UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SÍNTESIS DE TEXTURAS PROCEDURALES PARA UN SIMULADOR COMPUTARIZADO DE CIRUGÍA DE PRÓSTATA

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A :
GABRIELA GARCÍA GUEVARA

DIRECTOR DE LA TESIS:
M. C. MIGUEL ÁNGEL PADILLA CASTAÑEDA

CO-DIRECTOR DE LA TESIS:
DR. FERNANDO ARÁMBULA COSÍO

MÉXICO D.F. AÑO 2009

DEDICÓ ESTA TESIS A:

Mis padres:
Raquel Guevara *y* ***Efraín G. García****, que con su amor, cariño, paciencia, ejemplo y apoyo incondicional se ve culminada una de mis grandes metas.*

Mi hermana:
Angélica*, por ser mi amiga y compañera de toda mi vida.*

Mi tía:
Vange*, como un testimonio de cariño por estar siempre al pendiente de mí. Dándome consejos, palabras de aliento y superación.*

Mi familia:
A mis ***abuelos, tíos y primos****, por todas sus muestras de afecto y cariño; mejor familia no podía tener.*

AGRADECIMIENTOS

Como un testimonio de gratitud y eterno reconocimiento a ***M.C. Miguel Ángel Padilla Castañeda,*** *por su valiosa dirección e interés en la culminación de este proyecto de tesis; y que a pesar de nuestros distintos compromisos y actividades siempre me apoyo y alentó a concluir mis estudios.*

Agradezco al ***Dr. Fernando Arámbula Cosío*** *por su apoyo, ayuda y colaboración en la finalización de este trabajo.*

Mi más sincero reconocimiento al apoyo y la colaboración de todos los integrantes del equipo del Simulador RTU, que gracias al trabajo de todos, se ha ido mejorando dicho proyecto. Y en especial a ***Sergio Teodoro Vite*** *por sus consejos y su gran ayuda que recibí al final, en la incorporación de mi trabajo de tesis al simulador.*

También agradezco al laboratorio de ***Análisis de Imágenes y Visualización del CCADET, UNAM****. Por las facilidades otorgadas para la realización de este trabajo.*

Sin olvidar mi más sincero agradecimiento a mis amigos de la facultad: ***Fer, Erika, Guillermo, Juan Carlos y Argelia*** *por ayudarme y estar conmigo a lo largo de la carrera.*

En general agradezco a todas las personas que de manera desinteresada me brindaron su ayuda e influyeron positivamente para la terminación de este trabajo.

Finalmente agradezco enormemente a la ***UNAM****, y en especial a la* ***Facultad de Ingeniería,*** *por proporcionarme las bases y las herramientas de mi carrera profesional y poder llegar a ser lo que soy. ¡Gracias!*

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

ÍNDICE DE TABLAS

CAPÍTULO 4

RESUMEN

En este trabajo de tesis se presenta el desarrollo de un modelo de síntesis de texturas procedurales, para ser utilizado en un simulador virtual de entrenamiento de resección transuretral de próstata (RTU) diseñado en el Laboratorio de Análisis de Imágenes y Visualización CCADET UNAM, con la finalidad de conseguir una textura de tejido lo más parecido visualmente al tejido mostrado en imágenes médicas reales de dicho procedimiento.

La obtención y generación de las texturas procedurales se puede dividir en: *textura del tejido de la uretra (vascularidades), textura del tejido interno de la próstata (apariencia amarillo-pardo) y textura del tejido de la cápsula de la próstata (apariencia rugosa)*. Para posteriormente obtener una textura sólida procedural 3D del modelo anatómico de la próstata; la cual se obtiene con la composición de las tres texturas antes mencionadas, colocadas de acuerdo a la zona que le corresponde dentro del modelo anatómico (textura del tejido de la cápsula, textura del tejido de la uretra y textura del tejido interno).

Las texturas procedurales se desarrollaron a partir de funciones primitivas de patrones procedurales, con uso de funciones de ruido para la aleatoriedad, lo que nos proporcionó un aspecto más realista; con el uso de conceptos aplicables a los modelos fractales para la repetición de patrones, para obtener formas más naturales; y con uso de funciones trigonométricas y aritméticas con las cuales se pudo conseguir diferentes efectos sobre una misma función procedural. Una vez contando con todas las funciones y características antes mencionadas, se obtuvo una herramienta de software para la generación de dichas texturas en 2D y 3D de manera interactiva y rápida para el simulador de cirugía.

Los resultados obtenidos con este proceso de generación de texturas, lograron conseguir visualmente el efecto de continuidad sobre el modelo del simulador de cirugía, permitiendo realizar cortes sobre el tejido de la próstata sin perder las características y efectos visuales. Además de contar con una textura de vascularidades, lo que da pie a que en un futuro se lleve a cabo la implementación de la simulación de sangrado, basándose en la superficie de la textura vascular disparando dicho efecto cuando se corte una vena inflamada o un vaso sanguíneo de la textura.

Finalmente se obtuvieron las funciones y procedimientos de generación de texturas procedurales capaces de ser aplicables a diferentes modelos virtuales.

CAPÍTULO 1

1. INTRODUCCIÓN

En el presente capítulo se comenzará explicando el concepto de textura. Se mencionarán brevemente los medios con los cuales se puede obtener una textura sintética, la cual suele ser muy útil, en el texturizado de modelos virtuales generados por computadora, dando origen a superficies visualmente muy parecidas a las que encontramos en la realidad.

Así mismo se plantearán los objetivos de este trabajo de tesis, que consiste en la realización de una síntesis de texturas procedurales para un simulador de entrenamiento de resección transuretral de próstata, con la cual se desea obtener una apariencia lo más parecido visualmente al tejido real del procedimiento antes mencionado.

1.1 Texturas

Las formas que frecuentemente encontramos en la naturaleza no muestran regiones con una intensidad de color o material uniforme. Por el contario, suelen encontrarse con una rica y muy variada colección de texturas.

La superficie de cualquier objeto visible tendrá una cierta escala de textura. La cual se suele describir como suave, áspera, blanda, dura, gruesa, mate o brillante, etc.; patrones naturales que a menudo tienen una cualidad táctil.

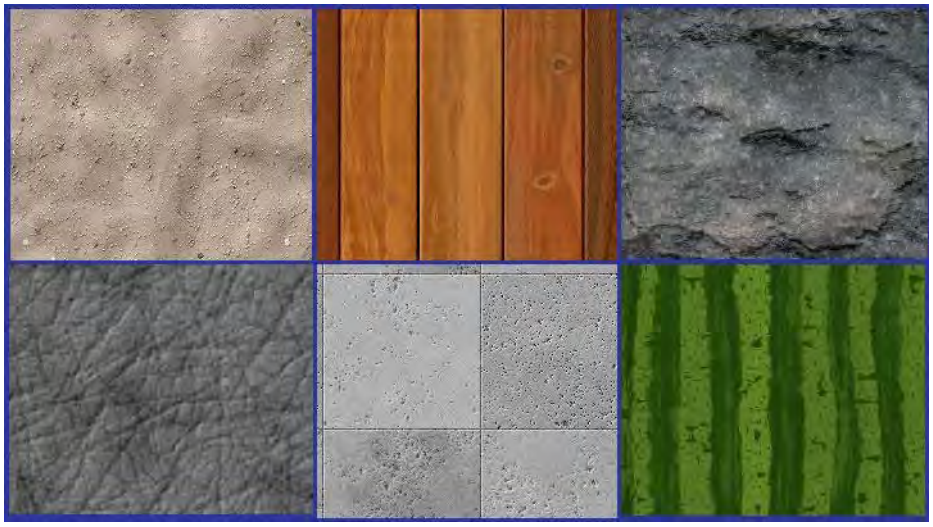
En términos generales, la textura se refiere a la apariencia y característica de la superficie de un objeto dado por el tamaño, forma, densidad, disposición y la proporción de sus partes elementales. Esta propiedad se relaciona enormemente con la reflexión y absorción de la luz sobre la superficie del objeto, es decir, absorben y reflejan colores. La textura, por lo tanto, es una cualidad diferencial que ayuda a distinguir y reconocer los objetos, enriqueciendo la apariencia de los mismos. *(Chen, et al., 1998)*

Una clasificación utilizada en el arte para identificar los tipos de texturas es la siguiente:

- ***Naturales:*** son las texturas que pertenecen a elementos de la naturaleza, como la piel de un elefante o la superficie de un pétalo de rosa.
- ***Artificiales:*** son las texturas de materiales fabricados por el hombre, como una pared o un papel de regalo.
- ***Táctiles:*** son las texturas que se percibe a través de la vista y del tacto, generalmente poseen relieve. Y son tridimensionales.
- ***Visuales:*** son las texturas que se perciben a través de la vista y no tienen relieve. Son bidimensionales y lo que intentan, a menudo, es imitar las texturas de los objetos reales para dar la sensación de realismo. Por ello también se les conoce como *texturas sintéticas*, existiendo diversos procedimientos para la obtención de estas texturas. *(Ahearn, 2006)*

1.1.1 Texturas sintéticas

Aprovechando las cualidades que nos dan nuestros sentidos, principalmente el del tacto y la vista. Es posible hacer simulaciones de diferentes texturas causando al espectador una apariencia "real" de la misma. Esta puede estar pintada, fotografiada, escaneada o generada por funciones o algoritmos por computadora. En la figura 1.1 se muestran diferentes texturas generadas por computadora, con las cuales es posible lograr efectos visuales muy interesantes aplicándolas a superficies de objetos tridimensionales.



de que en el proceso de escaneo de la imagen puede llegar a presentar deformaciones sobre la textura final que obtenemos, que generalmente no es lo que se desea.

Las formas de obtención de textura previamente mencionadas se refieren a imágenes en dos dimensiones, por lo cual podemos tener el problema de su limitada resolución, es decir, si se acerca demasiado a la imagen, puede presentar falta de detalles o incluso hacer visible los píxeles[1] originales de la textura.

Otro problema con este tipo de adquisición de textura, es que al mapearla sobre la superficie de nuestro objeto, puede presentar costuras o una repetición notable sobre el texturizado final. *(Apodaca, et al., 1999 págs. 243-245)*

Muchos de estos inconvenientes se corrigen con el uso de técnicas de generación de textura por medio de una computadora, las cuales se generan por funciones o formulas predefinidas.

Algunas de las ventajas de utilizar la generación de texturas por medios computarizados, es que pueden presentar los suficientes detalles a varias escalas sin tener distorsiones. Otra ventaja es que con estas texturas se pueden cubrir grandes áreas sin una repetición notable en el resultado final del mapeo.

Otra ventaja importante de las texturas generadas, es que se puede texturizar una modelo tridimensional, obteniendo una textura sólida de tres dimensiones, y al ser mapeadas a las superficies, podremos obtener vistas transversales de los modelos texturizados, dándonos la apariencia de continuidad del material como sucede en la realidad. *(Foley, et al., 1996)*

La mejor forma de obtener una textura dependerá en gran medida del resultado final que queramos obtener sobre el mapeado de nuestra superficie, así como las formas de adquisición de textura que tengamos más factible.

[1] **Pixel:** Unidad menor homogénea en color que forma parte de una imagen digital. Ampliando lo suficiente la imagen (zoom), puede observarse los píxeles que la componen.

1.3 Aplicación de textura sintética a un modelo computarizado

Como se menciono en el punto anterior, en la actualidad encontramos principalmente dos enfoques para la obtención de texturas para ser utilizadas en un modelo generado por computadora.

- El primer enfoque consiste en adquirir una textura plana la cual se define en un espacio de dos dimensiones, y puede ser obtenida de cualquiera de estas fuentes: parte de un dibujo, una imagen escaneada, o una fotografía digital.

- El segundo enfoque es obtener una textura sólida la cual se define en un espacio de dos o tres dimensiones por una composición de funciones matemáticas generadas por computadora. *(Turk, 1991)*

1.3.1 Textura plana

Se considera que la obtención de estas texturas puede hacerse muy rápidamente, aunque el resultado no siempre puede ser lo suficientemente bueno. Si un objeto es pequeño o no es un elemento central de la escena, una textura plana puede añadir bastante detalle. Pero en los objetos de mayor tamaño, dichas texturas probablemente tengan un relieve incorrecto.

El único requerimiento de la textura plana es que debe permitir el acceso a los distintos valores de los pixeles en la imagen. La textura debe almacenarse en una matriz bidimensional de tamaño T_u X T_v en donde a cada uno de los elementos de esta matriz de textura se le llama Texel[2].

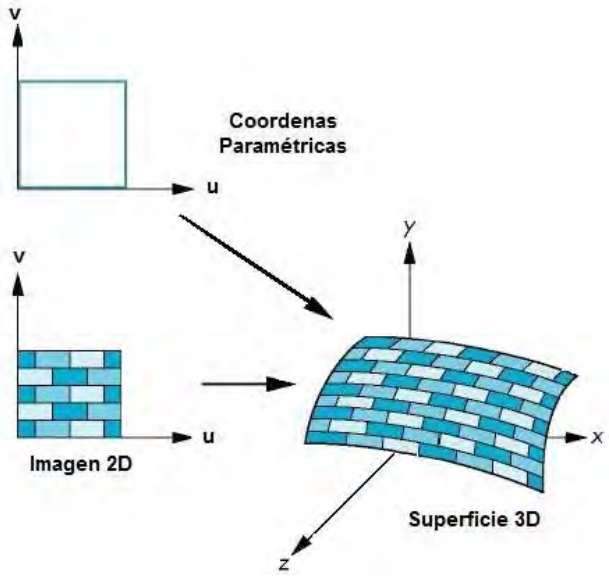
Para pintar la textura sobre el objeto, es necesario que haya una relación entre las coordenadas de la textura con respecto a las del objeto. Es decir, debe existir una función de transformación de los puntos de un objeto dentro del rango de coordenadas de la textura.

Los valores de la textura se accede por medio de una matriz mediante una mapeo de las funciones (x,y) de los componentes del objeto y los índices (u,v) de la textura. Con lo cual se puede hacer una analogía y concluir que una textura de dos dimensiones es "pintada" sobre una superficie (ver figura 1.2) *(McConnell, 2006)*

[2] **Texel:** Unidad fundamental de un espacio de textura usado en graficación por computadora.

TEXTURA 2D

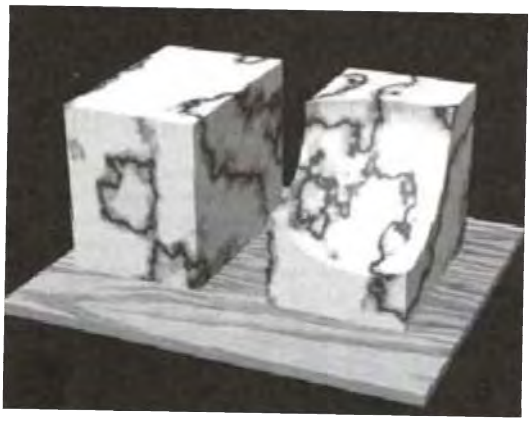
a)



b)

TEXTURA 3D

a)



b)

El problema es que encontrar la textura correcta implica experimentar con los modelos procedurales que las generan, por lo que se requiere de una herramienta de software que permita modelar las imágenes en 2D y 3D de manera interactiva y rápida.

Comercialmente existen paquetes que generan texturas con imágenes pequeñas y en 2D, pero no proporcionan la información de la "función" que generó tal textura, por lo que no es posible extender el resultado en imágenes más grandes de 2D y 3D; por otro lado el software comercial es muy costoso.

En este trabajo de tesis se desarrollara un modelo de síntesis de texturas procedurales para su utilización en un simulador virtual de resección transuretral de próstata (RTU) *(Arámbula, et al., 2006),* con la finalidad de conseguir una textura de tejido lo más parecido visualmente al tejido real. Utilizando funciones primitivas de ruido, funciones de patrones procedurales, conceptos de modelos fractales, así como funciones trigonométricas y aritméticas.

El simulador por computadora para RTU por vaporización**,** es una de las líneas de investigación dentro del CCADET (Centro de Ciencias Aplicadas y Desarrollo Tecnológico) en el laboratorio de Análisis de Imágenes y Visualización, UNAM, que consistente en un ambiente virtual con un modelo en 3D de la próstata que permite simular deformaciones y resecciones de tejido blando con un resectoscopio virtual interfaz electromecánica pasiva) semejante a un resectoscopio real, a través del cual puedan entrenarse residentes en urología. *(Padilla, et al., 2004 págs. 767-777)*

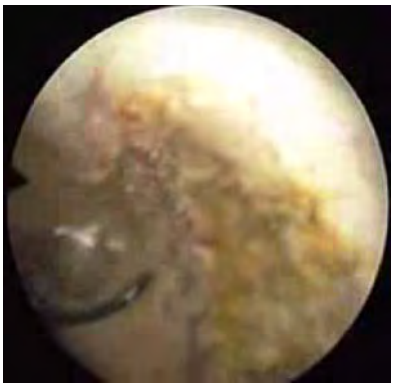
1.5 Hiperplasia prostática benigna (HPB)

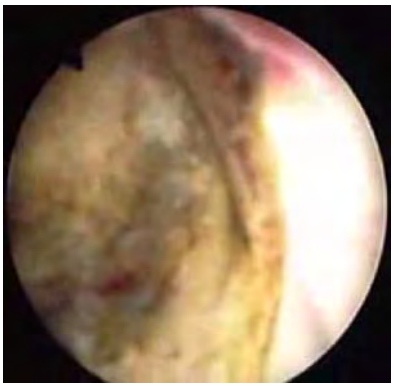
Para entender mejor el procedimiento de resección transuretral de próstata es necesario comentar brevemente que existe un padecimiento llamado *Hiperplasia Prostática Benigna* (HPB). La HPB consiste en un aumento de tamaño de la próstata de origen no cancerígeno

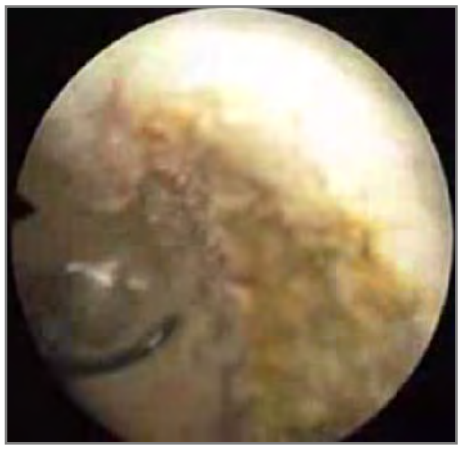
En la mayoría de los varones, la próstata empieza a crecer a partir de los 50 años y, en algunos casos, incluso antes. Con el paso de los años, la prevalencia de HPB histológicamente identificable aumenta, de manera que hacia los 60 años es más del 50% y hacia los 85 alcanza un 90% de los varones. *(Clemente Ramos, et al., 2001)*

La próstata del adulto tiene una forma similar a una castaña, con la base a nivel del cuello vesical y la uretra membranosa. Sus dimensiones suelen ser: 4 cm de ancho, 3 cm de largo y 2 cm de espesor. Está formada por tejido glandular y estroma[3] fibromuscular. En un determinado momento de la vida del individuo, próximo a los 50 años (e incluso antes), en el interior de la glándula prostática se inicia un crecimiento singular que se traduce en la formación de nódulos de tamaño variable, constituyendo lo que conocemos con el nombre de hiperplasia. Este proceso entraña una multiplicación de las células epiteliales y del músculo liso, así como un desarrollo exagerado del tejido conjuntivo en las glándulas que tapizan la

[3] **Estroma:** Armazón de un tejido, que sirve para sostener entre sus mallas los elementos celulares.







