ser desplegada como una gota de lluvia, en lugar de mantener su forma esférica propicia para el "Splatting" utilizado en el despliegue del resto del fluido que es percibido como un continuo.

El *vertex shader* recibe el estado de la partícula (densidad, posición, velocidad y color), algunos de ellos como parámetros *"varying"* o *"uniform"* y otro haciendo uso de los diferentes canales de *multitexture*. Estos atributos son transmitidos al *geometry shader*.

En este caso el *geometry shader* procesa un solo vértice dado que la geometría con la que está trabajando es un sprite. De este sprite que representa la partícula de fluido, recibe el estado de: densidad, posición, posición previa, dirección y color. Finalmente crea cuatro vértices adicionales a la geometría, alrededor del vértice original. Estos vértices quedan desplazados en x e y de la posición original del sprite. El desplazamiento en x corresponde al tamaño de la partícula mientras que el desplazamiento en y es calculado a partir del tamaño de la partícula y la dirección de precipitación de la partícula, todo esto multiplicado por cuatro.

En el *fragment shader* se muestrea una de las texturas representadas por la Figura 41 para dar apariencia de una gota y se mezcla con el color asignado para el factor de atenuación. Por razones de simplicidad solo se eligió una de las texturas disponibles, sin embargo para mejorar el efecto visual se pueden usar varios tipos de texturas dependiendo de otros parámetros de lluvia o del ángulo de visión.

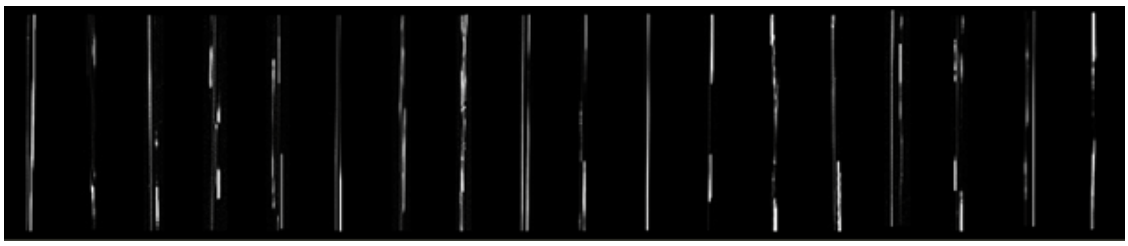


Figura 41: Diferentes texturas para la representación de una gota de agua de lluvia.

i) DebugTexture

Recibe una textura que ha de desplegar en pantalla. Este shader es utilizado para depurar los resultados intermedios de despliegue y para desplegar el render final producido por el shader "Rain".

CAPÍTULO VII. Pruebas de rendimiento

En este capítulo se detalla el conjunto de pruebas de rendimiento realizadas una vez terminada la etapa de implementación. La Tabla 8 describe las características de los equipos utilizados en las pruebas realizadas.

Equipo	Sistema Operativo	Procesador	Memoria RAM	Tarjeta gráfica	Memoria de Video
1	Windows 7	Intel Core i3 3.07GHz	4,0 GB	Nvidia GeForce GTX 550 Ti	1 GB
2	Windows 7	Intel Core i5 2.8GHz	4,0 GB	Nvidia GeForce GTX 460	768 MB
3	Windows 8	Inter Core i7 2.4 GHz	8,0 GB	Nvidia GTX 660M	2 GB

Tabla 8: Equipos utilizados en las pruebas de rendimiento.

1. Pruebas Cuantitativas

Para evaluar el rendimiento de la aplicación se realizaron una serie de pruebas midiendo la cantidad de *frames* por segundo (FPS) con diferentes configuraciones del sistema de techos y canaletas, diferentes parámetros para el despliegue del fluido y diversas configuraciones de iluminación y de intensidad de precipitación.

De esta forma se evaluó el comportamiento de la aplicación de acuerdo a: cantidad de partículas procesadas, complejidad en vértices de la(s) geometría(s) desplegadas, modelo de iluminación, tamaño del kernel de suavizado y número de iteraciones de suavizado.

Las configuraciones utilizadas buscan explotar tres aspectos que hacen denso el procesamiento. La complejidad de la geometría en el caso de la primera configuración asciende a los 32859 vértices. En la configuración 2 se utiliza la mayor cantidad de partículas de fluido, 600000 para una intensidad de precipitación de 200 mm/h. Finalmente la configuración 3 pone a prueba las opciones de despliegue, utilizando el mayor tamaño de kernel de suavizado (distancia 15) disponible y la mayor cantidad de iteraciones de suavizado (256 iteraciones).

En la Tabla 9 se aprecia en detalle cada una de las configuraciones utilizadas durante las pruebas de *frames* por segundo (FPS).

	Vértices	Partículas	Despliegue
Configuración 1	*Sistema de Techos* Techo inclinado de tejas 16x16 m. 32137 vértices. *Sistema de Canaletas* Semicircular central 30° inclinación. 722 vértices.	Intensidad de precipitación: 50 mm/h. 50000 partículas.	Sin iluminación. Tamaño de kernel de suavizado 2. 10 iteraciones de suavizado. DownSample habilitado.
Configuración 2	*Sistema de Techos* Techo a dos aguas corrugado tipo 4 10x12 m. 1268 vértices. *Sistema de Canaletas* Pecho Paloma derecho 22,6° inclinación. 256 vértices.	Intensidad de precipitación: 200 mm/h. 600000 partículas.	Iluminación Gouraud. Tamaño de kernel de suavizado 4,5. 20 iteraciones de suavizado. DownSample deshabilitado.
Configuración 3	*Sistema de Techos* Techo plano 6x6 m. 24 vértices.	Intensidad de precipitación: 125 mm/h. 250000 partículas.	Iluminación Phong. Tamaño de kernel de suavizado 15. 256 iteraciones de suavizado. DownSample habilitado.

Tabla 9: Configuraciones de prueba en *Frames* Por Segundo (FPS).

Para determinar el rendimiento en cada uno de los equipos de prueba por cada una de las configuraciones preestablecidas, se capturó la cantidad de FPS durante un minuto, obteniendo 60 valores que fueron posteriormente promediados. Esta operación se realizó tres veces por cada prueba para obtener resultados más precisos.

Estos valores se pueden apreciar en la Tabla 10. Nótese que solo aparecen tres valores por cada prueba (Configuración - Equipo), esto es debido a que la cantidad de *Frames* por Segundo capturados durante un minuto ya han sido promediados.

	Configuración 1	Configuración 2	Configuración 3
Equipo 1	13,41666667	7,983333333	19,03333333
	13,6	7,85	19,08333333
	13,76666667	7,816666667	19,15
Equipo 2	18,3166667	11,21666667	22,96666667
	18,31666667	11,4	22,73333333
	18,15	11,25	22,78333333
Equipo 3	11,83333333	7,05	11,13333333
	11,86666667	7,05	11,16666667
	11,55	7,183333333	11,18333333

Tabla 10: Resultados de prueba de FPS.

De acuerdo a lo esperado, los resultados obtenidos en cada prueba por cada combinación "Configuración – Equipo" son similares, al promediar estos valores obtenemos un resultado más preciso, el cual podemos apreciar en la Figura 42.

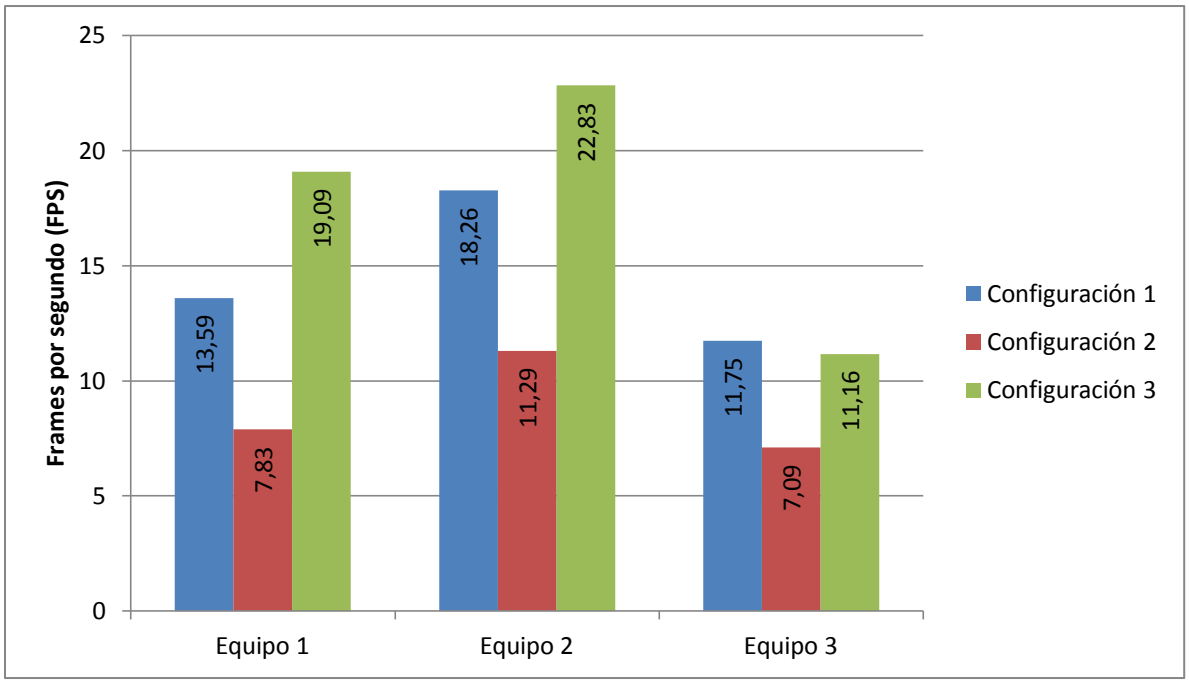


Figura 42: Resultados de las pruebas cualitativas efectuados en los diferentes equipos de prueba para cada una de las configuraciones.

De acuerdo a los resultados obtenidos, podemos notar que la configuración que se desempeñó con más alto rendimiento fue la "Configuración 3" en casi todos los equipos de prueba, a

excepción del “Equipo 3” cuya ventaja la tiene la “Configuración 1” por sólo unas pocas décimas.

A pesar de tener la iluminación Phong activada (la más costosa computacionalmente), el mayor tamaño de kernel de suavizado (tamaño 15) y realizar 256 iteraciones de suavizado, la cantidad de FPS obtenidos fue mayor que en las otras dos configuraciones. Una de las razones por las que esto podría haber ocurrido es porque la cantidad de vértices cargados para esa configuración son muy pocos (24 vértices), y en comparación con las otras dos configuraciones la diferencia es notable. Por ende, el arduo procesamiento que pudo haberse generado al ejecutarse el shader “Smooth”, se ve mermado por el hecho de que la cantidad de colisiones que tiene que procesar PhysX se hace contra una geometría muy sencilla y con una cantidad máxima de 250000 partículas, que aún cuando son una cantidad considerable, es menos de la mitad de lo máximo soportado por PhysX o de lo que se utiliza por ejemplo en la “Configuración 2”.

La “Configuración 1” presentó un rendimiento medio. A pesar de tener una cantidad de vértices muy por encima de las otras dos configuraciones (32859 vértices), sólo se generaba una cantidad máxima de 50000 partículas, no se activó ningún tipo de iluminación, y el tamaño de kernel y de iteraciones de suavizado estuvo por debajo de la configuración predeterminada (tamaño 2 de kernel de suavizado y sólo 10 iteraciones de suavizado). Todos estos factores permitieron que la mayor parte del procesamiento estuviera dedicado al procesamiento de los vértices de forma tal que el rendimiento no decayera demasiado, por ende la cantidad de FPS obtenida en todos los equipos fue razonable.

La “Configuración 2” fue la que obtuvo menor rendimiento en todos los equipos. La cantidad de vértices procesados era mayor que en la “Configuración 3” pero mucho menor que en la “Configuración 1” y las configuraciones de despliegue fueron las predeterminadas (tamaño de kernel de suavizado 4,5 y 20 iteraciones de suavizado), elegidas así para obtener un rendimiento medio, es decir, con un costo computacional moderado y sin sacrificar la calidad visual de la simulación. La iluminación Gouraud estuvo activada pero esta no representa un alto costo computacional. Podría decirse que lo que estuvo mermando el rendimiento fue la cantidad máxima de partículas, 600000 para esta configuración.

Finalmente un factor decisivo en el rendimiento fue la activación del shader “DownSample” en las configuraciones 1 y 3. Esta optimización permite que el tamaño del buffer de profundidad sea reducido a la mitad antes de aplicar el shader “Smooth”, con lo cual aplicar la técnica *Adaptative Curvature Flow Filtering* requiere de menos cómputo.

Esto explica el hecho de que el rendimiento de la “Configuración 2” sea menor y la diferencia en la cantidad de FPS sea mucho mayor con respecto a la diferencia existente entre las otras dos configuraciones. A pesar de que todas las configuraciones fueron definidas de manera equivalente, procurando explotar el rendimiento sólo en alguno de los elementos que requerían más procesamiento, la utilización del shader “DownSample” aventajó considerablemente a aquellas configuraciones que hacían uso de él.

Con respecto a los equipos de prueba, el equipo 3 fue el que presentó el peor rendimiento contrario a los pronósticos. A simple vista parecía ser el equipo con mayor capacidad debido al procesador, la memoria RAM y la memoria de video. Sin embargo, la mayor parte del procesamiento se hace a nivel de GPU y la tarjeta gráfica de este Equipo tiene algunas limitantes con respecto a las tarjetas gráficas de los otros dos equipos.

En principio el ancho de banda de la memoria (*Memory Bandwidth* en GB/seg) en el Equipo 3 es de 64.0 GB/seg, inferior a las del Equipo 1 (98.4 GB/seg) y el Equipo 2 (86.4 GB/seg). Esta característica está relacionada a la cantidad de información que puede ser transferida entre la memoria de video y el GPU y por ende afecta directamente a la velocidad de la memoria, desmejorando un poco el rendimiento a pesar de que la memoria de video y la capacidad del GPU puedan ser grandes.

Por otro lado, la cantidad de texels que pueden desplegarse en un segundo de acuerdo a la característica *Texture Fill Rate* (1000 millones/seg), corresponde al Equipo 3 a 30.4 mil millones por segundo, lo cual es menor a lo que puede alcanzarse con el Equipo 2 que corresponde a 37.8 mil millones por segundo. Esto es muy importante porque todo el proceso de despliegue se hace utilizando texturas.