Después de estudiar varios ejemplares de este tipo de imagen se observaron las siguientes características a sintetizar:

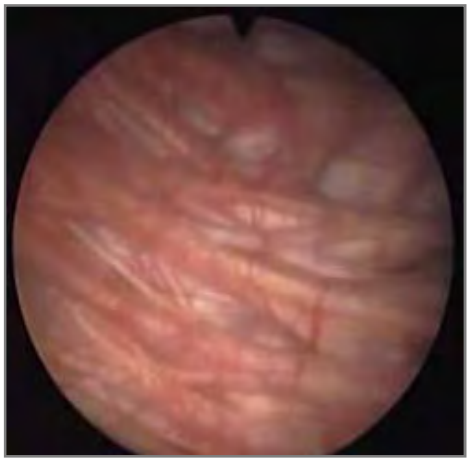
- Tejido de apariencia de color amarillo.
- Algunas áreas blancas.
- Con pocas manchas en rojo.

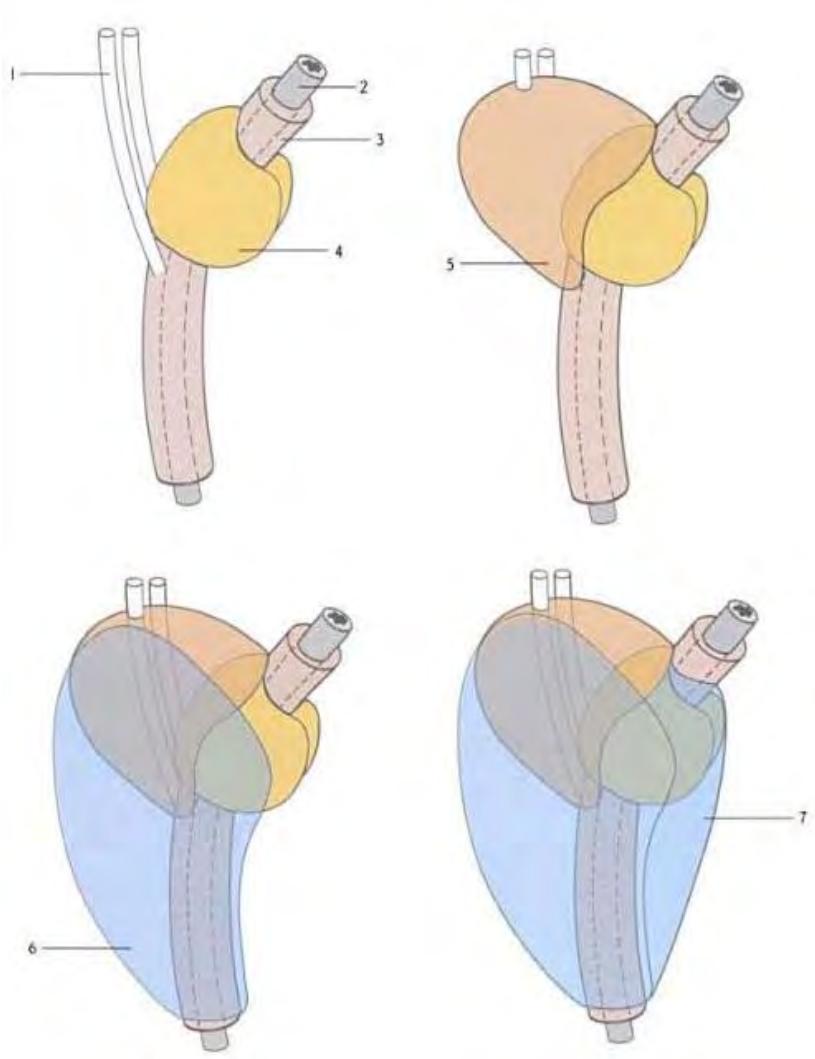
Como se menciono con anterioridad el procedimiento de la RTU se realiza con la ayuda de un resectoscopio, el cual tiene integrado un elemento de resección metálico en forma de asa que se encuentra conectado a una fuente de energía eléctrica con la posibilidad de cortar y coagular el tejido prostático. Para el corte se utiliza una potencia de 120-300 watts, mientras que para la cauterización se utiliza una potencia de 50-116 watts, esta medida será establecida dependiendo del grosor de la resección y la técnica del cirujano. *(CEC Electromedicina)*

Se pueden realizar cortes o cauterizar al tejido prostático debido a la corriente que pasa a través de la resistencia eléctrica del elemento de resección metálico, el cual calienta y eleva la temperatura del tejido. En el corte, el objetivo es calentar las células del tejido tan rápidamente como sea posible para que exploten vaporizándose, dejando una cavidad. Cuando se mueve el elemento de resección y hace contacto en otra parte del "tejido fresco", explota nuevas células realizando otra incisión.

En el caso de la cauterización puede centellear al tejido sin un efecto de corte significante porque el calor es más ampliamente dispersado por las chispas largas y porque el efecto de calentamiento es intermitente. La temperatura del agua en las células no se eleva lo suficiente para despedir vapor. De esta forma las células se deshidratan lentamente, pero no se rompen para formar una incisión o corte. Lo que se aprecia visualmente es un tejido que se torna de un color amarilloso-marrón claro. *(CEC Electromedicina)*

Por lo cual es importante la síntesis de una textura de un color amarillo-pardo para poder simular el efecto de cauterización del tejido, el cual está presente en el procedimiento real de una RTU, lo que nos daría en el simulador un aspecto visual más real cuando se trabaja con el elemento de resección, con el efecto de cauterización sobre el tejido prostático.




1
2
3
4
5
6
7

CAPÍTULO 2

2. FUNDAMENTOS TEÓRICOS

En el presente capítulo se plantearan los fundamentos y antecedentes teóricos en los que se ha basado el desarrollo de este trabajo de tesis.

Se profundizará más en la generación de texturas procedurales, conociendo sus ventajas y desventajas; además de conocer algunas de las funciones y patrones que ayudan en su generación. Para finalmente definir otros atributos importantes como son: color, matiz, saturación y brillo, los cuales también sirvieron como fundamentos y antecedentes en este trabajo.

2.1 Definición de textura procedural

El adjetivo de procedural se utiliza en ciencias computacionales para distinguir las entidades que se describen en el código de un programa; es decir el conjunto de acciones y el orden en que se ejecutan estas acciones. Por lo cual casi cualquier cosa que hagamos en la computadora, tendrá un aspecto procedural en algún nivel.

Por ejemplo en el trazado de mapas de textura en una superficie, el componente procedural o de procedimiento es la creación del módulo de la textura.

Por lo cual podemos definir una *textura procedural* como una imagen sintética, la cual es generada a partir de un algoritmo[4] o modelo. Sin embargo se puede incluir imágenes en la generación de texturas procedurales si a estas se les incorporaran algunas de las operaciones primitivas. Ya que algunas texturas procedurales con una buena apariencia visual, se han obtenido gracias a la combinación de procedimientos modificación o distorsión a partir de una imagen. *(Ebert, et al., 2002)*

Como se mencionó con anterioridad las texturas procedurales, son muy útiles en la creación de patrones de diferentes materiales, como vetas de madera o mármol, al utilizar funciones armónicas que se definen en un espacio tridimensional. Se puede obtener las variaciones en la textura de la madera o el mármol sobreponiendo una función de ruido en las variaciones armónicas, como se aprecia en la Figura 2.1. *(Foley, et al., 1996)*

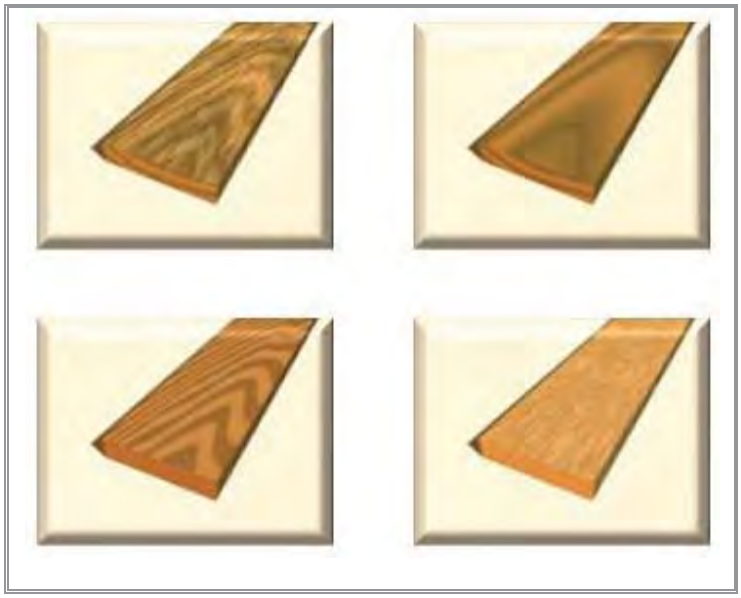


Figura 2.1 Obtención de variaciones en texturas procedurales con la modificación del algoritmo base. *(Blender, 2004)*

[4] **Algoritmo:** En matemáticas, ciencias de la computación, y disciplinas relacionadas, un algoritmo es una lista bien definida, ordenada y finita de operaciones que permite hallar la solución a un problema.

2.1.1 Breve cronología de texturas procedurales

Desde los primeros días del mapeado de texturas, una variedad de investigadores trabajaron con diferentes modelos de texturas para generar imágenes de texturas en lugar de imágenes escaneadas o pintadas. *(Ebert, et al., 2002)*

Blinn y Newell en el año de **1976** utilizaron la síntesis de Fourier.

Fu y Lu para **1978** propusieron una gramática sintética basada en la técnica de generación de texturas.

Schacter y Ahuja en **1979** y Schacter en **1980** utilizaron la síntesis de Fourier y modelos estocásticos de diversos tipos de texturas para generar imágenes de simuladores de vuelo.

Fournier, Fussell y Carpenter en el año **1982** además de Haruyama y Barsky en **1984** propusieron utilizar métodos de subdivisión estocásticos para generar texturas, mejor conocidos como fractales.

Otros investigadores desarrollaron modelos estadísticos analizando las propiedades de texturas naturales, y a continuación tratar de reproducir la textura con datos estadísticos.

Cook en el año de **1984** describió el sistema de "árboles de sombra", el cual fue uno de los primeros sistemas en los que fue conveniente generar un procedimiento de texturas durante el renderizado[5].

Los árboles de sombra permitieron el uso de un modelo diferente de sombreado para cada superficie, así como para las fuentes de luz y de atenuación a través de la atmósfera. Debido a las aportaciones del modelo de sombreado, el modelo hizo posible el uso de texturas para controlar cualquier parte del cálculo de sombreado. Color y transparencia, reflejo de la imagen, el desplazamiento y las texturas sólidas pudieron todas ser implementadas usando el sistema de "árboles de sombra".

Perlin en el año de **1985** describió un completo lenguaje para la generación de texturas procedurales y asentó las bases para la clase más popular de texturas procedurales, es decir, todas aquellas que se basan en el ruido, una generación estocástica de textura primitiva.

Turk, Witkin y Kass en **1991** describieron modelos de textura sintética, inspirados en procesos bioquímicos que producen entre otros efectos patrones de pigmentación en la piel de los animales.

---

[5] **Renderizado:** Proceso de generar una imagen desde un modelo. En términos de visualización en una computadora, la "renderización" es un proceso de cálculo complejo desarrollado por la computadora destinada a generar una imagen 2D a partir de una escena 3D.

Sims en el año **1991**, describió un muy novedoso sistema de síntesis de textura, en el que las texturas se representan como expresiones automáticamente modificadas y combinadas por un sistema de programación genética. *(Ebert, et al., 2002)*

Todos los métodos de síntesis de textura mencionados en este tema se les podrían llamar procedurales o de procedimiento.

2.1.2 Ventajas de las texturas procedurales

Las ventajas que presentan las texturas procedurales sobre una imagen de mapa de bits[6] son:

- La textura procedural es extremadamente compacta. El tamaño de la textura suele ser medidos en kilobytes, mientras que el tamaño de una imagen se suele medirse en megabytes. Esto es especialmente cierto en el caso de texturas sólidas en tres dimensiones.
  Nota: *Esto ventaja se refiere a la generación en tiempo real de la textura, no hay necesidad de almacenar propiamente la imagen.*

- Una representación procedural no tiene una resolución fija. En la mayoría de los casos se puede proporcionar una textura totalmente detallada no importando que tan cerca la tenga el observador.

- Una textura procedural no tiene un tamaño fijo. En otras palabras, es ilimitada en la amplitud y puede cubrir áreas grandes, sin bordes o costuras no deseadas; además de no utilizar un patrón de repetición.

- Una textura procedural puede ser parametrizada, por lo que puede generarse una clase de texturas relacionadas para lo que se necesite.

Muchas de estas ventajas son sólo posibles mejorías para lograr la textura deseada; teniendo en cuenta que se debe hacer un esfuerzo para obtener lo que se pretende. Ya que

[6] **Mapa de bits**: Estructura de datos que representa una imagen rectangular de pixeles o puntos de color, la cual puede se visualizada en la computadora.

un mal procedimiento para la obtención de una textura puede sacrificar algún potencial de esas ventajas. *(Ebert, et al., 2002)*

2.1.3 Desventajas de las texturas procedurales

Las desventajas de las texturas procedurales en comparación con una imagen son los siguientes:

- Una textura procedural puede ser difícil de construir y de depurar. La programación es a menudo compleja, y la programación implícita en la descripción de un dibujo es especialmente difícil en los casos no triviales.

- Una textura procedural puede generar resultados sorpresivos; ya que a menudo es más fácil predecir el resultado al momento en que se escanea o pinta una imagen. Por lo que las texturas procedurales son difíciles de controlar.

- En cuestión de evaluación, puede ser más lento el acceder a una textura procedural que a una imagen almacenada, en el caso de generarla en tiempo real.

- El aliasing[7] puede ser un problema en la generación de texturas procedurales; llevar a cabo el antialiasing puede ser complicado y es poco probable que sea resuelto de forma automática. *(Ebert, et al., 2002)*

[7] **Aliasing:** Artefacto gráfico característico que presenta en pantalla ciertas curvas y líneas inclinadas con un efecto visual tipo "sierra" o "escalón" no deseados.

2.2 Generación de patrones procedurales

La generación de imágenes procedurales es posible porque muchas cosas a nuestro alrededor, tanto naturales como sintéticas, tienen propiedades geométricas, estocásticas (aleatorias) y de simetría que se pueden describirse matemáticamente.

En cuanto a la simetría, como lo describe el profesor Stewart "Los objetos matemáticos más simples son los números, y los más simple patrones de la naturaleza son numéricos. Por ejemplo, las fases de la luna en realizar un ciclo completo de luna nueva a luna llena, se realiza cada veintiocho días. Un año es de trescientos sesenta y cinco días. La gente tiene dos piernas, los gatos tienen cuatro, algunos insectos tienen seis y arácnidos tienen ocho. En casi todas las flores, el número de los pétalos es uno de los números que ocurre en la extraña secuencia de 3, 5, 8, 13, 21, 34, 55, 89, …, también conocida como la Secuencia de Fibonacci". *(Stewart, 1997)*

En cuanto a la geometría, está se utiliza para describir con mayor facilidad a la naturaleza, por ejemplo, las hojas de un helecho en términos de modelos recursivos se define como la ocurrencia de una característica de forma a diferentes escalas, también conocido como *fractales*. Estas propiedades pueden verse en todas los escalas, desde las sierras y las formaciones de nubes hasta los detalles de las estructuras celulares en las plantas y los animales.

En cuanto a la aleatoriedad, otra función primitiva muy útil en la generación de texturas procedurales es la utilización de *ruido*. El ruido en sí mismo es sólo un número al azar o una colección de números aleatorios que por sí solos no son muy interesantes.

Sin embargo, si tomamos el ruido y lo utilizamos como una entrada de una función para cambiar la salida, las cosas cambian considerablemente. Para ilustrar mejor esto, imaginemos una función recursiva que genera un cilindro con un cilindro más pequeño dentro de él. Si ahora se añade algo de ruido en la función podemos cambiar la salida y producir algo que se asemeja a un tronco de un árbol como se aprecia en la figura 2.2. *(Gudlaugsson, 2006)*

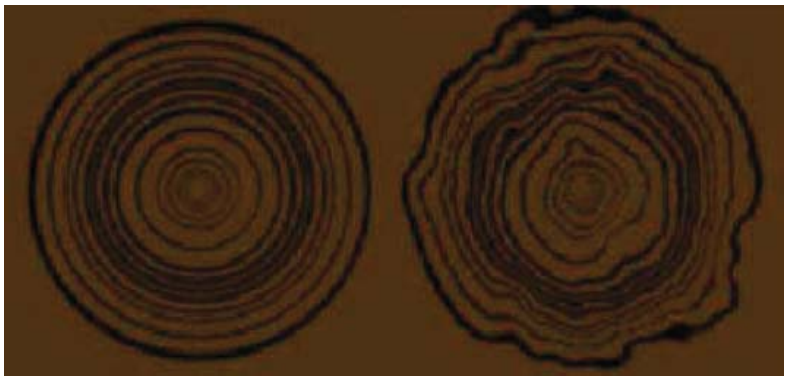


Figura 2.2 Imagen izquierda: función recursiva de cilindros. Imagen derecha: la misma función de cilindros pero añadiendo una función de ruido con lo cual obtenemos la simulación de un tronco de un árbol. *(Gudlaugsson, 2006)*

2.2.1 Función de ruido

Para la generación de texturas procedurales irregulares, se necesitan de funciones primitivas "irregulares" usualmente llamadas *ruido*. Esto entre otras cosas, es una función de naturaleza estocástica[8] y rompe con la monotonía de los modelos que de otro modo serían demasiado regulares.

Cuando usamos términos como "aleatorio" y "estocástico", casi siempre quiere decir "aparentemente aleatorio".

La textura primitiva estocástica más evidente es el *ruido blanco*, una fuente de números aleatorios distribuidos de manera uniforme sin correlación alguna entre los siguientes números sucesivos; el ruido blanco puede ser generado por un proceso físico al azar, como el ruido térmico que se produce dentro de muchos sistemas electrónicos analógicos y no es nunca el mismo dos veces.

Muchas personas han utilizado los generadores de números aleatorios para crear sus programas con efectos de imprevisibilidad, por ejemplo para generar movimientos y comportamientos de los objetos con apariencia más natural, o generar texturas. Los generadores de números aleatorios ciertamente tienen sus usos, pero a veces su salida puede ser demasiado dura para un aspecto natural.

Casi cualquier cosa en la naturaleza tiene una naturaleza estocástica: la desigual distribución de césped en un campo, las olas en el mar, los movimientos de una hormiga, el movimiento de las ramas de un árbol, las pautas de mármol, los vientos. Todos estos fenómenos muestran el mismo patrón de grandes y pequeñas variaciones. Una función muy común para generar este tipo de patrones es, como se explicará más adelante, la función de ruido de Perlin; esta se recrea simplemente sumando ruidosos funciones en una variedad de diferentes escalas. Para crear una función de ruido Perlin, se necesitan dos cosas, una función de ruido, y una función de interpolación. *(Ebert, et al., 2002)*

[8] **Estocástico:** Entendemos el concepto de estocástico como un fenómeno que generalmente tiene asociadas fuentes de variación o valores aleatorios, que se encuentran fuera del control del tomador de decisiones. *(Azarang, et al, 1996)*

2.2.2 Función de ruido de Perlin

Definiciones

Primeramente para entender la generación de la función de ruido de Perlin, se debe recordar lo que es amplitud y frecuencia. En la figura 2.3 se muestra una función periódica o senoidal y una función aleatoria (ruido).

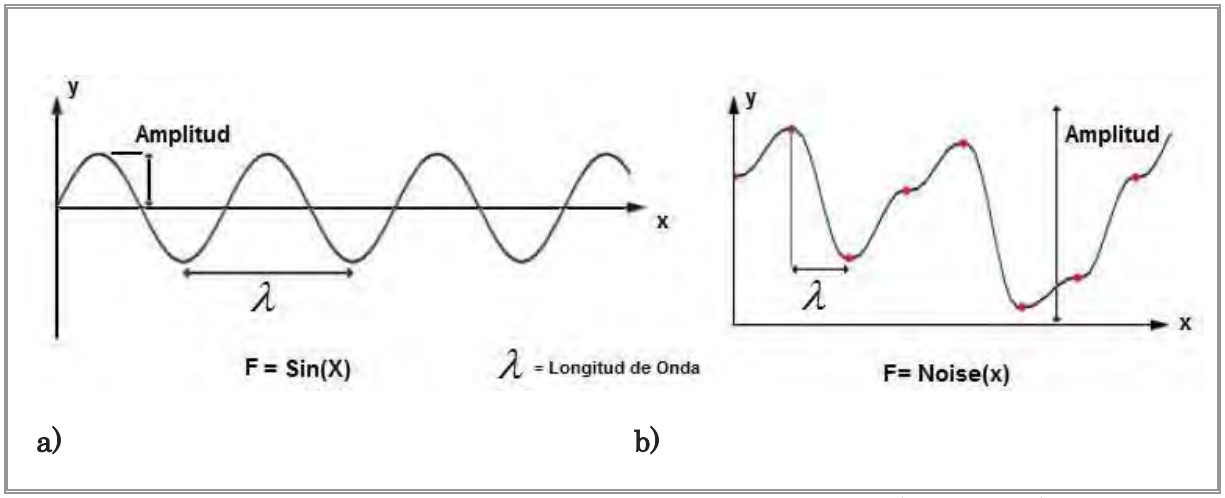


Figura 2.3 a) Función senoidal. b) Funcion de aleatoria (ruido). *(Elias, 2003)*

En el caso de la función senoidal, la *amplitud* es la distancia que hay entre el eje x y el valor máximo o mínimo del la función (cresta o valle). La *longitud de onda* es la distancia entre crestas, valles o partes idénticas sucesivas. El *periodo* lo podemos definir como el tiempo necesario para la repetición completa de un ciclo. Y el número de ciclos por unidad de tiempo recibe el nombre de *frecuencia*, la frecuencia y el periodo son recíprocos entre si, por lo cual se puede escribir la siguiente fórmula. *(Resnick, 1999)*

$$frecuencia = \frac{1}{período}$$

En la función de ruido los puntos rojos indican los valores aleatorios que se obtuvieron de la función. En este caso, la *amplitud* es la diferencia entre el valor mínimo y máximo de la función. La *longitud de onda* es la distancia desde un punto al siguiente. Y la *frecuencia* se obtiene con la misma fórmula antes menciona.

Creación de la función de ruido de Perlin

Ahora bien, si se trabaja con funciones aleatorias a diferentes frecuencias y amplitudes, y se suman todas, se crea una agradable función de ruido. Esto es a lo que se conoce como la función de ruido de Perlin ejemplificada en la figura 2.4.

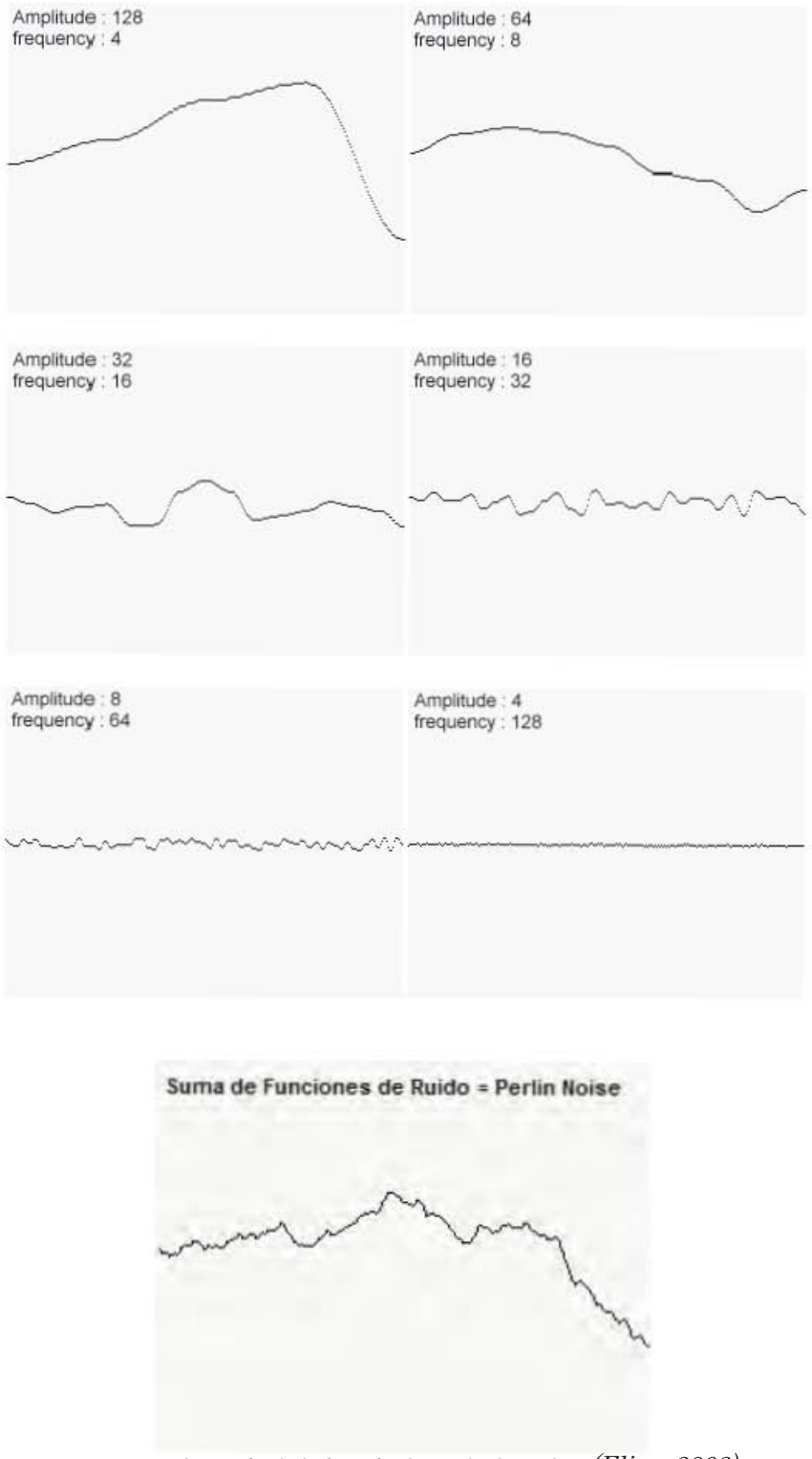


Figura 2.4 Obtención de la función de Ruido de Perlin. *(Elias, 2003)*

Sin embargo, se pueden crear funciones con ruido Perlin de diferentes características mediante el uso de otras frecuencias y amplitudes en cada paso. Para hacerlo más sencillo, y para evitar la repetición de las palabras de amplitud y frecuencia todo el tiempo, se utiliza un solo número para especificar la amplitud de cada frecuencia. Este valor es conocido como *persistencia*. Y se puede definir como el multiplicador que determina la rapidez con que las amplitudes para cada una de las octavas sucesivas disminuyen en la función de ruido Perlin. En términos matemáticos tenemos:

$$frecuencia = 2^i$$

$$amplitud = persistencia * (octava\ anterior)$$

En donde i es la i^n función de ruido que se añade. Y la amplitud es igual al producto del valor de la persistencia y la octava anterior calculada. Cada una de las sucesivas funciones que se añaden en la función de ruido se le conoce como una octava. La razón de esto es que cada función de ruido es el doble de la frecuencia de la anterior.

Para ilustrar el efecto de la persistencia en la salida del ruido Perlin, se presentan los resultados en la figura 2.5; en donde tenemos la frecuencia de la función, el valor de la amplitud con el factor de persistencia, y como resultado la función del ruido de Perlin. *(Perlin, 1999)*

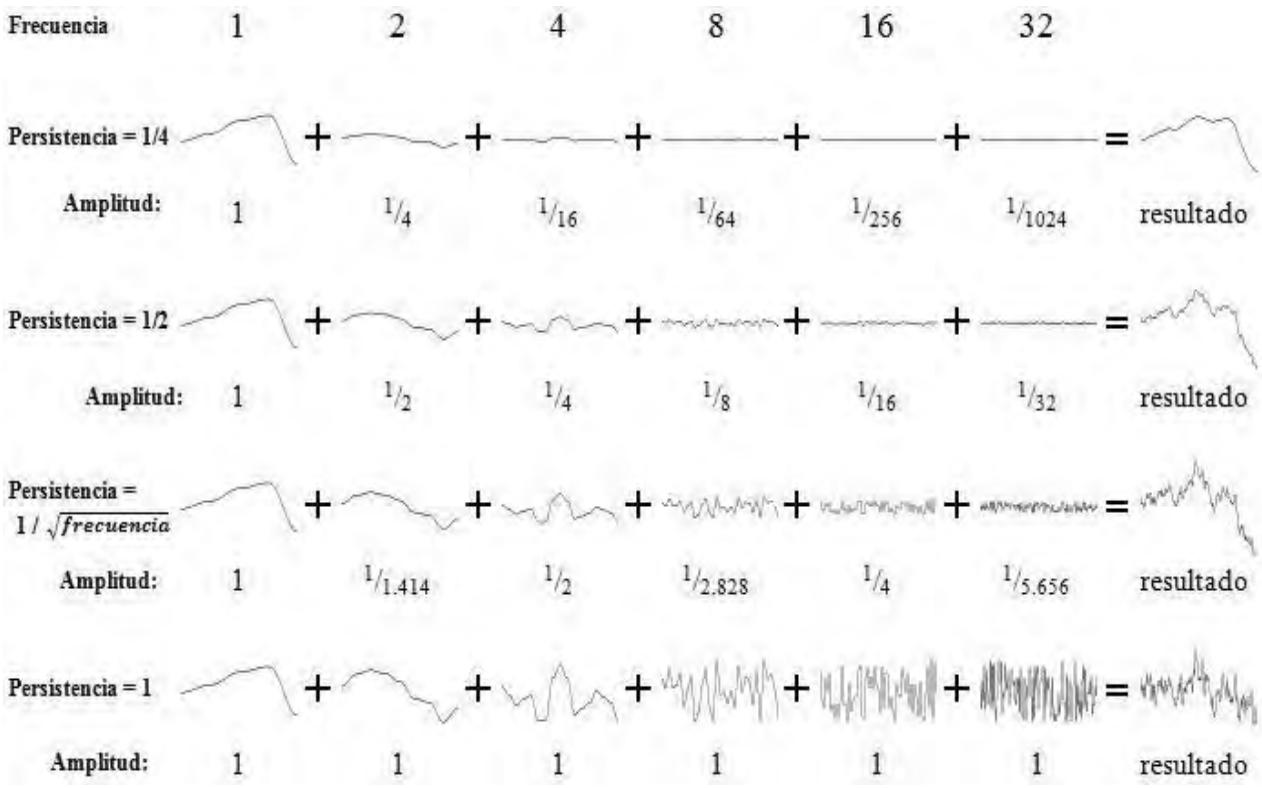


Figura 2.5 Diferentes resultados de la función de Ruido de Perlin con diferentes valores de persistencia. *(Elias, 2003)*

En la función de ruido de Perlin se aprecian grandes, medianas y pequeñas variaciones. Es por eso que muchos paisajes generados por computadora se hacen con este método.

Utilizando diferentes funciones aleatorias de generación de ruido en 2D, se obtienen los patrones que se muestran en la figura 2.6.

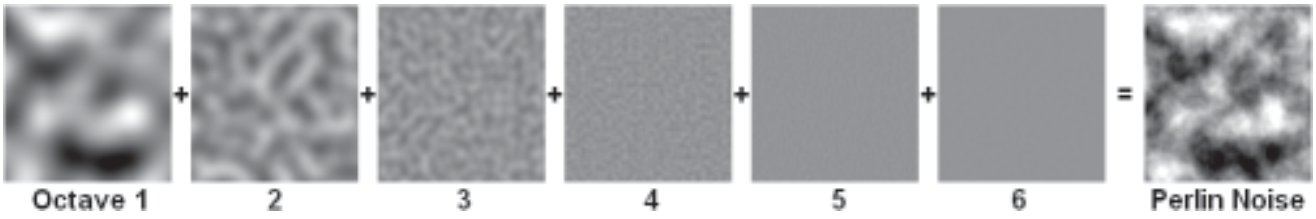


Figura 2.6 Imagen que muestra la combinación de 6 octavas para crear la función de ruido de Perlin 2D. *(Perlin, 1999)*

Interpolación

Después de haber creado la función de ruido, se tendrán que suavizar los valores que devuelve. Una función estándar para la interpolación se puede escribir de la siguiente manera: se deben tomar 3 valores de entrada (a, b, x), en donde $0 \leq x \leq 1$. La función de interpolación devuelve un valor entre a y b sobre la base del valor x. Es decir, cuando x es igual a 0, devuelve una a, cuando x es 1 devuelve b. Cuando x está entre 0 y 1 devuelve algún valor entre a y b, esta función también es conocida como interpolación lineal. En donde en términos de código quedaría de la siguiente forma:

```
funcion_Interpolacion _Lineal (a, b, x)
return a*(1-x) * b*x
fin de funcion
```

Integrando todo

La parte principal de la función de Perlin es generar un ciclo en donde cada iteración del ciclo añade otra octava de dos veces la frecuencia. Cada iteración pide una función diferente de ruido, denotado por $Noise_i$. Una vez obtenidos los valores estos deben ser interpolados para obtener una función suave, la cual podemos ocupar por ejemplo en la generación de texturas procedurales.