2. Pruebas Cualitativas

A continuación podemos apreciar algunos de los resultados visuales obtenidos con la aplicación. Hay una serie de parámetros que contribuyen directamente en el proceso de despliegue de la escena con la intención de modificar la apariencia de las partículas del fluido (originalmente un *sprite*) y obtener así un fluido de agua con apariencia realista, tanto las partículas de agua que caen en forma de gotas de lluvia, como la masa de agua que se acumula como un flujo continuo sobre los techos y canaletas.

El cómo intervienen todos estos elementos en el proceso de despliegue fue previamente explicado en el CAPÍTULO V: Render y CAPÍTULO VI: Diseño e Implementación. Sin embargo, el valor que han de tener cada uno de estos parámetros para lograr buenos resultados visuales en un tiempo de procesamiento aceptable, ha sido fijado luego de varias pruebas.

a) Grosor del fluido y tamaño de las partículas

Ambos parámetros contribuyen al espesor de la masa del fluido, permitiendo que la capa que conforma el fluido al expandirse por una superficie se perciba más o menos gruesa, al igual que las gotas de agua al caer. Mientras más grande sea el tamaño de las partículas o mayor el valor de grosor del fluido, su color tiende a verse con más nitidez, llegando incluso a aumentar el valor de profundidad y oscurecerlo, como se aprecia en la Figura 43(D).

El no tener valores muy pequeños para estos parámetros ayuda a tener una mejor percepción del fluido como un continuo, tal como se aprecia en la Figura 43(A), en especial si la acumulación de las partículas es favorable y se ve desde un punto de vista cercano. Sin embargo, si las partículas están muy dispersas y la superficie por la cual se expanden es muy amplia y plana, la tendencia será a detectar la forma esférica de las partículas con mayor facilidad como se puede observar en la Figura 43(B).

Finalmente la Figura 43(C) nos muestra una captura de la simulación con los valores predeterminados para el grosor del fluido y el tamaño de las partículas en donde las partículas son lo suficientemente grandes para que el fluido pueda ser percibido como un continuo, pero tampoco demasiado grandes que intensifiquen de más la densidad del color.

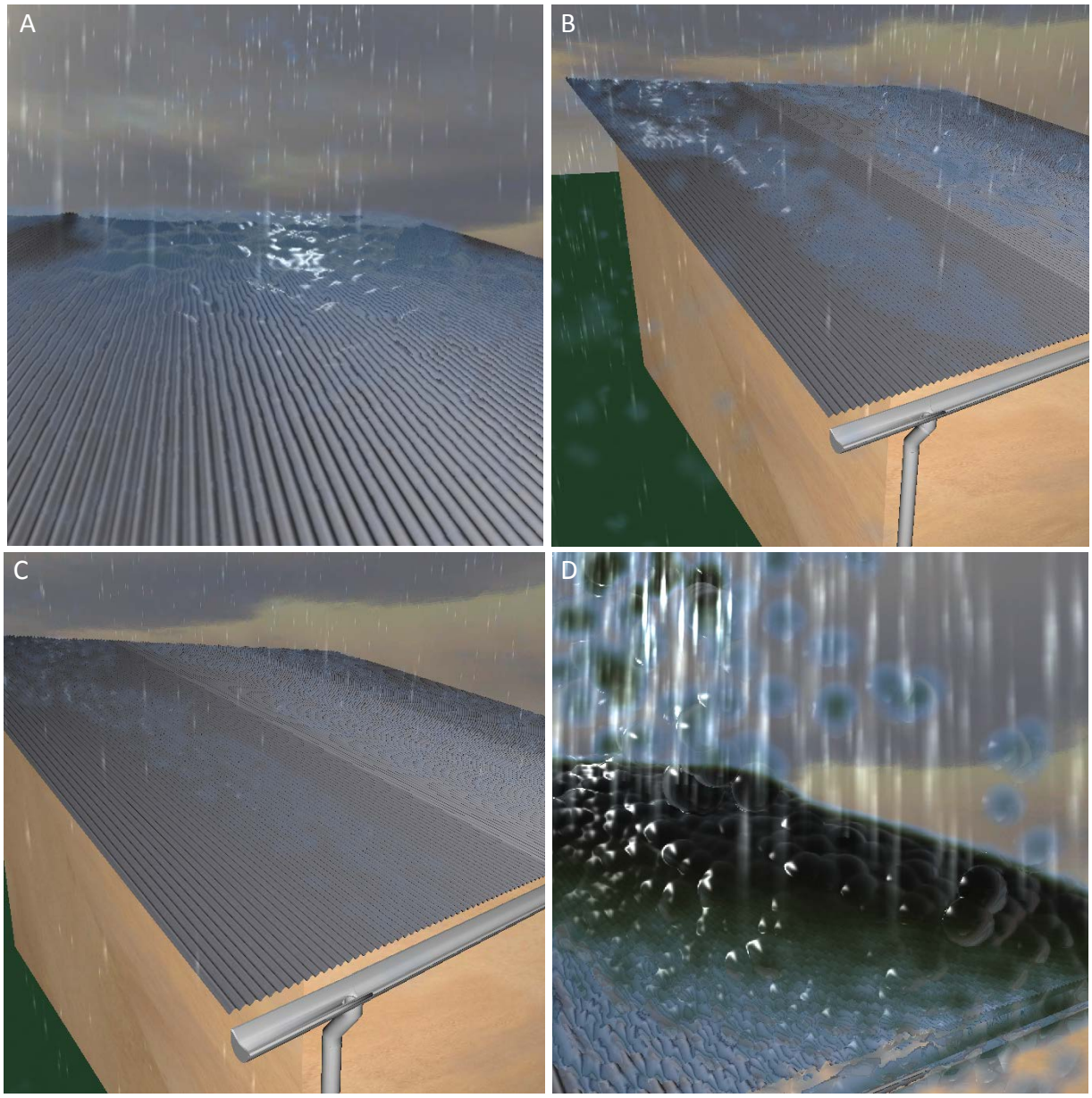


Figura 43: Pruebas visuales con diferentes valores de grosor del fluido y tamaño de partícula. (A) Acercamiento al fluido con grosor 1 y tamaño de partículas 2. (B) Vista semicompleta de la estructura con grosor del fluido 1 y tamaño de partículas 2. (C) Vista semicompleta de la estructura con configuración predeterminada (grosor 0,7 y tamaño de partículas 0,6). (D) Acercamiento con valor máximo de grosor de fluido (2,047) y de tamaño de partícula (5,128).

b) Espesor de refracción

Este parámetro controla la magnitud de refracción en las partículas del fluido. El valor asignado por defecto 0,4, es un valor aproximado para obtener un buen efecto de refracción en el fluido como se aprecia en la Figura 44(A). No conviene asignar valores demasiado altos porque el efecto puede ser exagerado, como se percibe en la Figura 44(B).