Se puede jugar tanto como se quiera, probando diferentes persistencias con diferentes frecuencias y diferentes dimensiones. Utilizar incluso una función de Perlin y afectar las propiedades de otra función, etc. *(Elias, 2003)*

Las siguientes texturas mostradas en la figura 2.7 se realizaron con base en la función de ruido de Perlin:

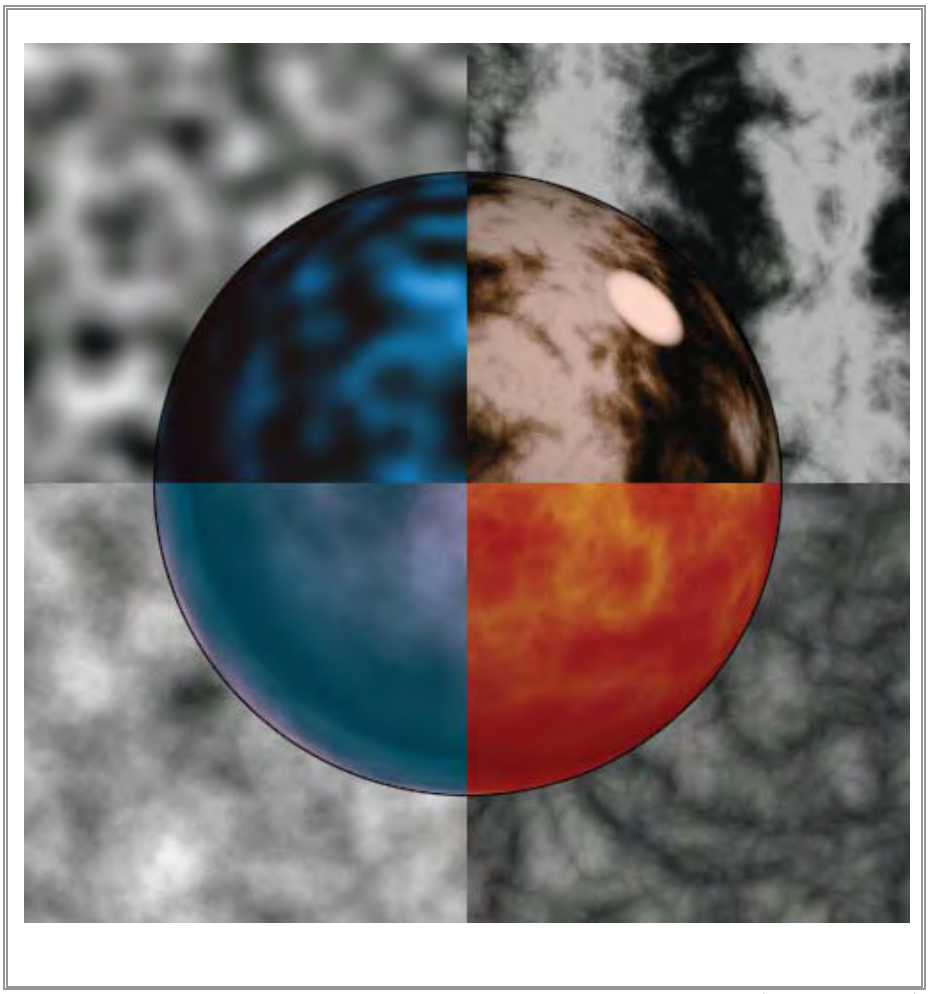


Figura 2.7 Texturas 3D generadas en base a la función de Ruido de Perlin. *(Perlin, 1999)*

2.2.3 Fractales

Hoy en día los fractales han llamado mucho la atención. Las imágenes que se obtienen con ellos son espectaculares, y se han desarrollado varios métodos para generarlos. El termino fractal se refiere a cualquier cosa con gran cantidad de auto-similitud exacta o estadística, por tanto los únicos objetos fractales verdaderos son aquellos generados por procesos infinitamente recursivos.

La mejor forma de ilustrar lo que significa auto similitud es con el ejemplo de copo de nieve de von Koch. A partir de una línea recta con una protuberancia, reemplazamos cada segmento de línea con una figura exactamente igual a la línea original. Si repetimos este proceso un número infinito de veces, se dice que el resultado es auto similar: todo el objeto es similar a una sub-porción de él mismo como se ilustra en la figura 2.8. *(Foley, et al., 1996)*

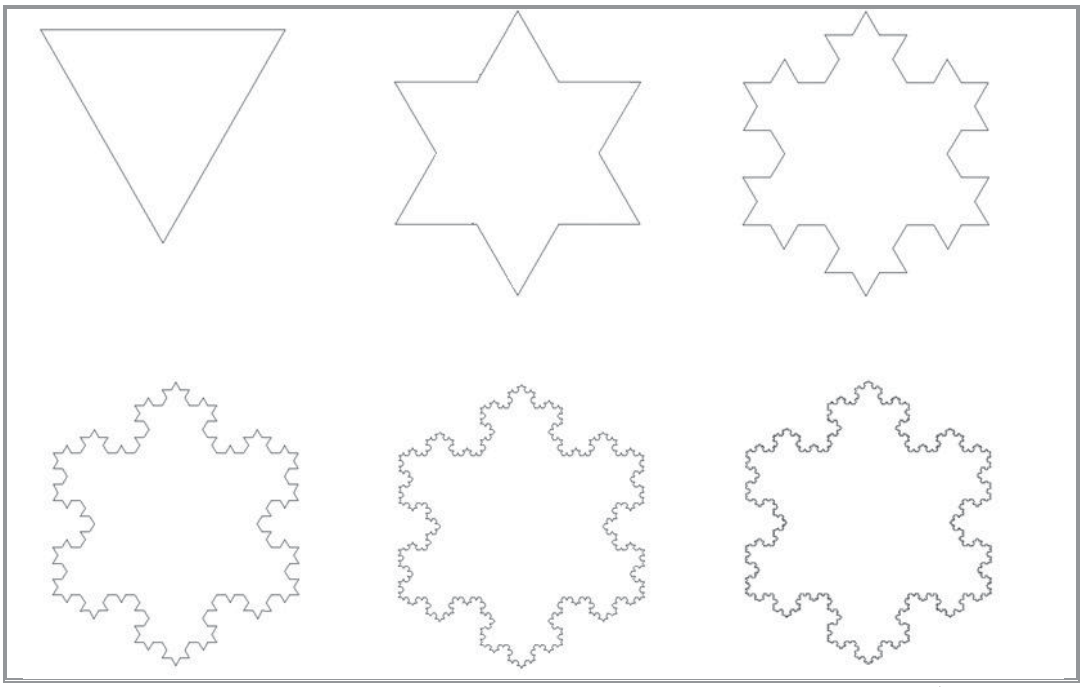


Figura 2.8 Concepto de la generación de fractal con la construcción del copo de nieve de von Koch. *(Foley, et al., 1996 pág. 406)*

2.2.3.1 Uso de fractales en la generación de texturas

Estos resultados son muy sugerentes para el modelado de formas naturales, ya que muchos objetos naturales parecen exhibir una gran auto-similitud. Las montañas tienen picos, picos más pequeños, rocas y gravas, todo lo cual parece similar; los árboles tienen troncos, ramas y tallos, que también se ven similares; las costas tienen bahías, caletas, estuarios, riberas y zanjas de drenado, que también son de apariencia similar. Si nos fijamos en muchas cosas en la naturaleza, nos daremos cuenta de que son fractales con diferentes niveles de detalle.

Así que el modelado de la auto-similitud a cierta escala parece ser una forma de generar modelos atractivos de fenómenos naturales. *(Foley, et al., 1996)*

Dado que para la construcción de fractales, se necesitan de varias operaciones repetitivas las cuales se pueden generar en base en algún algoritmo ejecutado por computadora. Logrando además con su construcción una complejidad visual potencialmente ilimitada que se obtiene relativamente de una pequeña cantidad de código. *(Ebert, et al., 2002)*

Los métodos fractales han probado ser útiles para modelar una amplia variedad de fenómenos naturales. En las aplicaciones de graficación por computadora son muy utilizados para modelar terrenos, nubes, agua, árboles y otras plantas, plumas, piel y distintas texturas de superficie, siendo muy eficientes en elaborar patrones atractivos. *(Hearn, et al., 1995)*

2.2.4 Conceptos básicos para la generación de texturas procedurales

Es importante mencionar que las texturas procedurales descritas en este apartado no tienen ninguna base en principios físicos o biológicos, más bien están realizadas básicamente por el cálculo de los algoritmos por medio de una computadora con el fin de obtener resultados visuales aceptables. *(Ferguson, 2001)*

Así que lo más conveniente es empezar a definir algunos de los conceptos básicos para la generación de la mayoría de texturas procedurales:

2.2.4.1 Exponente de Hurst (H)

Este parámetro en términos de matemáticas fractales, está directamente relacionada con la dimensión fractal[9], lo que da una medida de la rugosidad en la superficie. La relación entre la dimensión fractal (D) y el exponente de Hurst (H) es:

$$D = 2 - H$$

Por lo tanto si tenemos H=0.50 entonces D=1.50. Un valor de $1 < H \leq 1.50$, resultara en una dimensión fractal más cercana a una línea. Esto daría como resultado una línea más suave con menos picos. Con un rango de $0 < H < 0.50$ arrojaría una dimensión fractal mayor, más puntiaguda. *(Heinz-Otto Peitgen, 2004)*

La figura 2.9 muestra los resultados para diferentes valores de H.

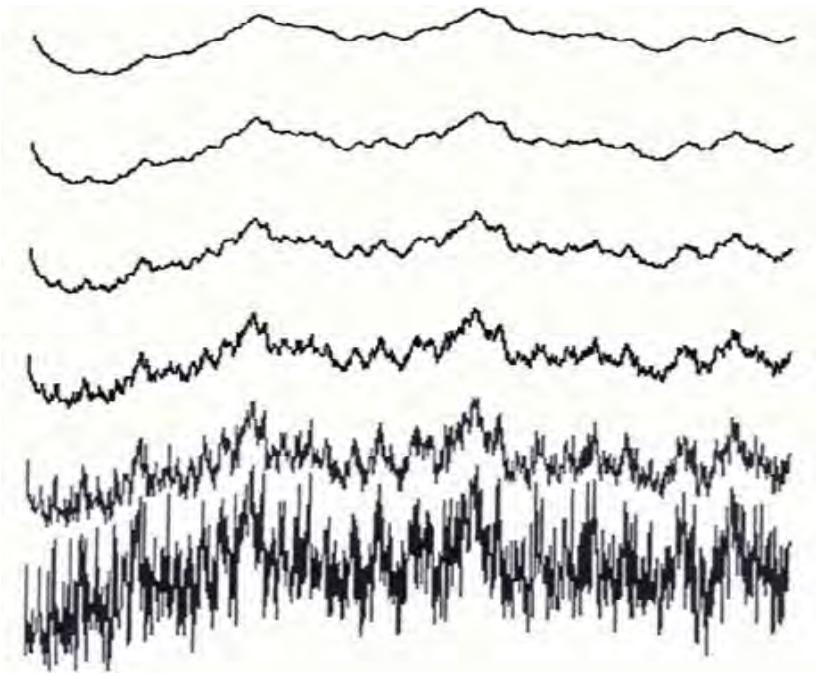


Figura 2.9 Resultados de la variación del parámetro H (exponente de Hurst), variando de 1.0 a 0 con incrementos de 0.2.
(Ebert, et al., 2002 pág. 439)

[9] **Dimensión fractal:** Se define como una medida cuantificable de las características de auto semejanza de las formas fractales. Por ejemplo, la dimensión fractal se ha utilizado para medir la rugosidad de las costas.

Este parámetro es utilizado para saber la afección rugosidad en nuestro patrón de texturas. *(Ebert, et al., 2002)*

2.2.4.2 Lacunaridad

La noción de "lacunaridad" fue propuesta por Mandelbrot en 1982, como una medida complementaria para determinar estructuras geométricas con la misma dimensión fractal, pero con diferente textura. La lacunaridad, que deriva del vocablo latino "laguna", es una medida de la distribución de los espacios vacíos en una estructura geométrica. Por lo cual tiene que ver con la distribución del tamaño de los agujeros. Si un fractal tiene grandes lagunas o agujeros, el parámetro de *lacunaridad* es alto; por otro lado, si un fractal es casi invariante, su *lacunaridad es de un valor* bajo, como podemos apreciar en la figura 2.10. *(Mandelbrot, 1997)*

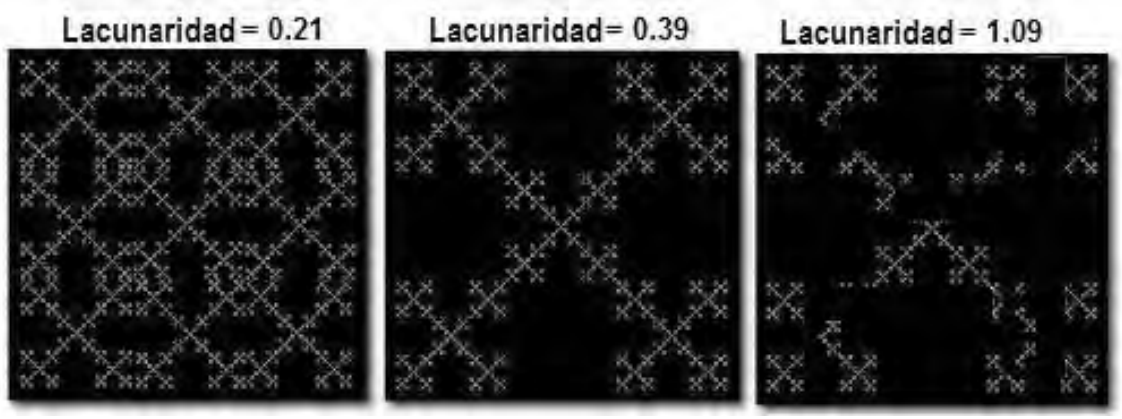


Figura 2.10 Imágenes de fractales en donde se aprecia el concepto de lacunaridad. *(Turcotte, 1997 pág. 110)*

En la generación de texturas procedurales se utiliza este término, para saber la afección de lagunas o huecos en nuestro patrón de texturas. *(Ebert, et al, 2002)*

2.2.4.3 Octavas

Los fractales pueden tener potencialmente ilimitados detalles. Pero esos detalles deben tener "límites" para que puedan ser generados por una computadora. De hecho en la naturaleza, los fractales "desaparecen" a cierta escala de nuestra vista, con lo cual se puede definir que tienen una banda limitada. Ese nivel de detalle en la generación de una textura procedural, nos lo proporcionan las octavas.

El valor de la octava nos proporciona el número de veces en que se tiene que hacer la iteración de la función, sumando en cada iteración una textura a escala del patrón original. Un valor grande en la octava agregará detalles más pequeños a la textura.

Sin embargo hay que tener control del número de veces que se va a repetir un ciclo, recordando que también la cantidad de detalle requerido en un punto dado de una imagen, depende de la distancia de la cámara, la resolución de la pantalla, el campo de visión, y otros factores. Además, un exceso de detalle no sólo incrementara el tiempo de cálculo, también puede causar aliasing. *(Ebert, et al., 2002)*

En la figura 2.11, podemos apreciar el concepto antes mencionado. Teniendo la función original sumada a ½ y ¼ de la misma, para obtener un textura con más detalles.

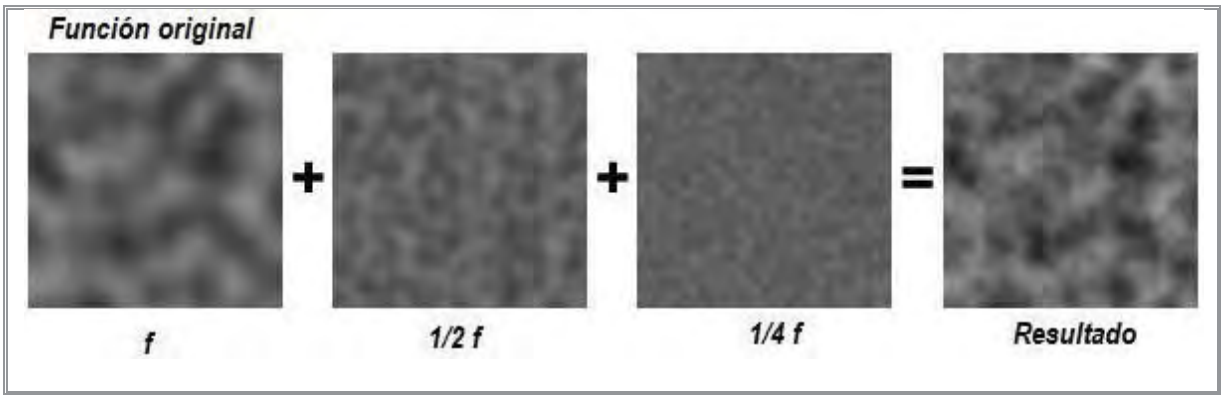


Figura 2.11 Ejemplo del concepto de octavas, teniendo la función *f* original sumada a ½ y ¼ de la misma, dando como resultado una textura con más detalles. *(Poissant, 2002)*

2.2.4.4 Offset

Parámetro utilizado en modelos procedurales de terreno también conocidos como modelos multifractales, por contener altas y pequeñas variaciones en la superficie de una textura. Dicho parámetro se puede interpretar como un valor de altura para generar la textura de un terreno. *(Texture, 2006)*

2.2.4.5 Umbral

Se especifica como un valor umbral que se utiliza para determinar si la textura se debe mostrar o no. Si el valor en un punto de la textura procedural es superior al valor umbral, este modifica el valor final de la textura. Si el valor de la textura es inferior al valor umbral, este no altera el valor final de la textura. *(Texture, 2006)*

2.2.5 Patrones utilizados para generación de texturas procedurales

La construcción de texturas procedurales para este trabajo de tesis, se hizo bajo los parámetros antes mencionados. En primer lugar, se tiene una *función base* (función de ruido de Perlin), la cual es repetida en una variedad de escalas. Se cuenta también con parámetros que controlan la rugosidad (*exponente de Hurst*). Con *octavas* que nos marcarán el número de veces de la iteración en la construcción de la textura agregando más o menos detalles. Por último, tenemos el parámetro de *lagunaridad*, dicha propiedad cambia la frecuencia de la función base en de cada iteración agregando lagunas a la textura final generada. *(Ebert, et al., 2002)*

2.2.5.1 Función movimiento Browniano fraccional

La función movimiento Browniano fraccional, sus siglas en inglés (fBm), es generada a partir de varios ejemplares de ruido juntos, cada ejemplar tiene una frecuencia y amplitud

diferentes. Las frecuencias y amplitudes de las sucesivas adiciones de ruido están relacionadas por el factor de lacunaridad y persistencia, respectivamente. Se le considera una función "auto-similar" dado que es una copia de sí misma a diferentes escalas, ver figura 2.12.