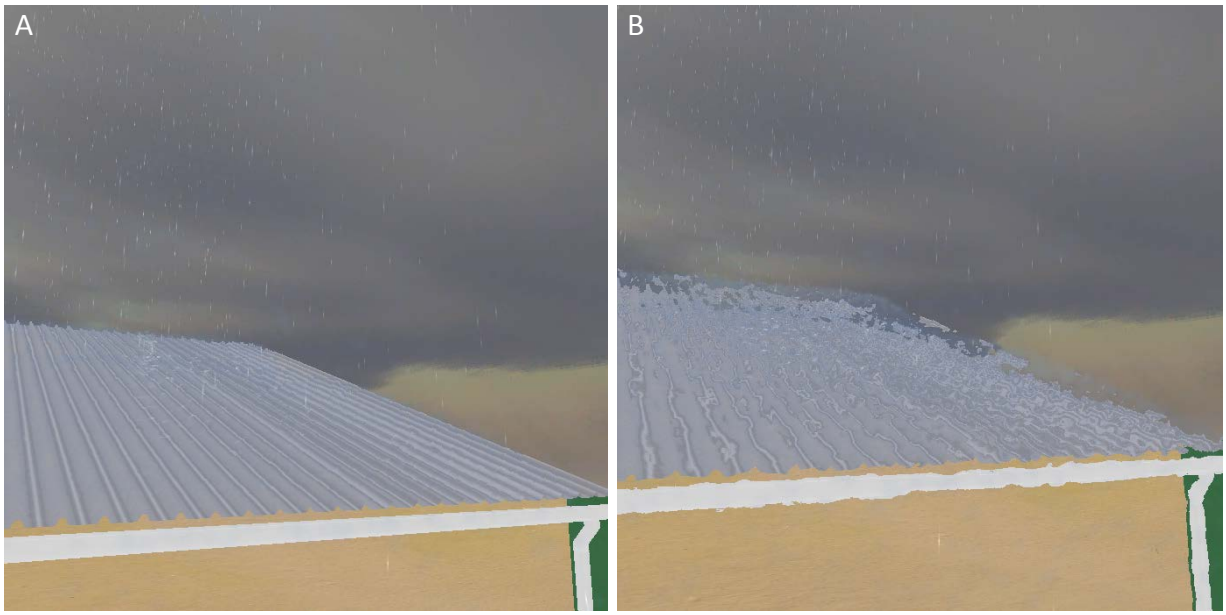


Figura 44: Capturas con diferentes valores de espesor de refracción. (A) Valor predeterminado 0,4. (B) Valor máximo 4,78.

c) Escala de atenuación

Este parámetro es utilizado en el cálculo del grosor y modifica el color base del fluido al atenuar su valor con el valor del color de atenuación. Por esta razón, el valor de escala de atenuación influye en la percepción de la transparencia del fluido. Mientras más grande sea la escala de atenuación la transparencia será menor. En la Figura 45(A) vemos la simulación con los valores por defecto, en donde las partículas de agua son apenas perceptibles, en la Figura 45(B) las partículas pueden apreciarse con más facilidad, sin embargo la apariencia de agua es totalmente irreal.

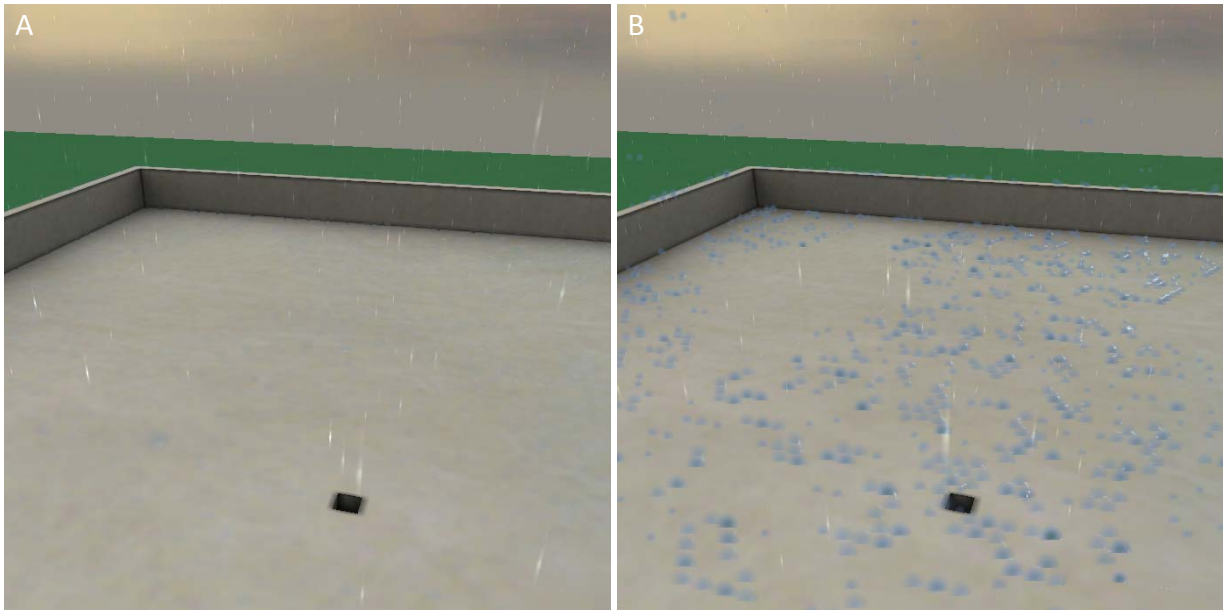


Figura 45: Capturas de la aplicación con diferentes valores para la escala de atenuación. (A) Valor predeterminado 0.03. (B) Valor máximo 0,67.

d) Brillo especular

Controla la cantidad de brillo especular presente en las partículas del fluido. A menor valor mayor será la componente de luz especular en el fluido. El color de la luz especular es por defecto el blanco, tal como se percibe en la Figura 46 sin embargo, es posible cambiar este valor desde la interfaz gráfica y obtener otros resultados visuales.

El valor de brillo especular para la aplicación está configurado por defecto en 80, lo cual permite percibir un brillo especular moderado tal como se observa en la Figura 46(A). Aún así es posible conseguir más especularidad disminuyendo el valor del brillo, como es el caso de la Figura 46(B). Finalmente en la Figura 46(C) se puede apreciar una captura en donde únicamente se despliegan los valores de brillo especular presente en las partículas.

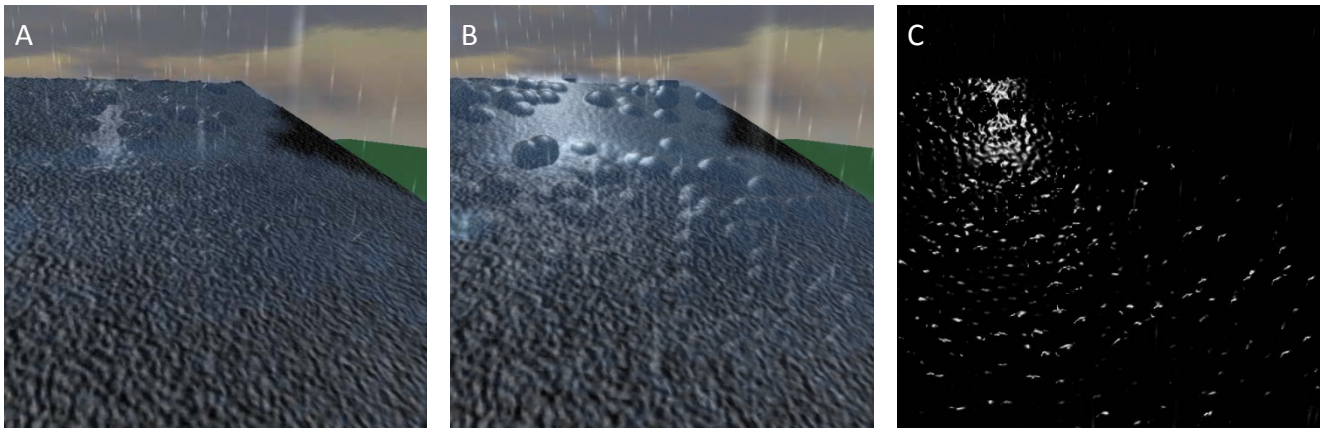


Figura 46: Captura de la aplicación con diferentes valores de brillo especular. (A) Valor por defecto 80. (B) Valor mínimo 10. (C) Despliegue de capa intermedia componente especular en el *pipeline* del *render* con valor por defecto 80.

e) Iteraciones de suavizado y tamaño de kernel de suavizado

Estos parámetros influyen directamente en la calidad visual que ha de proporcionar el *splatting* y la técnica *Adaptative Curvature Flow Filtering.* Mientras mayor sea el tamaño del kernel, mayor será la distancia de influencia del suavizado. Este parámetro tiene que ver con el umbral de profundidad en el que el *splatting* trabaja. La cantidad de veces que se aplicará el proceso de suavizado dependerá de la cantidad de iteraciones de suavizado elegidas por el usuario.

Por ende, estos dos parámetros controlan qué tanto ha de suavizarse la superficie del fluido, con lo cual mejorará la percepción visual de la misma, garantizando la sensación visual del fluido como un continuo.

En la Figura 47(A) y la Figura 47(B) se puede observar un suavizado moderado en la superficie al utilizar los parámetros predeterminados, cuyos valores corresponden a 4,5 el tamaño del kernel de suavizado y a 20 iteraciones de suavizado. En la Figura 47(C) y la Figura 47(D) se observa una superficie mucho más suavizada, donde las partículas casi no se perciben de manera individual sino que la superficie se aprecia bastante lisa, dando una sensación más realista al fluido al mostrarse como un continuo. En este caso los valores utilizados fueron los máximos, 15 para el tamaño de kernel y 256 iteraciones de suavizado.

Es importante destacar que aún cuando la calidad visual mejora conforme se incrementa el valor de estos dos parámetros, la capacidad de procesamiento es mermada por el alto costo

computacional que implican las operaciones que realiza el shader “Smooth” en la implementación del *splatting* y el suavizado con la técnica *Adaptative Curvature Flow Filtering.*

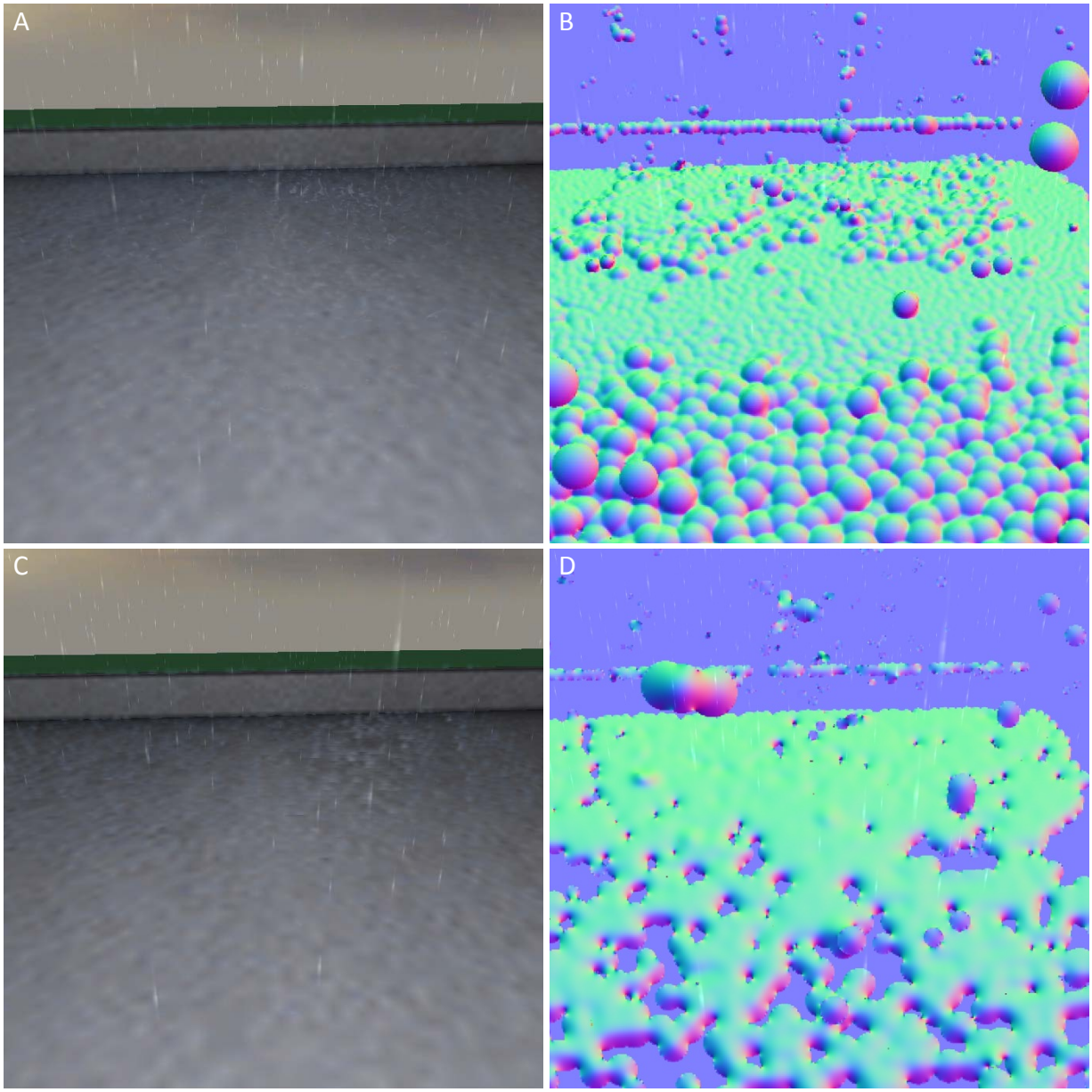


Figura 47: Capturas del fluido suavizado con diferentes valores de tamaño de kernel e iteraciones de suavizado. (A) Despliegue con parámetros predeterminados. (B) Despliegue de capa intermedia "normal" con valores predeterminados. (C) Despliegue con parámetros de suavizado en sus valores máximos. (D) Despliegue de capa intermedia "normal" con valores máximos.

f) Umbral de densidad

Este parámetro determina si una partícula ha de ser desplegada como una esfera o con una textura que asemeje la lluvia al caer. Si la densidad del fluido es grande en un espacio determinado, se asume que se trata de una masa de agua que se encuentra junta, por lo cual su apariencia debería ser la de una esfera para que luego pueda ser suavizada por el shader "Smooth". Sin embargo, si la densidad de una porción de fluido es baja, implica que las partículas en ese espacio determinado se encuentran muy separadas, y por ende se interpretan como gotas de agua aisladas, posiblemente cayendo en forma de lluvia y en este caso el shader "Rain" le aplica una textura.

El parámetro umbral de densidad define la relación de cantidad de partículas por metros cuadrados bajo la cual se rige la decisión de desplegar las partículas como esferas o con una textura.