Esta función tiene un aspecto visual agradable, complejo y natural que imita el patrón de muchas texturas en la naturaleza. *(Apodaca, et al., 1999)*

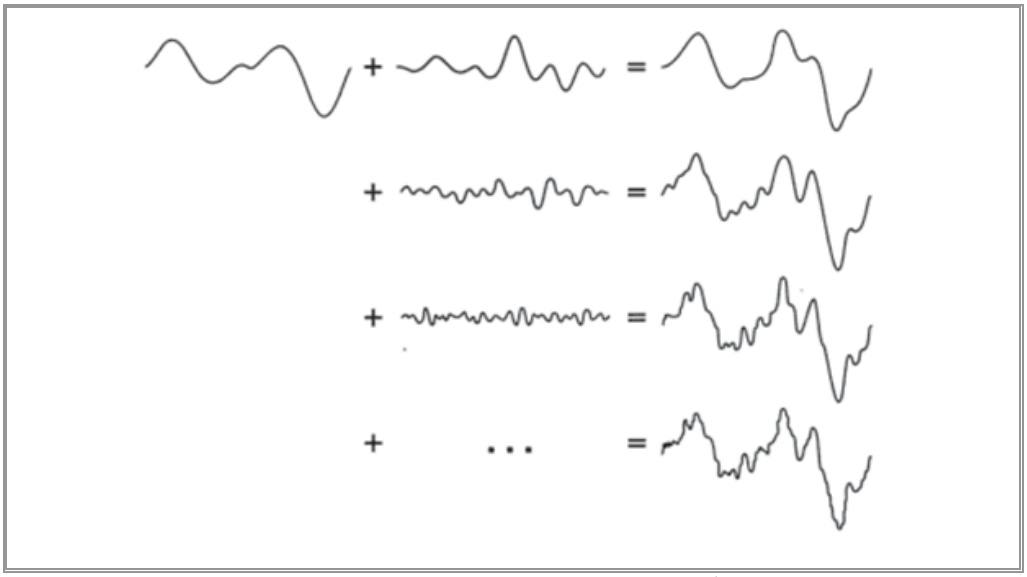


Figura 2.12 Esquema de la función movimiento Browniano fraccional (fBm). *(Apodaca, et al., 1999 pág. 252)*

A continuación se presenta el pseudocódigo de generación de dicha función a fin de comprender los parámetros necesarios para la generación de esta textura. *(Ebert, et al., 2002 pág. 437)*

```
/*
* Procedural fBm evaluated at “point”.
*
* Parameters:
* “H” is the fractal increment parameter
* “lacunarity” is the gap between successive frequencies
* “octaves” is the number of frequencies in the fBm
*/
double fBm( Vector point, double H, double lacunarity, double octaves )
{
   double value, remainder, Noise();
   int i;
   value = 0.0;
   /* inner loop of fractal construction */
    for (i=0; i<octaves; i++) {
    value += Noise( point ) * pow( lacunarity, -H*i );
       point *= lacunarity;
    }
   remainder = octaves - (int)octaves;
   if ( remainder ) /* add in “octaves” remainder */
      /* ‘i’ and spatial freq. are preset in loop above */
      value += remainder * Noise3( point ) * pow( lacunarity, -H*i );
return value;
}
```

En la figura 2.13 se muestras dos cubos texturizados con la función movimiento Browniano fraccional (fBm).

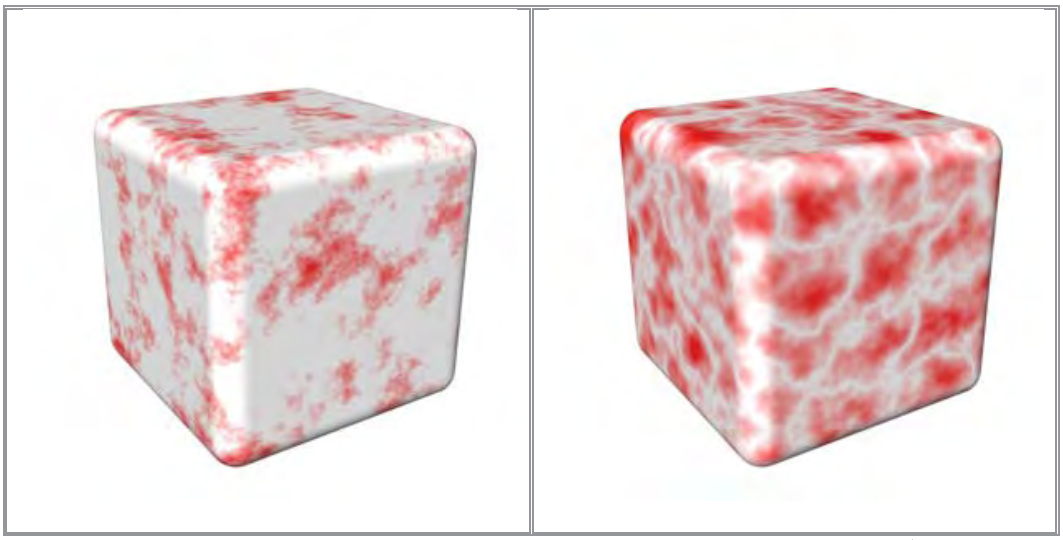


Figura 2.13 Cubos texturizados a base de la función movimiento Browniano fraccional (fBm). *(Texture, 2006).*

2.2.5.2 Función de turbulencia

Un familiar de la función fBm es la función de Perlin llamada turbulencia. La única diferencia entre la turbulencia y la de fBm es el valor absoluto. El resultado es de una apariencia más "ondulante". *(Apodaca, et al., 1999)*

Esta textura combina capas de ruido fractal con diferentes frecuencias para crear complejos patrones detallados e interesantes. Estableciendo una mayor gama de efectos en términos de alteración de los detalles. *(Van, 2004)*

A continuación se muestra el pseudocódigo de la función de turbulencia para la generación de texturas. *(Ebert, et al., 2002 pág. 86)* Para crear este tipo de textura, dado que la función tiene componentes positivos y negativos, solo toma su valor absoluto.

```
float
turbulence(point Q)
{
   float value = 0;
   for (f = MINFREQ; f < MAXFREQ; f *= 2)
      value += abs(snoise(Q * f))/f;
   return value;
}
```

En la figura 2.14 se muestras dos cubos texturizados con la función de turbulencia.

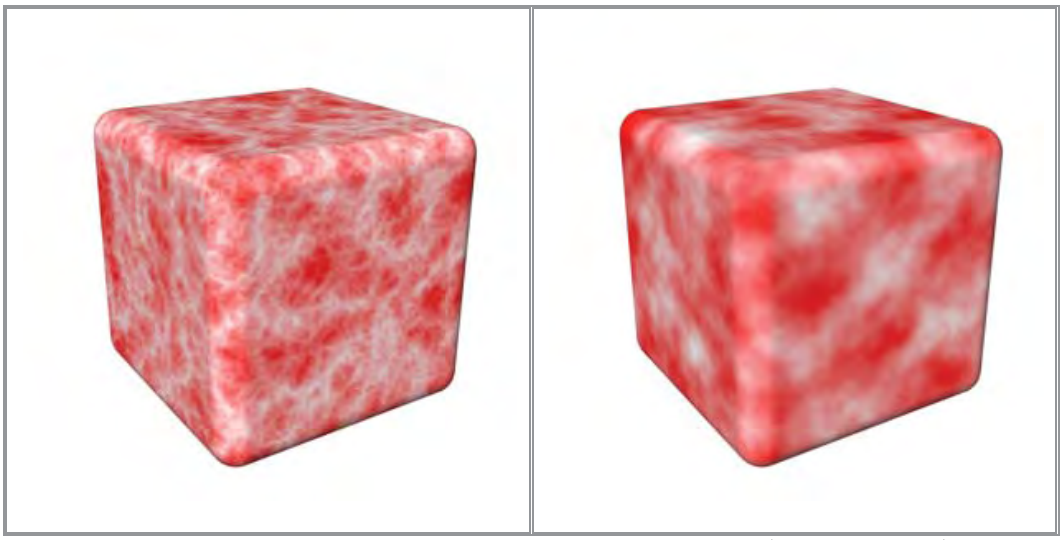


Figura 2.14 Cubos texturizados a base de la función Turbulencia. *(Texture, 2006).*

2.2.5.3 Función de terreno heterogéneo

La función de terreno heterogéneo es conocida como un multifractal. Un multifractal puede ser definido como un fractal que contiene una multiplicidad de medidas, es decir que está formado heterogéneamente y no tienen los mismos "valores" de medidas en todas partes. Así una imagen o textura multifractal podrá tener variaciones a lo largo de la textura, por ejemplo en su dimensión fractal. *(Ebert, et al., 2002)*

En la generación de texturas procedurales la función de terreno heterogéneo generará una apariencia heterogénea en la imagen final obtenida; controlando la cantidad de detalles añadidos de acuerdo a un valor de "offset". Cuando el parámetro offset sea igual a cero la función será heterogénea en extremo, y conforme el valor del offset vaya aumentando la función va ir cambiando de multifractal a monofractal, acercándose a una textura de superficie plana. Dicha función es muy útil para la creación de paisajes (sobre todo planetas), ya que simula la forma en que la tierra tiende a ser más plana y suave en las zonas bajas, y áspera y desigual en el aumento de altura. *(Van, 2004)*

El pseudocódigo para la generación de la función procedural de terreno heterogéneo, se muestra a continuación. *(Ebert, et al., 2002 pág. 500)*

```
/*
* Heterogeneous procedural terrain function: stats by altitude method.
* Evaluated at "point"; returns value stored in "value".
*
* Parameters:
* "H" determines the fractal dimension of the roughest areas
* "lacunarity" is the gap between successive frequencies
* "octaves" is the number of frequencies in the fBm
```

```
* “offset” raises the terrain from “sea level” */
double
Hetero_Terrain( Vector point,
         double H, double lacunarity, double octaves, double offset )
{
    double value, increment, frequency, remainder, Noise3();
    int i;
    static int first = TRUE;
    static double *exponent_array;

   /* precompute and store spectral weights, for efficiency */
   if ( first ) {
         /* seize required memory for exponent_array */
        exponent_array =(double *)malloc( (octaves+1) * sizeof(double) );

        frequency = 1.0;
        for (i=0; i<=octaves; i++) {
             /* compute weight for each frequency */
             exponent_array[i] = pow( frequency, -H );
             frequency *= lacunarity;
        }
        first = FALSE;
   }

    /* first unscaled octave of function; later octaves are scaled */
     value = offset + Noise3( point ); point.x *= lacunarity;
     point.y *= lacunarity; point.z *= lacunarity;

   /* spectral construction inner loop, where the fractal is built */
   for (1=1; i<octaves; 1++) {
       /* obtain displaced noise value */
       increment = Noise3( point ) + offset;

       /* scale amplitude appropriately for this frequency */
       increment *= exponent_array[i];

       /* scale increment by current “altitude” of function */
       increment *= value;

       /* add increment to “value” */
       value += increment;

       /* raise spatial frequency */
       point.x *= lacunarity;
       point.y *= lacunarity;
       point.z *= lacunarity;
   } /* for */

   /* take care of remainder in “octaves” */
   remainder = octaves - (int)octaves;
   if ( remainder ) {
       /* “i” and spatial freq. are preset in loop above */
       /* note that the main loop code is made shorter here */
       /* you may want to make that loop more like this */
       increment = (Noise3( point ) + offset) * exponent_array[i];
       value += remainder * increment * value;
   }
   return( value );

} /* Hetero_Terrain() */
```

En la figura 2.15 se muestras un cubo texturizado con la función de terreno heterogéneo.

Figura 2.15 Cubo texturizado con la función de terreno heterogéneo. *(Texture, 2006).*

2.3 Aliasing

Aliasing es un término del ámbito de procesamiento de señales. Se refiere a una variedad de defectos desagradables en la reconstrucción de una señal digital como consecuencia de una limitada toma de muestreo, como se aprecia en la figura 2.16.

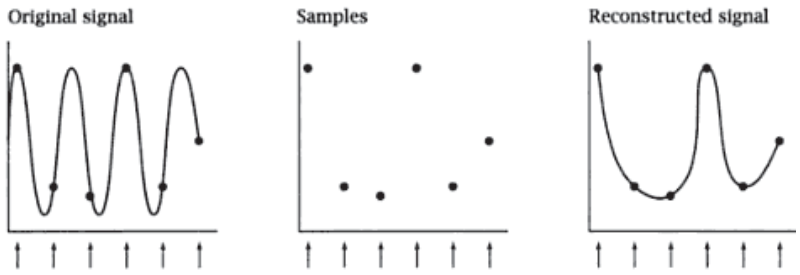


Figura* 2.16 *Muestro de una señal, en donde la reconstrucción de la señal presenta problemas de aliasing.

La cantidad de información en la señal original se le llama ancho de banda. La cantidad de información que puede ser capturado por las muestras depende de la frecuencia de muestreo (número de puntos de muestreo por unidad de distancia).

En la graficación por computadora, aliasing se refiere a una variedad de alteraciones y artefactos desagradable en una imagen que resulta de la inadecuada toma del muestreo. Que desafortunadamente las condiciones para la correcta toma de muestras y la reconstrucción no son siempre fáciles de cumplir, y cuando no se cumplen, se produce aliasing.

El aliasing visualmente se observará en una imagen con curvas y líneas inclinadas en una pantalla, pero que debido a la resolución de está, resulta incapaz de representar la curva como tal, y por tanto dichas curvas se muestran en la pantalla con el efecto visual tipo "sierra" o "escalón" no deseados.

Para evitar el aliasing es necesario contar con un proceso el cual nos ayude a minimizarlo, a este proceso se le conoce como antialiasing. *(Ebert, et al., 2002)*

2.3.1 Antialiasing

Para mejorar la distorsión de la información como consecuencia del muestreo se deben aplicar métodos de creación antialiasing que compensen el proceso del muestreo. Ya que en todos los casos, el sistema de muestreo determina la calidad potencial de la imagen que se está captando y los medios que habrá que emplear para corregir los defectos.

Con el fin de evitar la pérdida de información, se necesita establecer la frecuencia de muestreo como mínimo del doble de la frecuencia más alta que presenta la señal, a la cual se denomina frecuencia del muestreo de nyquist: f_s.

$$f_s = 2f_{máx}$$

Otra manera de expresar esto es que el intervalo de muestreo no debe ser mayor que la mitad del intervalo del ciclo (intervalo de muestreo de nyquist).

$$\Delta x_s = \frac{\Delta x_{ciclo}}{2}$$

Donde $\Delta x_{ciclo} = 1 / f_{máx}$. *(Hearn, et al., 1995 págs. 178-179)*

Desafortunadamente, siempre hay un límite práctico en la resolución de una imagen debido al espacio de memoria o a las limitaciones de pantalla, y la tasa de muestreo de una imagen es proporcional a su resolución. Por lo cual es imposible que una imagen muestre demasiado pequeños detalles para ser visible en determinadas resoluciones.

Hay otra razón por la cual el aumento de la tasa de muestreo no es nunca una solución completa para el problema de aliasing. Algunas señales tienen ancho de banda ilimitado, por lo que es difícil obtener una frecuencia máxima. Los cambios bruscos en la señal tienen componentes de alta frecuencia de manera arbitraria. No importa cuán grande sea la resolución de la imagen, el aumento de la tasa de muestreo a un valor finito no puede eliminar el aliasing de muestreo. Esta es la razón de porque líneas inclinadas son dentadas incluso en los muestra de mayor resolución.

En ocasiones el aliasing no siempre puede resolverse mediante el aumento de la tasa de muestreo, estamos obligados a encontrar la manera de eliminar las altas frecuencias de

la señal antes del muestreo. La técnica se denomina filtro paso-bajas, ya que pasa la información de baja frecuencia mientras que elimina la de mayor frecuencia. El efecto visual de un filtrado paso-bajas es difuminar la imagen. El reto consiste en borrar la imagen lo menos posible, al mismo tiempo de atenuar las frecuencias altas no deseadas. *(Ebert, et al., 2002)*

2.4 Atributos de las superficies

La elección de los atributos visuales es esencial para una representación exitosa de una superficie. En la construcción de las texturas procedurales utilicé algunos de estos atributos para obtener la apariencia deseada.

2.4.1 Color

Utilizamos el color por estética, para establecer un ambiente o estado de ánimo por cuestiones de realismo, como realce, para identificar la relación entre áreas.

Muchos colores son producidos como resultado de una combinación entre dos o más colores primarios[10] , o a través de añadiduras o mezclas.

Desde una perspectiva de la ciencia, el color es la longitud de onda dominante reflejada por un objeto de color. Es decir, el color es el producto de las longitudes de onda que son reflejadas o absorbidas por la superficie de un objeto, que al ser captadas a través de nuestros ojos y que luego son transmitidas al cerebro, quien procesa para darnos una apariencia de un cierto rango de color. *(Capilla, 2002)*

2.4.2 Uso del color

El color se utiliza en las texturas de dos maneras. En primer lugar, porque simplemente el objeto está texturizado particularmente de ese color en la realidad. Por ejemplo, se puede crear una textura de color azul brillante para cubrir la piel de un ser humano pero la piel humana en la realidad no en azul. Por lo cual se tendrá que utilizar tonos parecidos a la piel. Así que se trata de buscar semejanza en colores con el objetivo de tener realismo.

El segundo uso del color es más subjetivo y artístico. Este entra en juego cuando se tiene la libertad artística para decidir sobre los colores utilizados para determinadas sensaciones. Este uso artístico del color también puede aplicarse a grupos de colores en la vida real, sutilmente alterando las percepciones de las personas hacia las cosas, incluso si el conjunto de colores que ha utilizado en las texturas se han creado simplemente como una recreación de cómo ese objeto se parece en la realidad. Un ejemplo de esto sería si tal vez se

[10] **Colores primarios:** Son aquellos colores que no pueden obtenerse mediante la mezcla de ningún otro por lo que se consideran únicos. Los tres colores que cumplen con esta característica son: amarillo, rojo y azul. Mezclando pigmentos de éstos colores pueden obtenerse todos los demás colores.

quiere texturizar a un personaje maligno, donde es más probable el uso de colores de tonos oscuros, tales como: azules, púrpura o incluso pardos o negros para indicar una sombra más oscura de personalidad.