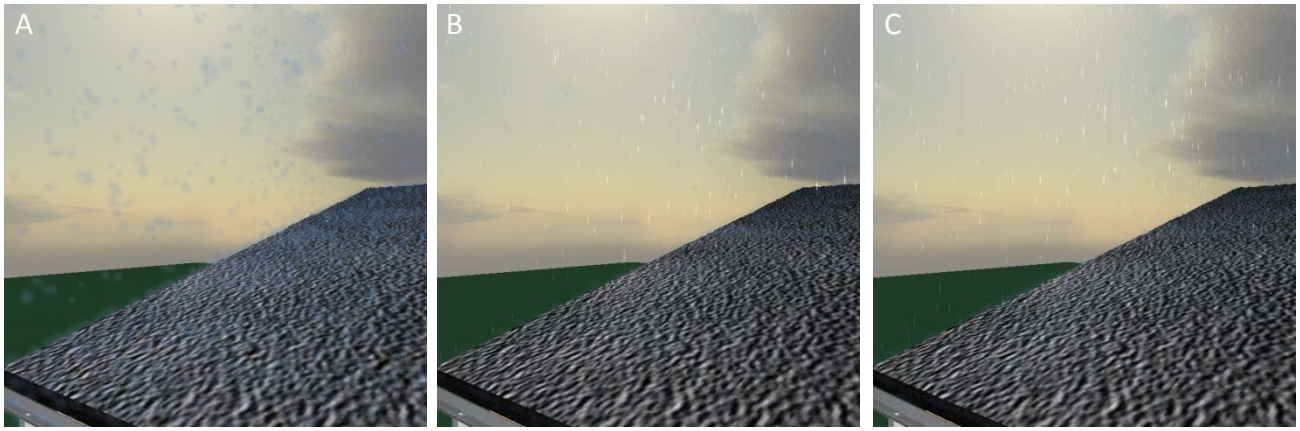


Figura 48: Captura de las partículas con diferentes valores de umbral de densidad. (A) Umbral de densidad mínimo 0.0. (B) Umbral de densidad predeterminado 200.0. (C) Umbral de densidad máximo 300.0.

En la Figura 48(A) se aprecia una captura de la simulación con el umbral de densidad en 0, con lo cual, las partículas se despliegan siempre como esferas. La Figura 48(B) despliega las partículas con el umbral predeterminado, en donde la mayoría de las partículas en el aire se ven como gotas de lluvia al caer, sin embargo se observa de vez en cuando alguna desplegada como esfera. Finalmente la Figura 48(C) utiliza el umbral de densidad máximo, y de esta forma todas las partículas que caen en forma de lluvia se ven como tal, sin embargo se encuentra una que otra partícula aislada que ya ha colisionado con la superficie del techo que sin embargo se sigue viendo con la textura de una gota de lluvia al caer.

g) Artefactos visuales

Durante el proceso de ajuste de parámetros predeterminados de despliegue hechos a partir de las pruebas cualitativas se detectaron algunas deficiencias visuales en algunos momentos del despliegue.

En la Figura 49, podemos apreciar a lo largo del segmento final, en especial en el extremo izquierdo del mismo, un artefacto visual en una partícula que atraviesa esta superficie. Es posible que las partículas se desborden visualmente, por el hecho de que conforme aumente el grosor o el tamaño de las partículas estas se verán mucho más grandes, aunque el cálculo de la colisión siga siendo calculado por PhysX a partir de la estructura que representa a la partícula (en este caso un sprite) cuya posición corresponde al centro de la partícula y su dimensión es simplemente ese punto.

Sin embargo, no debería percibirse este desbordamiento puesto que la escena se despliega antes que el fluido y al hacerlo se determina el Z-buffer, el cual es consultado por el shader "Depth" para poder dibujar los sprites como esferas. Cuando lo hace, modifica el valor de profundidad si la partícula se encuentra más cercana al viewport, pero si hay un elemento de la escena delante de la partícula este valor no se modifica, por lo que estos fragmentos del fluido no deberían desplegarse.

El problema ocurre cuando la superficie del objeto en la escena no se interpone entre el punto de vista y la partícula, pero aún así la atraviesa. En este caso el sprite será modificado como una esfera y la partícula será desplegada atravesando la superficie.

Figura 49: Artefacto visual de desbordamiento de partícula.

En la Figura 50 se puede observar que las partículas del fluido se acumulan como parches cuadrados sobre la superficie del techo cuando debería expandirse por toda la superficie de manera uniforme.

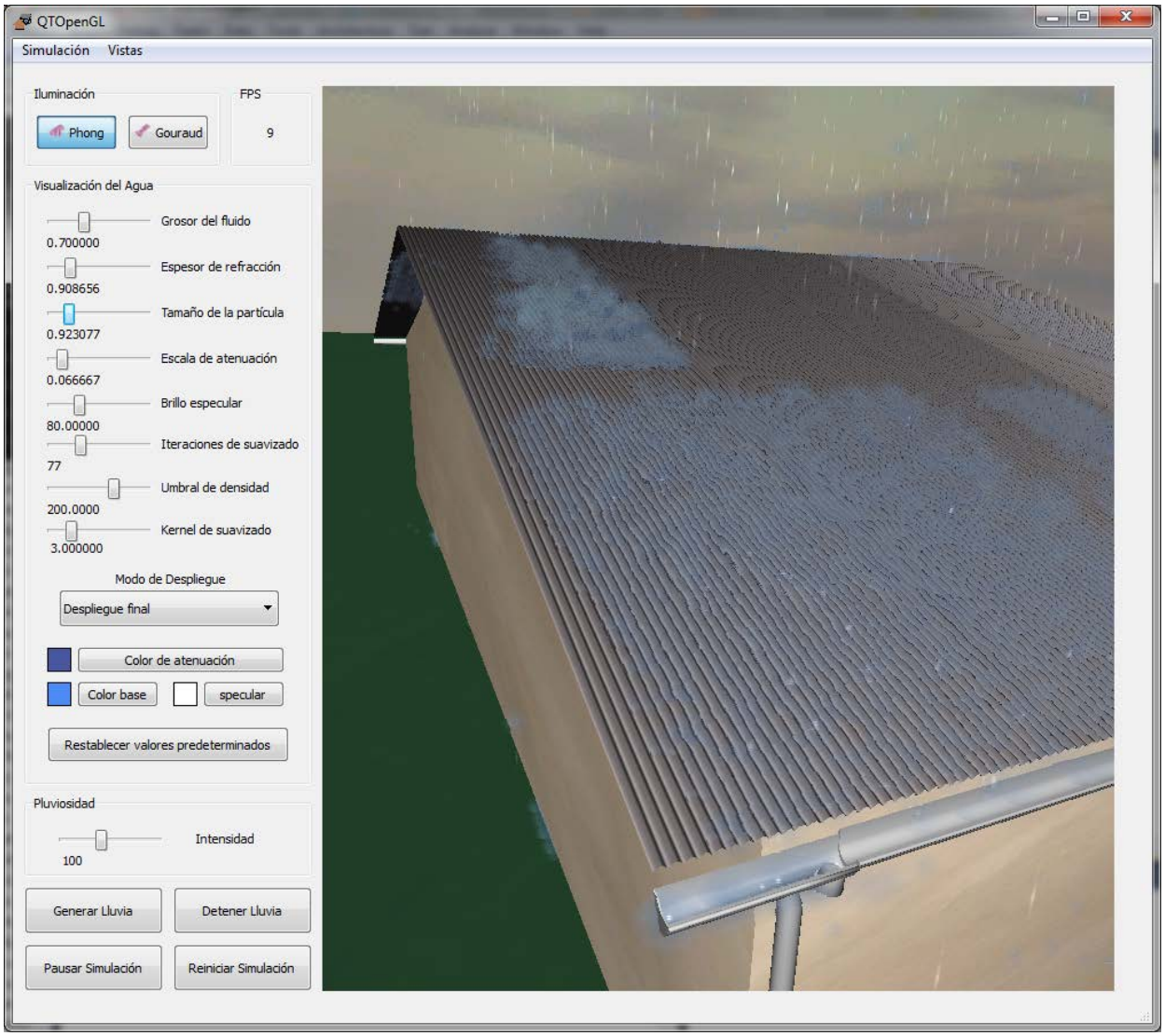


Figura 50: Artefacto visual de parches de fluido en el despliegue.