Por otro lado, si se quisiera obtener un personaje más angelical, se trabajaría más con una serie de colores deslumbrantes, brillantes, como el blanco y oro. Este tipo de uso del color es muy subjetivo, y puede ayudar a transmitir ideas o percepciones al usuario. *(Van, 2004)*

2.4.3 Matiz, Saturación y Brillo

Cuando se trabaja con el color, es vital comprender la forma de manipularlo para que lo podamos ajustar a nuestras necesidades. Así cuando se trabaja con imágenes en color, se tiene que tener una buena comprensión de los distintos componentes a fin de que se puedan manipular según sea necesario. *(Van, 2004)*

Estos componentes se puede dividir en: matiz, saturación y brillo.

Matiz

El matiz es una de las principales propiedades de un color, que se describe con nombres tales como: rojo, amarillo, azul, etc. En términos de graficación por computadora se conoce al matiz básicamente como el color de los píxeles de una imagen.

Así cuando se trabaja con diferentes imágenes que se desean mezclar, se adaptan los colores de las imágenes para llegar dar la tonalidad de la mezcla hecha. Se puede además hacer ajustes a las imágenes con lo cual pueden cambiar drásticamente los colores de la imagen, dándole el control sobre la apariencia de la misma. *(Van, 2004)*

Saturación

Se conoce como saturación a la cantidad de intensidad de color. Es decir la pureza del color. En mayor o menor grado de mezcla de colores las cuales modifican a un color fundamental, hace que un color esté más o menos saturado. Hablando en términos de graficación por computadora se sabe que la saturación de una imagen es la cantidad de color que contiene cada píxel y la intensidad de ese color. Si se quita toda la saturación de una imagen, esta se convierte en una imagen en escala de grises. Cuanto más alto sea el contenido gris, menor será la saturación. Es por eso que se debe utilizar correctamente la saturación ya que un problema muy común es la sobresaturación[11] de texturas, presentando un efecto desagradable en nuestra imagen. *(Van, 2004)*

Brillo

Es el grado de claridad y oscuridad de un color. También se lo conoce como el atributo de la percepción visual en el que una fuente parece ser reflejo de la luz o radiación. Más fácil de entender con el ejemplo de la diferencia entre una imagen luminosa y una imagen oscura. *(Van, 2004)*

---

[11] **Sobresaturación:** Imágenes que utilizan niveles de saturación extremos, es decir, imagen con un resultado de colores irreales o tonalidades exageradas.

2.5 Espacio de color en software grafico

Dado que los seres humanos tenemos tres tipos de sensores (conos de la retina) activos para detectar los altos niveles de iluminación. Las señales de estos tres tipos de sensores determinan la respuesta de color del observador. Por esta razón, el color natural es un fenómeno tridimensional. Para describir cuantitativamente el color se necesita usar un bien definido sistema de coordenadas en un espacio tridimensional. En los gráficos por computadora se suele utilizar los colores *rojo*, *verde* y *azul* (RGB, por sus siglas en inglés) para proporcionar tal sistema de coordenadas. Sin embargo, son infinitamente muchos de esos sistemas de coordenadas que podrían aplicarse al espacio, y ninguno de ellos es intrínsecamente superior a cualquier otro sistema. Por lo tanto, cuando tratamos con los colores, existen otros formatos o espacios de color, como HSV, LUV y CMY, que se definen con tres sistemas de coordenadas llamados ejes. *(Shirley, et al., 2005)*

2.5.1 Modelo de color RGB

El modelo de colores RGB emplea un sistema de coordenadas cartesianas. El cual se puede representar con un cubo unitario que se define con los ejes de R, G y B, como lo ilustra la figura 2.17. El origen representa el negro y el vértice de las coordenadas (1, 1, 1) es el blanco. Los vértices del cubo en los ejes representan los colores primarios y los vértices restantes representan el color complementario para cada uno de los colores primarios.

El esquema de color RGB es un modelo aditivo, es decir, se suman las intensidades de los colores primarios para producir otros colores. Cada punto de color en las fronteras del cubo se puede representar como el conjunto de tres coordenadas (R,G,B), donde a estas coordenadas se les asignan los valores en el rango de 0 a 1. *(Hearn, et al., 1995)*

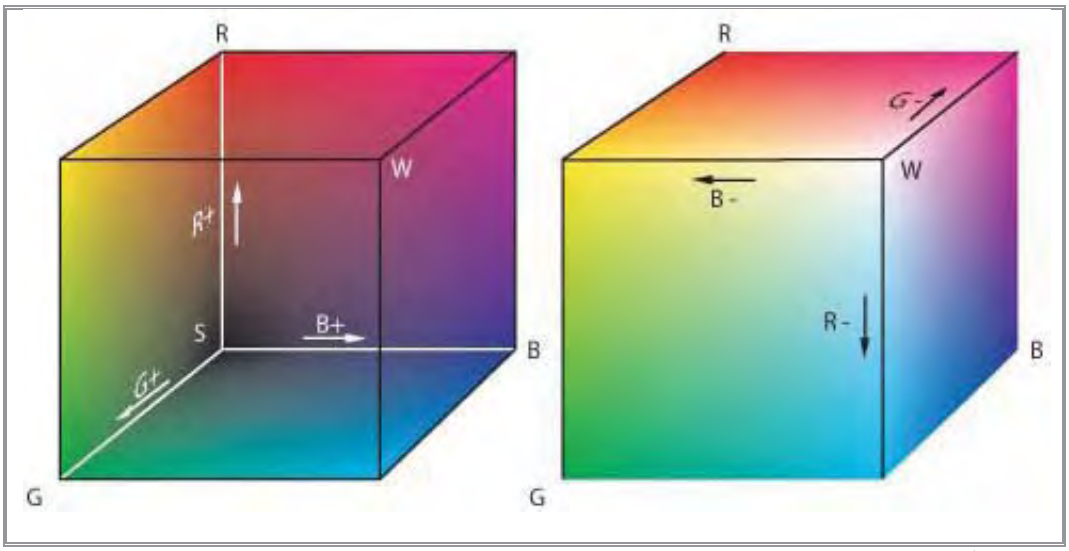


Figura 2.17 Modelo de color RGB, el cual define los colores con un proceso aditivo en el cubo unitario. *(Hearn, et al., 1995 pág. 603)*

2.5.2 Modelo de color HSV

En vez de un conjunto de colores primarios, el modelo HSV utiliza descripciones de color que tienen una aplicación más intuitiva por parte del usuario. Para dar una especificación de color, un usuario selecciona un color espectral y las cantidades de blanco y negro que se deben agregar para obtener diferentes sombras, tintes y tonos. Los parámetros de color en este modelo son matiz (H), saturación (S) y valor (V).

La representación tridimensional del modelo HSV se deriva del cubo RGB. Si se imagina que se ve el cubo a lo largo de la diagonal del vértice blanco al origen (negro), con un contorno hexagonal. La frontera del hexácono representa los diversos matices y se utiliza como la parte superior del hexácono HSV de la figura 2.18. En el hexácono, se mide la saturación a lo largo de un eje horizontal y el valor se da a lo largo de un eje vertical a través del centro del hexácono. *(Hearn, et al., 1995)*

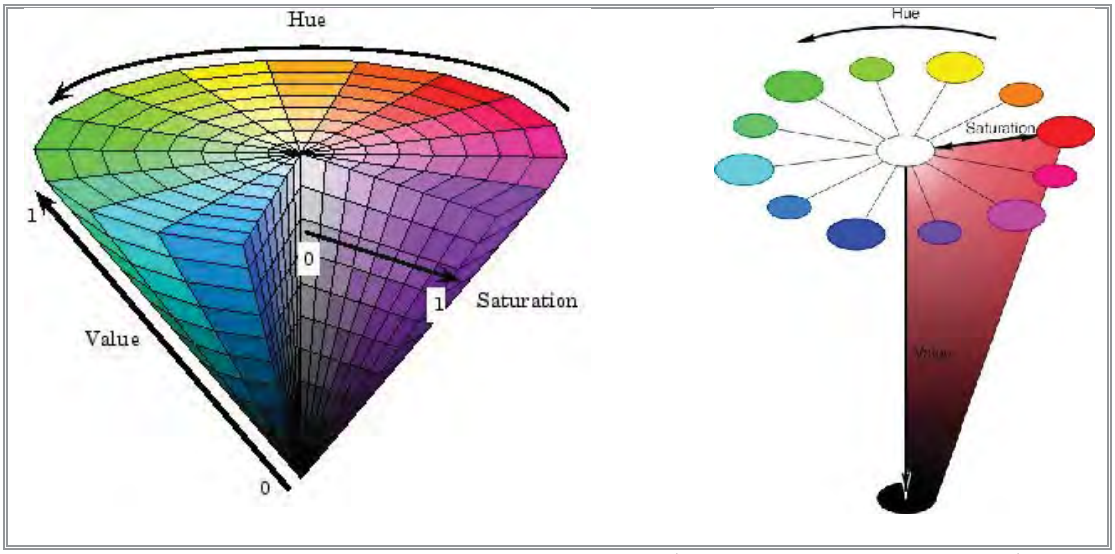


Figura 2.18 Modelo HSV representado por un hexácono. *(Hearn, et al., 1995 pág. 607)*

El matiz (H) se representa con respecto del eje vertical, en el intervalo 0° en el rojo a 360°. Los vértices en el hexágono están separados en intervalos de 60°. El amarillo está en 60°, el verde a 120° y el cian opuesto al rojo en H=180°. Los colores complementarios están a una separación de 180°, ver figura 2.19 para comprender mejor el concepto.

Figura 2.19 Imagen en donde se visualiza el matiz del modelo de color HSV. *(Hearn, et al., 1995 pág. 607)*

La saturación (S) varía de 0 a 1 y se representa en este modelo como la razón de pureza de un matiz seleccionado en su pureza máxima en S=1. Se dice que un matiz seleccionado tiene una pureza de un cuarto con el valor de S=0.25. Con S=0, tenemos la escala gris.

El valor (V) varía de 0 en el pico del hexácono a 1 en la parte superior. El pico representa el negro. Los colores tienen su intensidad máxima en la parte superior del hexácono. Cuando V=1 y S=1, tenemos matices "puros". El blanco es el punto en V=1 y S=0. En la figura 2.20 podemos visualizar el efecto que produce el cambio de saturación y valor. *(Hearn, et al., 1995)*

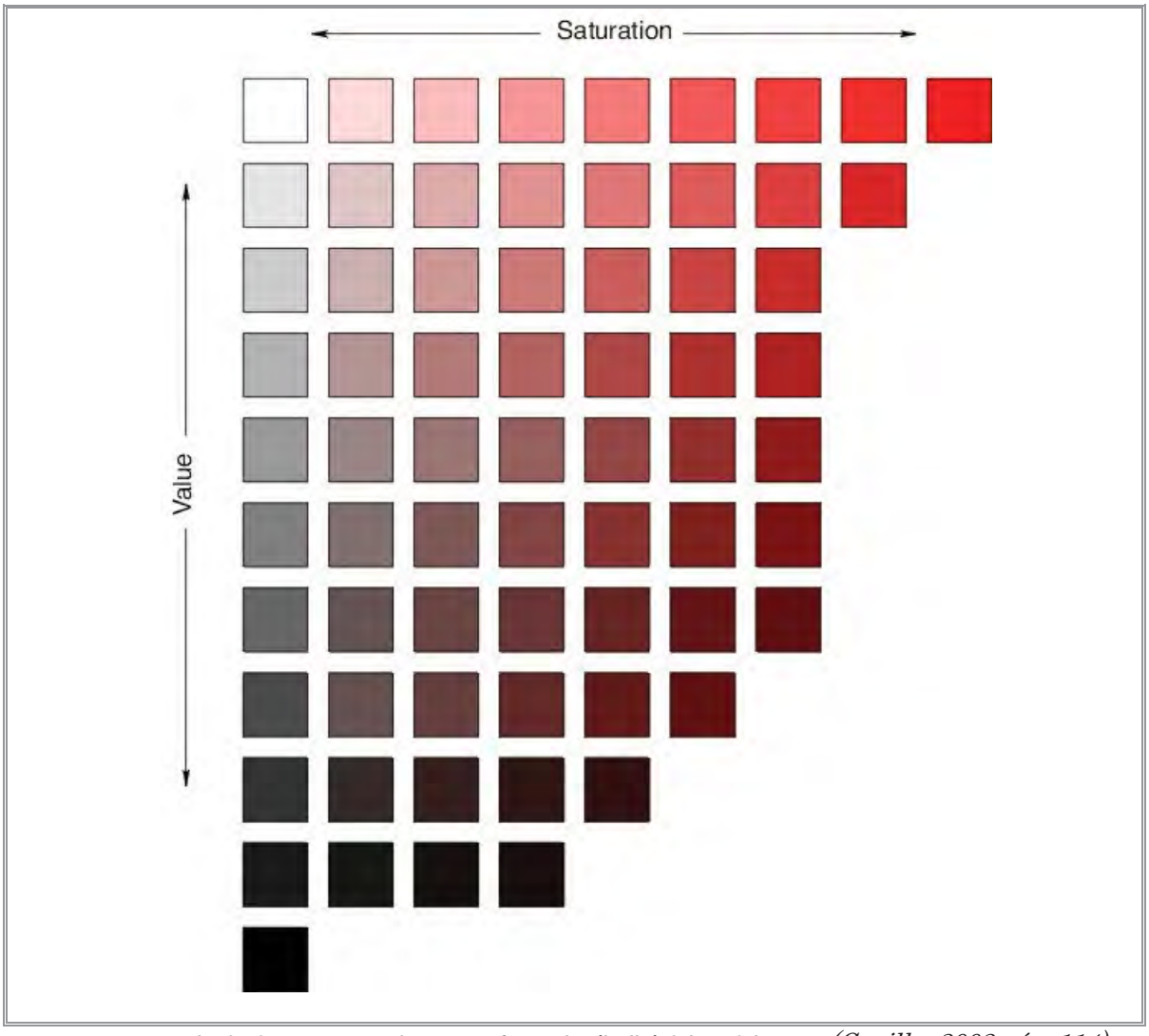


Figura 2.20 Resultado de variaciones de saturación y valor (brillo) del modelo HSV. *(Capilla, 2002 pág. 114)*

El ojo humano puede distinguir alrededor de 128 matices distintos y más o menos 130 tintes diferentes (niveles de saturación). Para cada uno de éstos, se puede detectar un número de sombras (especificaciones de valores), dependiendo del matiz que se selecciona. *(Hearn, et al., 1995)*

2.5.3 Conversión entre modelos HSV y RGB

La conversión permite a los programadores "pensar" con HSV ya que dicho modelo está más orientado a entender los matices, tonos y brillo al usuario, mientras que con el modelo RGB se dan las instrucciones a la máquina con la finalidad de ser visualizadas en un monitor de color.

Para determinar las operaciones necesarias en esta transformación, primero consideramos como se puede derivar el hexácono HSV del cubo RGB. La diagonal de este cubo del origen (negro) al blanco corresponde a un área de corte transversal hexagonal del hexácono. En cualquier corte transversal, todos los lados del hexágono y todas las líneas radiales desde el eje V hasta cualquier vértice tienen el valor V. Para cualquier conjunto de valores RGB, V equivale al valor máximo en este conjunto. El punto HSV que corresponde al conjunto de valores RGB se encuentra en el corte transversal hexagonal en el valor V. De este modo el parámetro S se determina como la distancia relativa de este punto desde el eje V. El parámetro H se determina al calcular la posición relativa del punto entre cada sextante del hexágono (ver figura 2.21). *(Foley, et al., 1996 págs. 464-471)*

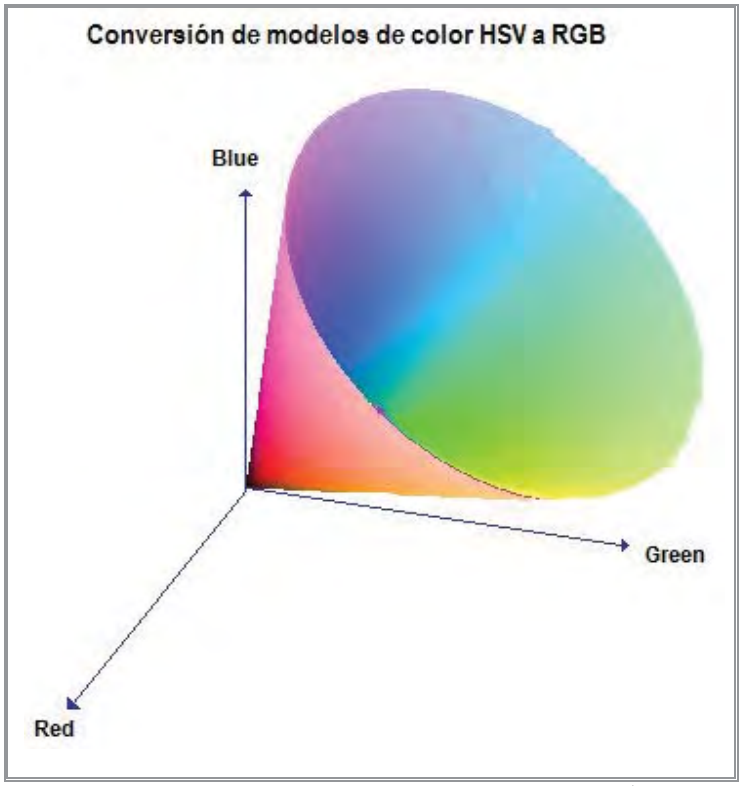


Figura 2.21 Imagen representativa de la conversión de modelos de color HSV y RGB. *(Foley, et al., 1996 pág. 470)*

Existen varios algoritmos para realizar esta conversión. Dado que no es el objetivo principal de este trabajo, no se profundizara más en este tema.

CAPÍTULO 3

3. SÍNTESIS DE TEXTURAS PROCEDURALES PARA UN SIMULADOR VIRTUAL DE CIRUGÍA

En este capítulo se explicará la programación, desarrollo, generación e integración de la síntesis de texturas procedurales, para el simulador de entrenamiento de resección transuretral de próstata. Las cuales están divididas principalmente en:

- textura del tejido de la uretra (tejido con vascularidades)
- textura del tejido interno de la próstata (apariencia amarillo-pardo)
- textura del tejido de la cápsula de la próstata (apariencia rugosa)

Se realizó primero el estudio de las imágenes médicas reales, del procedimiento RTU, obtenidas de un video médico (ver capítulo 1, sección 1.5). Posteriormente con ayuda de funciones primitivas de generación de texturas procedurales, se simuló la apariencia visual de las imágenes reales.

3.1 Funciones básicas para la generación de texturas procedurales

Se decidió utilizar la generación de texturas procedurales porque nos ayudan a mejorar y obtener las texturas que se necesita para el simulador de cirugía, tomando en cuanto los siguientes puntos:

- El ojo humano es muy perceptivo de los artefactos repetitivos que pueden ser causados por un mapeo 2D en una superficie 3D. Ya que la mayoría de los objetos en las escenas digitales no son puramente planos. Asignar intrínsecamente una textura 2D a una superficie en 3D puede ser una tarea difícil, dando lugar a distorsiones y costuras visibles. *(Jagnow, 2004)*

- Una técnica para lograr una variación espacial de las texturas de los materiales en tres dimensiones sin lugar a distorsiones o costuras visibles, es generarlas a partir de una textura sólida, ofreciendo así una representación más natural. *(Jagnow, 2004)*

- Las texturas sólidas son vitales para mantener una apariencia constante realista y detallada, ya que los objetos "aparecerán" esculpidos en una sustancia sólida. *(Ebert, et al., 2002)*

El uso de métodos procedurales para la generación de texturas en este trabajo de tesis, nos ayudará a lograr una apariencia y aspecto visual que se requieren, logrando con ello obtener una textura continua en la cual se podrá modelar la glándula prostática; teniendo además la ventaja de contar con una textura que nos dará la apariencia de profundidad al realizar los cortes sobre el tejido, apareciendo las condiciones visuales de las texturas que se encuentran en su interior, que al final el usuario lo visualizará como se aprecia en el procedimiento real.

Se tendrá además la posibilidad de contar con las funciones procedurales base, las cuales nos ayudaran a extender el resultado a imagen más grandes, logrando con ello generar un modelo de síntesis de textura útil, visto y programable para el simulador virtual de cirugía.

3.1.1 Funciones procedurales básicas

El desarrollo de los algoritmos y las funciones básicas para la generación de texturas, se tomaron principalmente del libro Texturing and Modeling a Procedural Approach. *(Ebert, et al., 2002)*

Obteniendo las siguientes funciones procedurales que se explicarán y mostrarán a continuación.

3.1.1.1 Función de ruido

Para generar texturas procedurales irregulares lo primero que necesitamos es contar con una función primitiva irregular normalmente llamada *ruido*.

Las propiedades de una función de ruido ideal son:

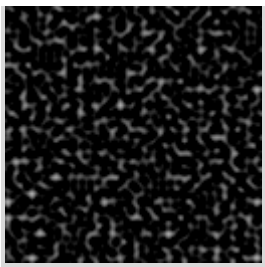
- Ser una función pseudo-repetible en sus entradas.
- Tener un rango conocido, que normalmente vaya de -1 a 1.
- Tener una banda limitada, con una frecuencia máxima de aproximadamente 1.
- No presentar periodicidad evidente o patrones regulares. Tales funciones pseudo-aleatorias son siempre periódicas, pero la periodicidad puede ser muy larga y por lo tanto no es visible.
- Ser una función estacionaria, es decir, su carácter estadístico debe ser de traslación invariante. *(Ebert, et al., 2002 pág. 68)*

A continuación se presenta parte del código fuente de la función de ruido de Perlin[12], útil en la generación de texturas procedurales, el cual visualmente agrega a la textura una apariencia más compleja.

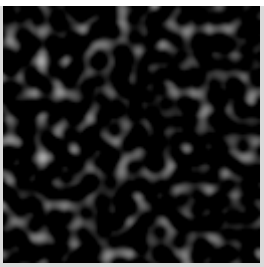
```
/* coherent noise function 2 dimension */
/* (copyright Ken Perlin) */


float noise2(float vec[2])
{
        int bx0, bx1, by0, by1, b00, b10, b01, b11;
        float rx0, rx1, ry0, ry1, *q, sx, sy, a, b, t, u, v;
        register i, j;

        if (start) {
                start = 0;
                init();
        }

        setup(0, bx0,bx1, rx0,rx1);
        setup(1, by0,by1, ry0,ry1);

        i = p[ bx0 ];
        j = p[ bx1 ];

        b00 = p[ i + by0 ];
        b10 = p[ j + by0 ];
        b01 = p[ i + by1 ];
        b11 = p[ j + by1 ];

        sx = s_curve(rx0);
        sy = s_curve(ry0);

#define at2(rx,ry) ( rx * q[0] + ry * q[1] )
```

[12] **Código fuente de la función de ruido de Perlin:** Para consultar el código completo se puede visitar la siguiente referencia web. *(Noise Perlin, 1997)*

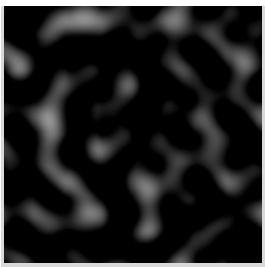
```
        q = g2[ b00 ] ; u = at2(rx0,ry0);
        q = g2[ b10 ] ; v = at2(rx1,ry0);
        a = lerp(sx, u, v);

        q = g2[ b01 ] ; u = at2(rx0,ry1);
        q = g2[ b11 ] ; v = at2(rx1,ry1);
        b = lerp(sx, u, v);

        return lerp(sy, a, b);
}
```



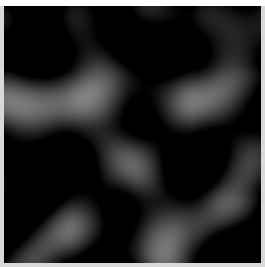
a) Escala = 4



b) Escala = 8



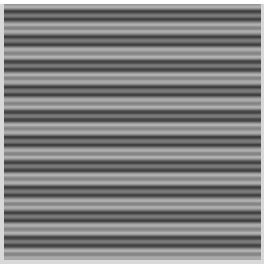
c) Escala = 16



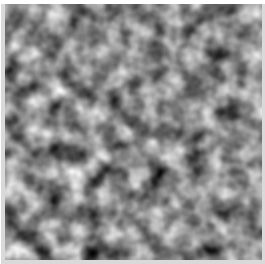
d) Escala = 32

a)Textura uniforme

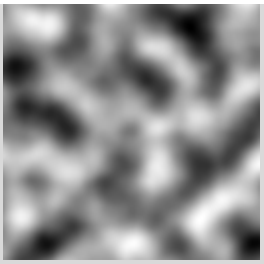
b)Textura + función de ruido (parámetros de entrada aleatorios)



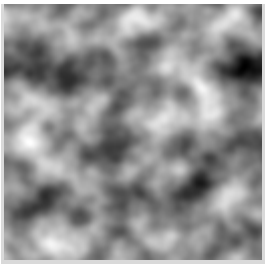
c) Textura + función de ruido (parámetros de entrada fijos y evaluados con una función trigonométrica)



a) escala = 10
h = 1.12
lacunaridad = 1.74
octavas = 3.45



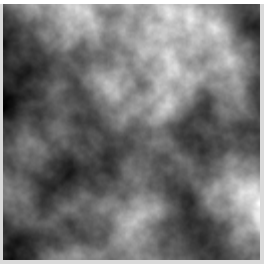
b) escala = 22
h = 2.8
lacunaridad = 1.74
octavas = 9.80



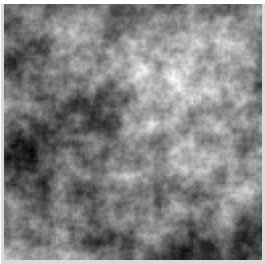
c) escala = 22
h = 1.2
lacunaridad = 1.74
octavas = 3.45

d) escala = 22
h = 2.8
lacunaridad = 10
octavas = 3.45

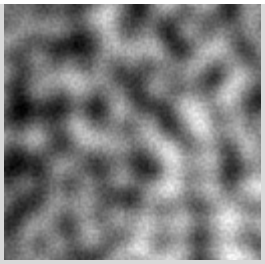
a) h = 1
lacunaridad = 2.5
offset = 0.8
octavas = 1.2



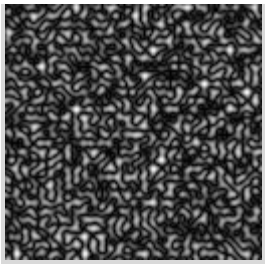
b) h = 1
lacunaridad = 2.5
offset = 0.8
octavas = 9.6



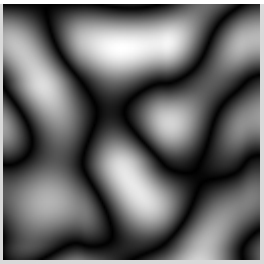
c) h = 1
lacunaridad = 2.5
offset = 9.0
octavas = 4.6



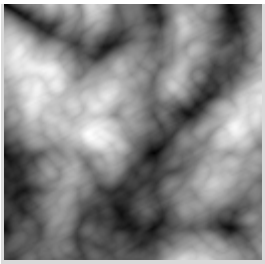
d) h = 8.5
lacunaridad = 2.5
offset = 1.5
octavas =4.6



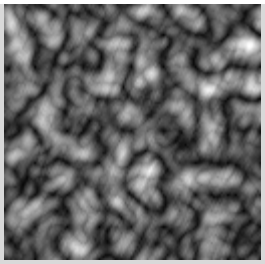
a) escala = 38
octava inicial = 4
octavas = 5.5



b) escala = 322
octava inicial = 4
octavas = 5.5

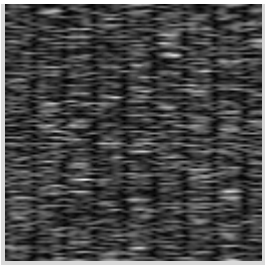


c) escala = 150
octava inicial = 1
octavas = 10.5



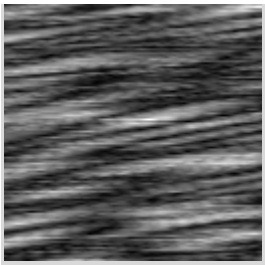
d) escala = 128
octava inicial = 4
octavas = 9

```
Function Turbulencia_modificada = tex_turb(xmax, ymax, scale, initoctave,
octaves)

      fx = xmax/scale;    %64
      fy = ymax/scale;    %64
      [p, g1, g2, g3] = init_noise;


      for x=1:xmax
         for y=1:ymax
            point(1) = fx*(x/xmax);
            point(2) = fy*(y/ymax)* 0.1; %%  Multiplicación de valor
                                         %%  constante
            Turbulencia_mod = turb(point, octaves, initoctave, p, g2, R);
            T(x,y) = turbulencia_mod;
         end
      end
```

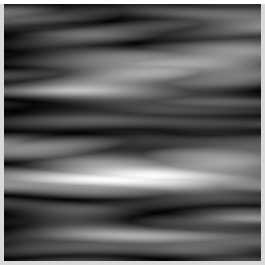


a) escala = 6
octava inicial = 3
octavas = 4.5

b) escala = 210
octava inicial = 3
octavas = 4.5

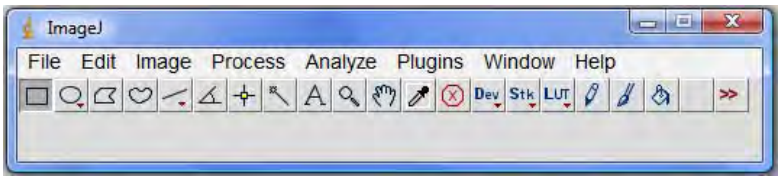


c) escala = 115
octava inicial = 6
octavas =10.5



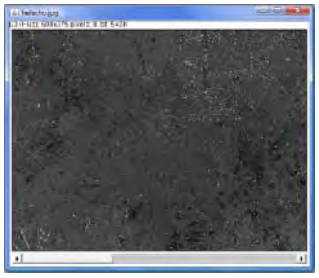
d) escala = 64
octava inicial = 3
octavas = 4.6

13



Programa ImageJ

Imagen en RGB

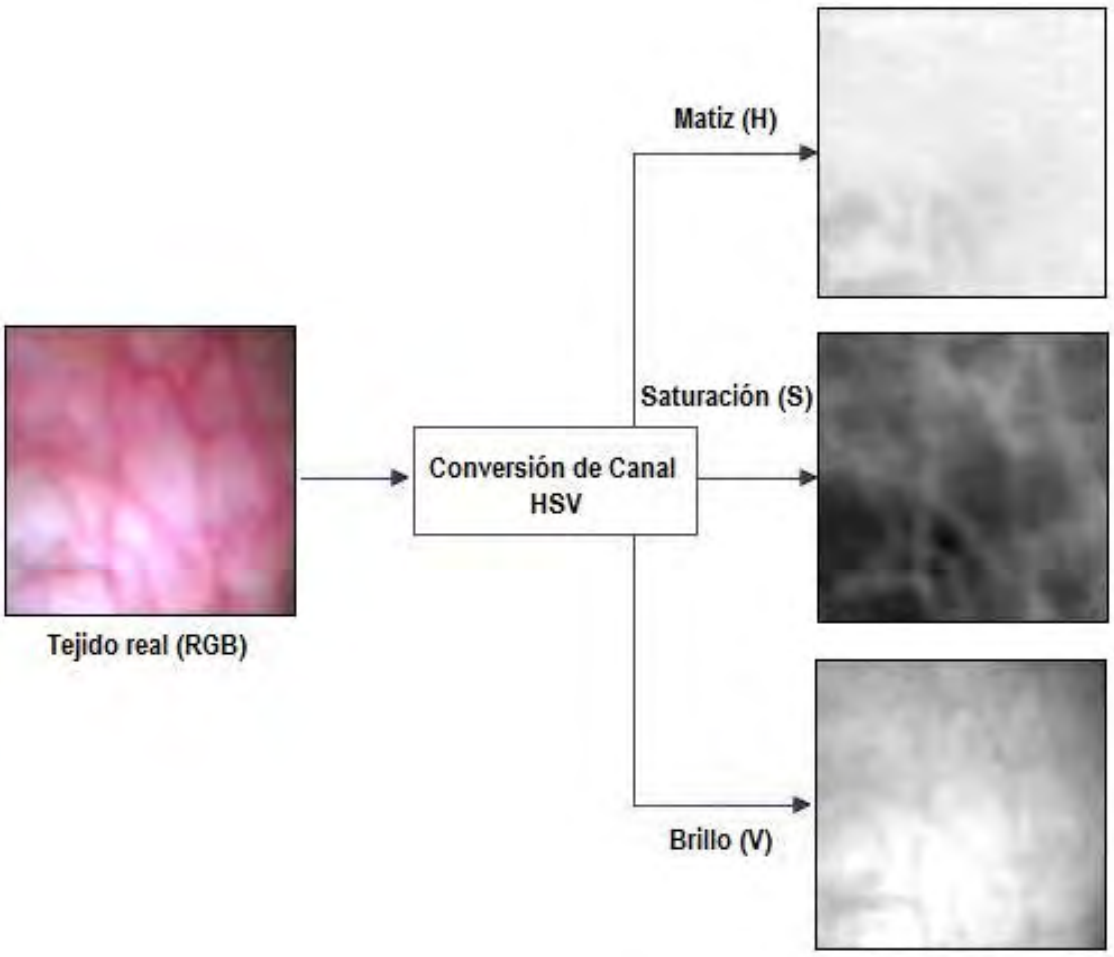


Matiz

Saturación

Valor

Imagen en HSV


Matiz (H)
Saturación (S)
Conversión de Canal HSV
Brillo (V)
Tejido real (RGB)

Una vez analizadas las tres imágenes en escala de grises del modelo HSV de la imagen real, se procede a desarrollar la síntesis de esta textura con el uso de las tres funciones procedurales básicas: *terreno heterogéneo, turbulencia y fBm*.

Cabe mencionar que la selección de dichas funciones procedurales, se llevo a cabo mediante el análisis de los resultados previos obtenidos con los algoritmos de cada función, siendo estas las que más cumplen con las características del tejido real que se desea sintetizar, así como el uso de los valores de sus parámetros con los cuales obtenemos dichos resultados. Por ejemplo, la función de turbulencia nos proporciona las características de venas o vascularidades dentro de la textura, la cual indudablemente nos será útil para darle dicha apariencia a nuestra textura sintética.

Para profundizar más en el desarrollo y generación de la textura vascular, se presenta el siguiente diagrama de flujo, en el cual se encontrará la explicación de la integración de las tres funciones procedurales básicas, para obtener la textura final deseada.