Esto ocurre por la distribución alternada de los emisores para generar el fluido. Fue necesario crear varios emisores para el soporte de gran cantidad de partículas, lo cual no podría haberse hecho con un solo emisor. Sin embargo esto crea también varios paquetes de fluido que no siempre pueden agruparse y se generan artefactos de este tipo.

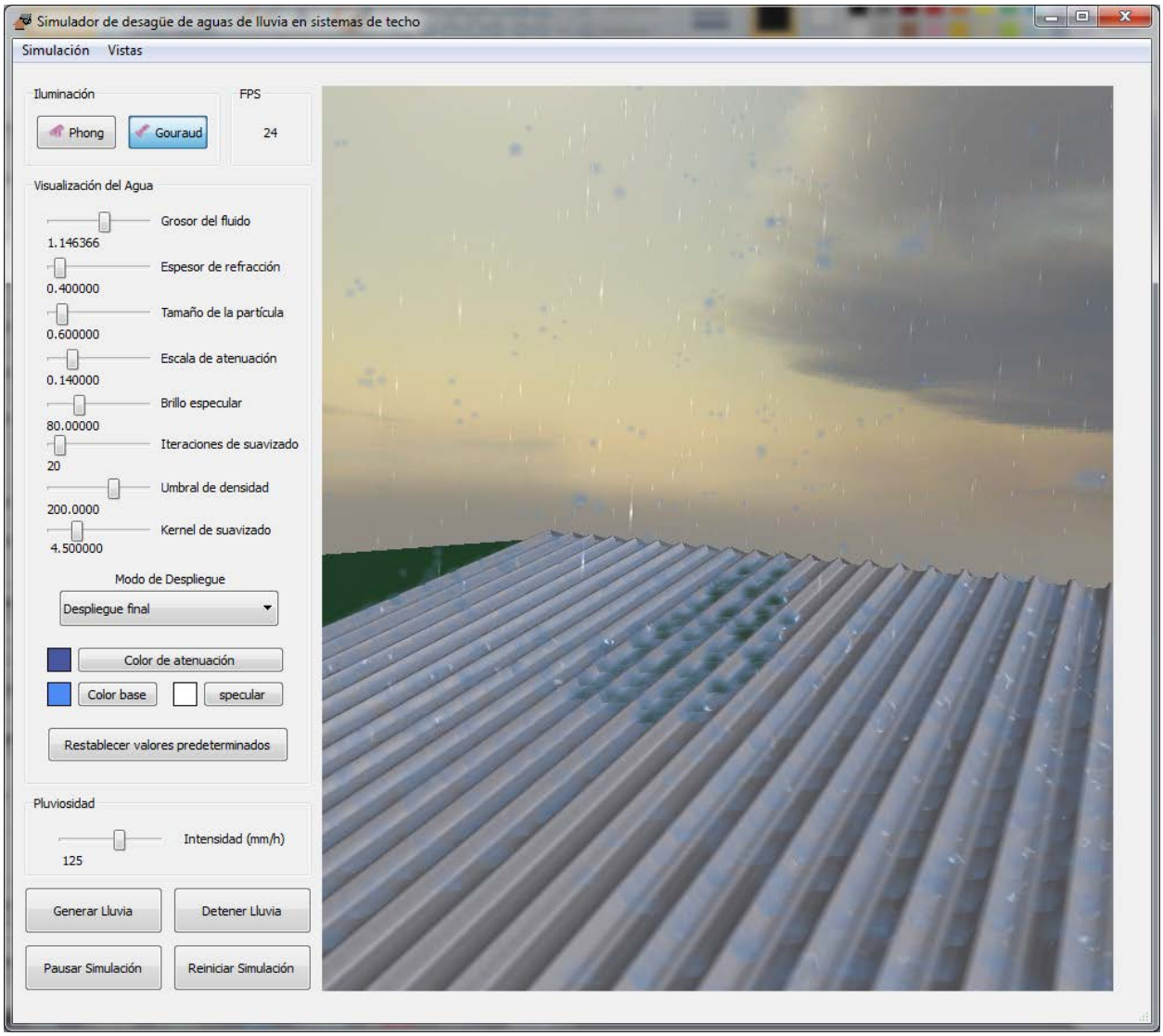


Figura 51: *Flickering* en algunas zonas del fluido.

Finalmente se puede observar un último artefacto en la Figura 51 que genera un oscurecimiento intermitente en algunas zonas del fluido. La intermitencia es debido al comportamiento natural del *Z-fighting*, un fenómeno que ocurre en el despliegue 3D cuando dos o más primitivas tienen valores similares en el *Z-buffer,* desplegando así fragmentos pertenecientes a una u otra primitiva de manera arbitraria [21]. La razón del oscurecimiento es un error en el cálculo del espesor del fluido, que ocurre cuando hay una capa de aire intermedia entre dos capas de fluido. Cuando el shader "Composite" calcula el color final en esos fragmentos, suma al cálculo completo del espesor del fluido el color de la capa frontal de agua, con lo que el color final se va oscureciendo de manera imprecisa.

CAPÍTULO VIII. Conclusiones y trabajos futuros

En este Trabajo Especial de Grado se pudo desarrollar una aplicación para la simulación del desagüe de aguas de lluvia en un sistema de techos y canaletas, a partir de una diversidad de opciones de techos y drenajes regidos por un conjunto de normas y parámetros arquitectónicos que contemplan soluciones para diferentes intensidades de precipitación en diversos tamaños de techos.

Efectivamente el SDK de PhysX facilitó muchos de los procesos en cuanto a la simulación del comportamiento físico del agua de lluvia, variando la cantidad de partículas de fluido utilizadas para simular las diversas intensidades de precipitación, así como realizando el cálculo de las colisiones generadas entre el fluido y los cuerpos rígidos (sistema de techos y canaletas). Por otro lado, la creación de partículas de fluido a partir de emisores permitió simular de forma adecuada la precipitación a partir de la condensación del agua en forma de nubes, y su desaparición a partir de "*Drains*" emula el fenómeno de percolación del agua a través de la tierra.

Se logró desarrollar una interfaz gráfica utilizando el *framework* QT 4.7, que permite la interacción del usuario con la aplicación para la creación del sistema de techos y canaletas, generar una precipitación ajustando los valores de intensidad, posicionar la cámara en vistas predeterminadas, editar los valores de los parámetros de despliegue, cargar y guardar escenas predefinidas a partir de un archivo de texto y ejecutar pruebas de rendimiento. Así mismo, es posible navegar por la escena para enfocar mejor algún área de interés.

Se hicieron diversas pruebas de rendimiento que permitieron vislumbrar el alcance del SDK de PhysX en cuanto a la generación de partículas de fluido, así como el alcance de diversos equipos en el procesamiento de vértices, cálculos de despliegue y soporte al desempeño de PhysX en el cálculo de la física.

De acuerdo a nuestras pruebas y los resultados obtenidos, los elementos cuyos valores tienen mayor impacto sobre el rendimiento de la aplicación desarrollada son: la cantidad de partículas del fluido que son generadas, la cantidad de vértices de los modelos en la escena y la activación del shader "DownSample" del buffer de profundidad. En el caso de las partículas y los vértices, a medida que aumenta su valor, disminuyen los FPS de la aplicación. En el caso del shader "DownSample", si se encuentra activo se incrementan los FPS de la aplicación.

Por otro lado, el proceso de *Adaptative Curvature Flow Filtering*, realizado por el shader "Smooth" tiene un alto costo computacional, especialmente cuando el valor del kernel de suavizado y la cantidad de iteraciones aumenta, a pesar de que con ello se logran mejores resultados visuales.

Así mismo, las pruebas permitieron definir los valores de los parámetros de despliegue que mejor se ajusten para garantizar un equilibrio entre la calidad de visual del fluido y la complejidad de cálculo. Los parámetros más relevantes son:

- Grosor del Fluido: 0,7.
- Tamaño de las partículas: 0,6.
- Espesor de refracción: 0,4.
- Escala de atenuación: 0,03.
- Brillo especular: 80.
- Iteraciones de suavizado: 20.
- Tamaño de kernel de suavizado: 4,5.
- Umbral de densidad: 200.

A continuación se presentan algunas posibles mejoras a la aplicación desarrollada para dar más alcance al trabajo en cuanto a la simulación:

- Incluir la simulación de viento durante las precipitaciones, para lograr una simulación más fiel a la realidad.
- Añadir la posibilidad de construir estructuras con sistemas de techo más complejos.
- Agregar otros efectos visuales como cáustica para dar más realismo al fluido.
- Desarrollar una mejor distribución de los emisores para que la generación de partículas de fluido cubra más áreas de extensión y al mismo tiempo permita que las partículas no se dispersen demasiado.
- Voxelizar el fluido para obtener una composición más exacta en cuanto al cálculo del grosor del fluido para solucionar los artefactos producto de múltiples capas de fluido.
- Estudiar la posibilidad de utilizar el stencil buffer para solucionar el artefacto del desbordamiento de partículas.
- Implementar el *frustum culling* tanto para los cuerpos rígidos como para el fluido con la intención de mejorar el rendimiento de la aplicación.
- Adaptar la aplicación a la versión más reciente de PhysX para comparar el rendimiento de la misma con una versión más actualizada de este motor de física.

- Implementar una versión de la aplicación utilizando otro motor de física y otro *framework* para el desarrollo de la interfaz gráfica, con el fin de comparar el rendimiento de dicha aplicación con la de la aplicación desarrollada en este trabajo. Esto nos proporcionaría un indicativo de si nuestra elección de utilizar PhysX y QT fue la más adecuada.

Referencias

[1] Corporation, NVIDIA, «Physics SDK API Reference», [En línea]. Disponible en: www.nvidia.com/.

[2] N. Corporation, 2008 - 2011. [En línea]. Disponible en: http://doc.qt.digia.com/qt/index.html.

[3] R. Serway y J. Jewett, Física para ciencias e ingeniería, vol. I, Thomson, 2005.

[4] P. A. Sturrock, «Plasma Physics: An Introduction to the Theory of Astrophysical», McGrawHill, 1994.

[5] P. A. Tipler, Física, Barcelona: Ed. Reverté S.A., 1983.

[6] D. Nguyen, R. Fedkiw y H. Wann, «Physically Based Modeling and Animation of Fire» en Proceedings of SIGGRAPH, pp. 129–136, 1995.

[7] R. Bridson y M. Müller, «Fluid Simulation» en *SIGGRAPH 2007 Course Notes*, 2007.

[8] M. Müller, D. Charypar y M. Gross, «Particle-Based Fluid Simulation for Interactive Applications» en *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association*, Aire-la-Ville, Suiza, 2003.

[9] L. Cueto, «Partículas, volúmenes finitos y mallas no estructuradas: Simulación numérica de problemas de dinámica de fluidos» La Coruña, Marzo, 2005.

[10] E. Araujo, «Teorema del valor de la densidad de contacto de fluidos en nanoporos», Merida, Venezuela, 2008.

[11] P. Gourou y L. Papy, Compendio de Geografía General, RIALP, 1967.

[12] Organización Meteorológica Mundial, «Manual de observación de nubes y otros meteoros», *Atlas Internacional de Nubes,* vol. I, nº 407, Ginebra 1993.

[13] R. Pizarro, J. Pizarro, C. Sangüesa y E. Martínez, «Módulo 2: Curvas Itensidad Duración Frecuencia» en *Sociendad Estándares de Ingeniería para Aguas y Suelos LTDA*, 2003.

[14] M. Martelo, «La Precipitación en Venezuela y su relación con el Sistema Climático», Caracas, Mayo, 2003.

[15] D. da Silva,. G. de Sousa, M. Silva, V. Teichrieb, J. Kelner, T. Cordeiro, «SBGAMES 2006 TUTORIAL AGEIA PHYSX», Pernambuco, Brasil, 2006.

[16] L. A. López R., AGUA - Instalaciones Sanitarias en los Edificios, Maracay: Betanzos, 1990.

[17] F. Ching y C. Adams, Guía de Construcción Ilustrada, México: Limusa, 2006.

[18] R. Manning, «On the flow of water», 1891.

[19] PAVCO de Venezuela, «Manual: Canales y bajantes para aguas de lluvia», Septiembre 2009.

[20] J. Hoberock y Y. Jia, «High-Quality Ambient Occlusion» en *GPU Gems 3*, H. Nguyen, Ed., Addison Wesley, 2008.

[21] T. Akenine-Möller, E. Haines y N. Hoffman, Real Time Rendering, Tercera ed., Massachusetts: A K Peters, Ltd., 2008.

[22] F. Bagar, «A Layered Particle-Based Fluid Model for Real-Time Rendering of Water» en Eurographics Symposium on Rendering 2010. Junio 2010.

[23] M. Zwicker, H. Pfister, J. van Baar y G. M., «Surface Splatting» en *SIGGRAPH*, 2001.

[24] M. Botsch, A. Hornung, M. Zwicker y L. Kobbelt, «High-Quality Surface Splatting on Today's GPUs» en *Eurographics Symposium on Point-Based Graphics*, 2005.

[25] W. van der Laan, S. Green y M. Sainz, «Screen space fluid rendering with curvature flow» en *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, New York, EE.UU., 2009.

[26] B. Peters y M. Schäf. [En línea]. Disponible en: http://www.mpi-inf.mpg.de/departments/irg3/ws0304/lcn/projects/Martin/index.htm.

[27] A. Baldor, Geometría plana y del espacio y trigonometría, 2004.