3.3.1.1 Diagrama de flujo para la generación de textura con vascularidades

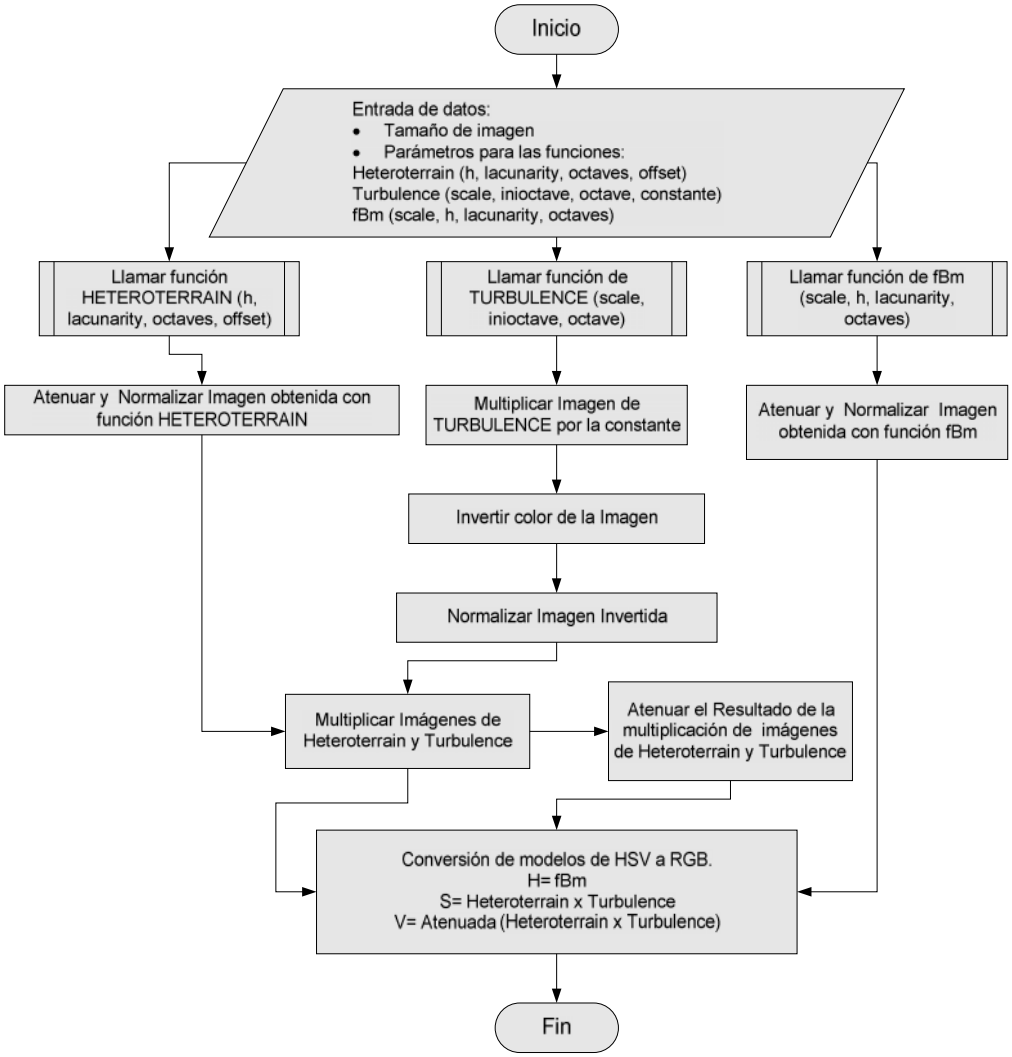
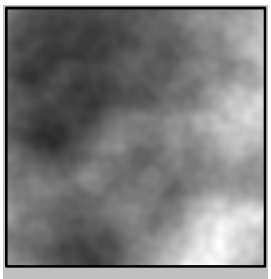
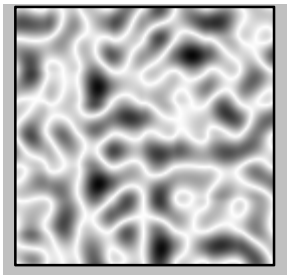
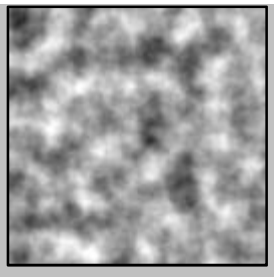


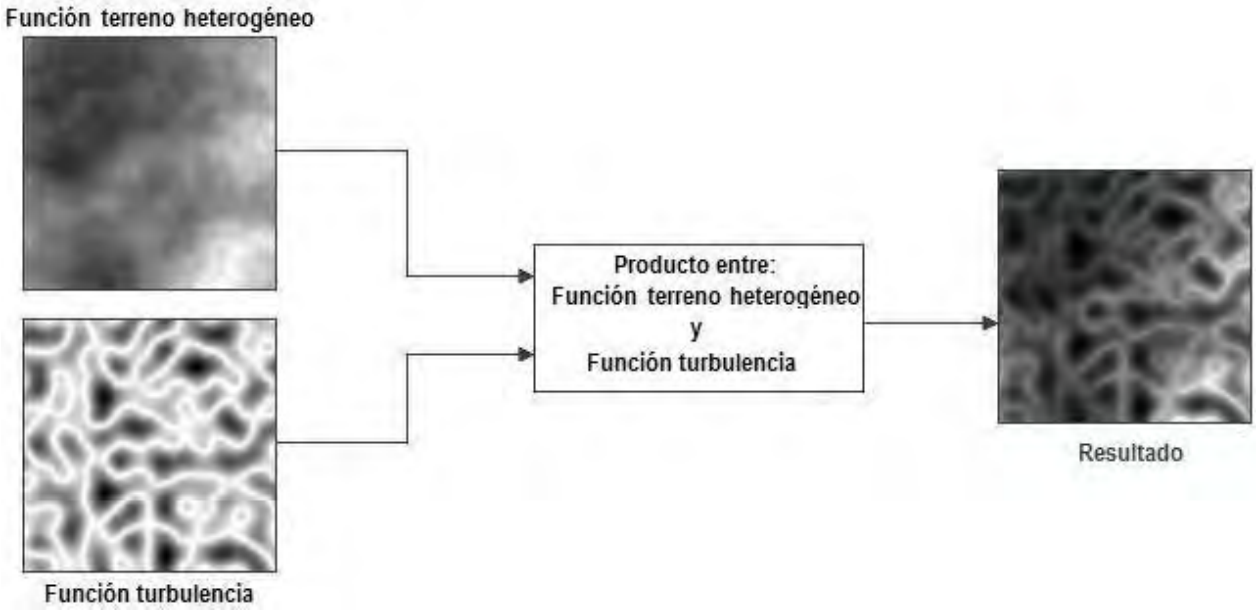
Diagrama Flujo

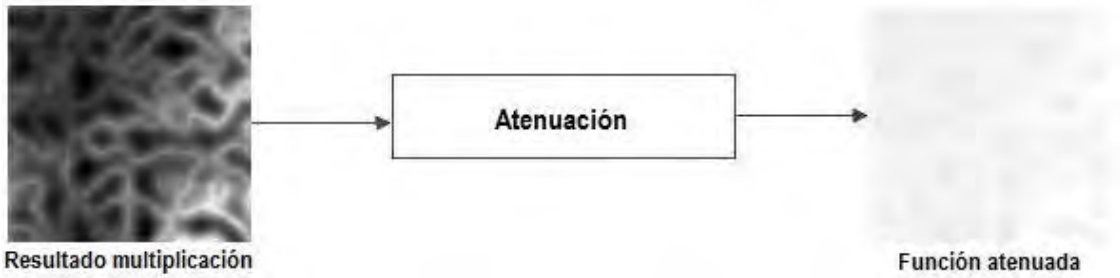
Parámetros *	Textura
H=1.2 Lacunaridad=1.9 Offset=1.5 Número octavas=4.63	

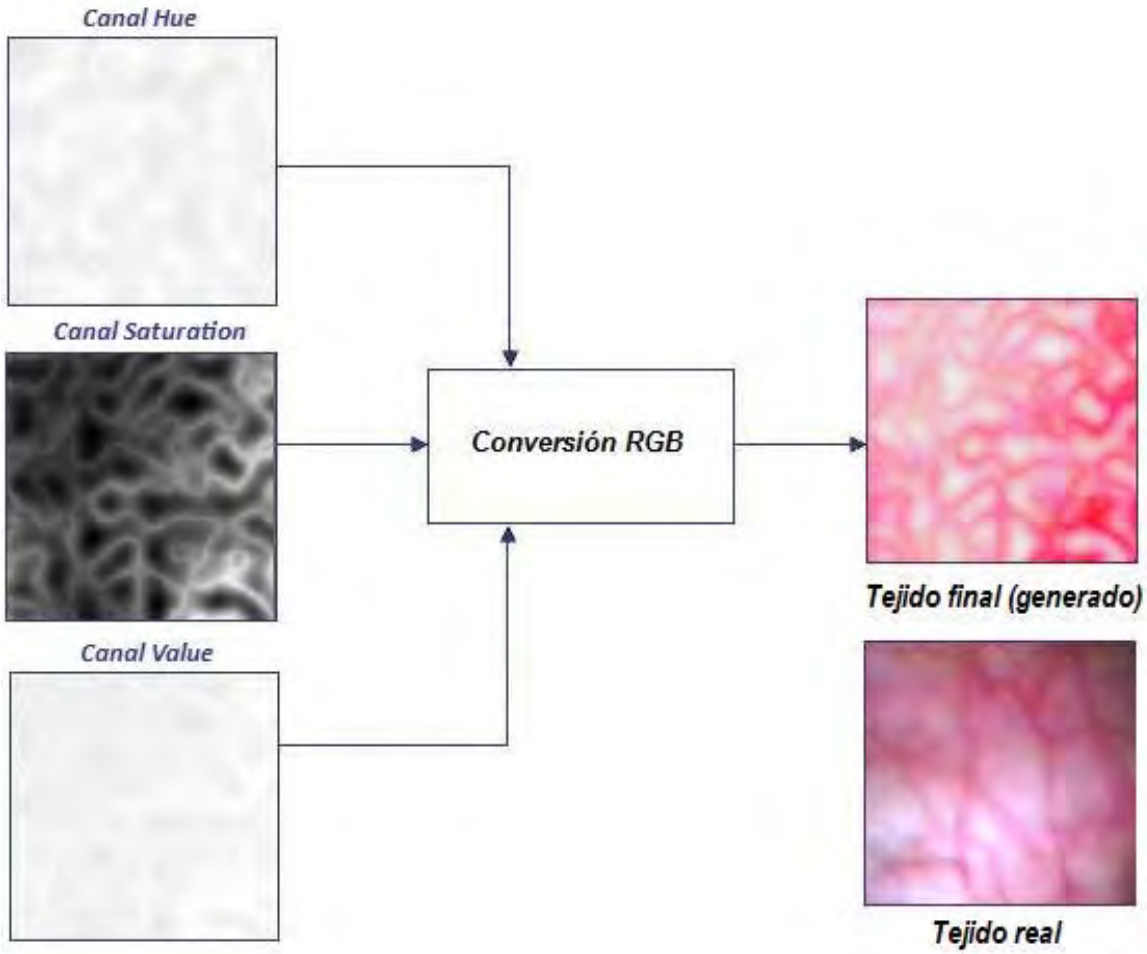
Parámetros *	Textura
Escala=128 Octava inicial=4 Octava final=4.87 Factor=3.24	

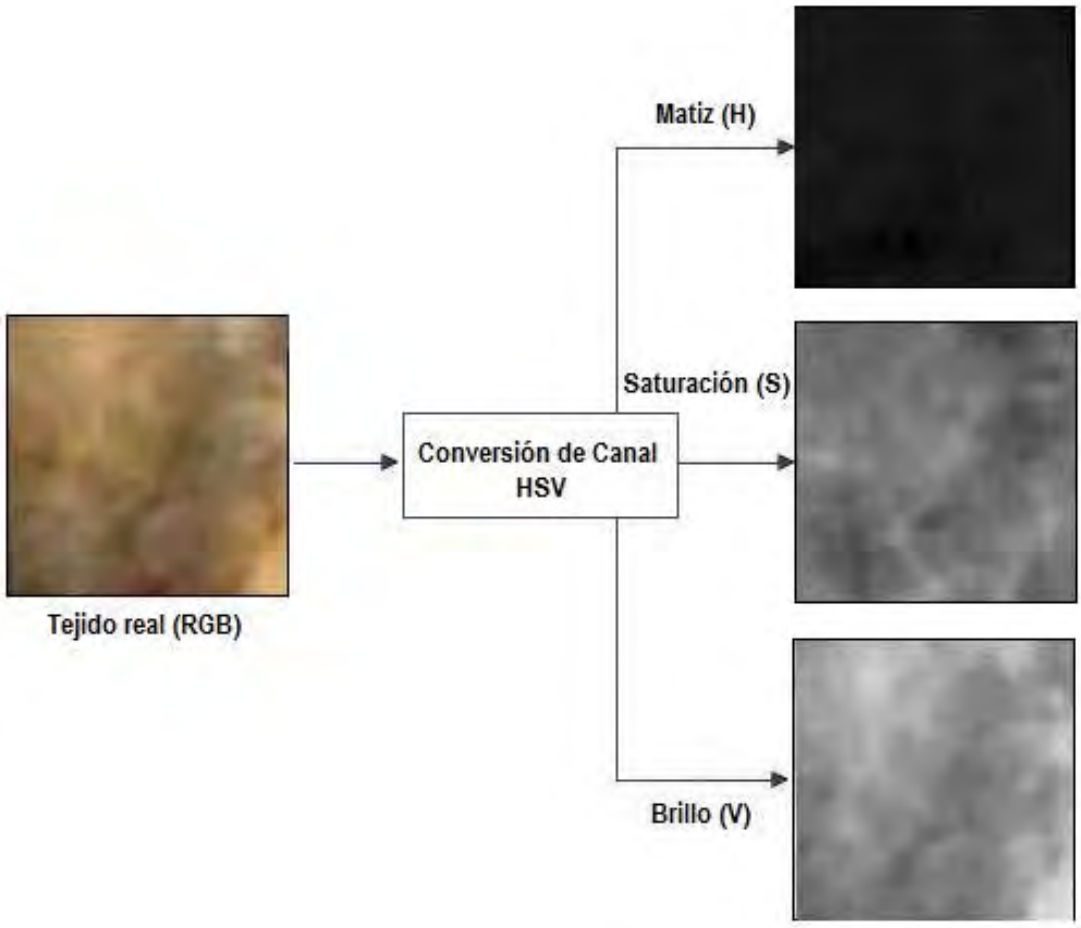
Parámetros	Textura
Escala=16 H=1.12 Lacunaridad=2 Número octavas=3.52	

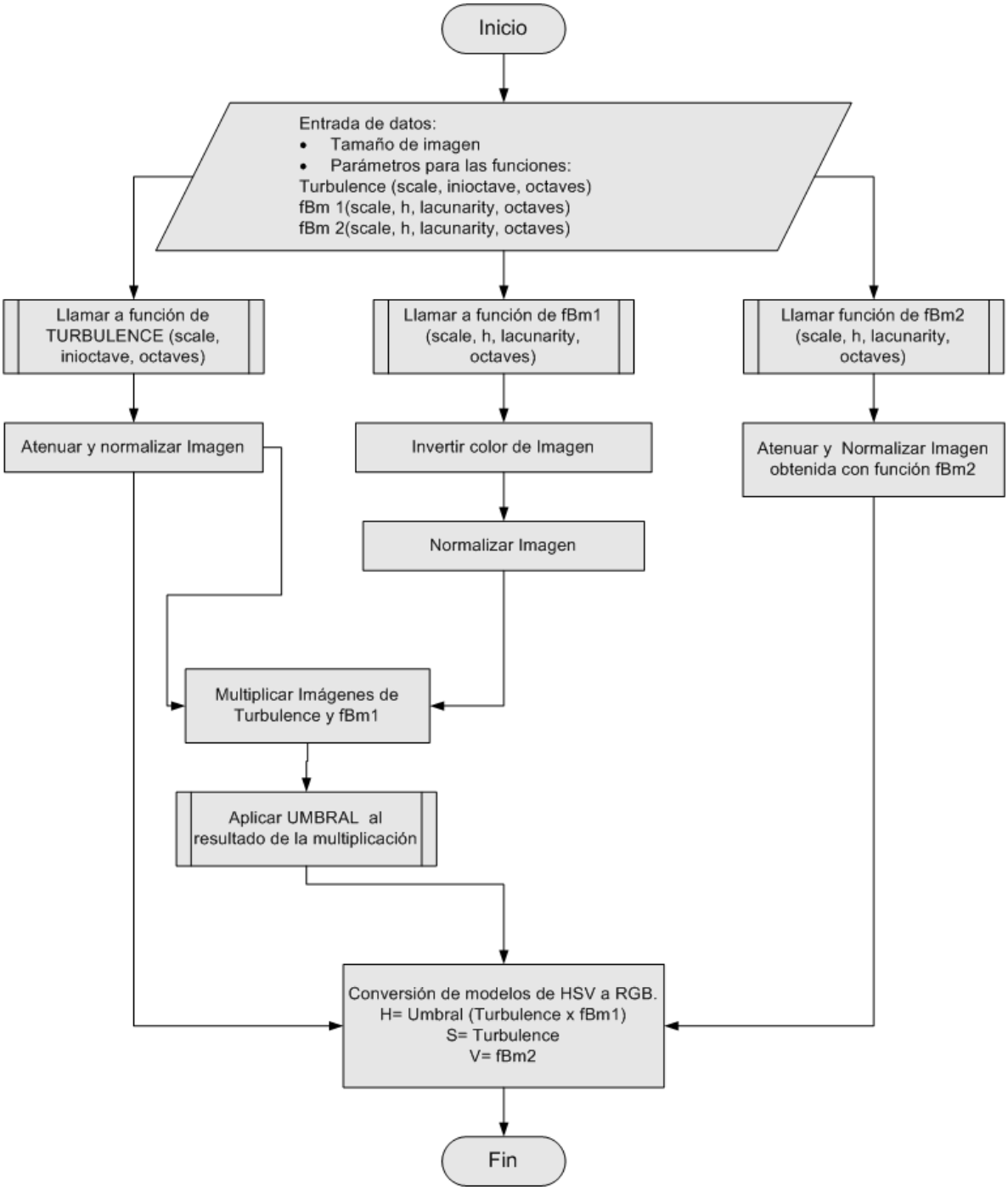
Nota: Se modifico el contraste de la imagen con fines de impresión de la tesis, ya que la visualización de esta textura es un color blanco con pequeñas manchas en gris claro, con lo cual dificulta su apreciación visual.


Función terreno heterogéneo
Producto entre:
Función terreno heterogéneo
y
Función turbulencia
Resultado
Función turbulencia


Atenuación
Resultado multiplicación
Función atenuada


Canal Hue
Canal Saturation
Canal Value
Conversión RGB
Tejido final (generado)
Tejido real


Matiz (H)
Saturación (S)
Conversión de Canal HSV
Tejido real (RGB)
Brillo (V)


Inicio
Entrada de datos:
• Tamaño de imagen
• Parámetros para las funciones:
Turbulence (scale, inioctave, octaves)
fBm 1(scale, h, lacunarity, octaves)
fBm 2(scale, h, lacunarity, octaves)
Llamar a función de TURBULENCE (scale, inioctave, octaves)
Llamar a función de fBm1 (scale, h, lacunarity, octaves)
Llamar función de fBm2 (scale, h, lacunarity, octaves)
Atenuar y normalizar Imagen
Invertir color de Imagen
Atenuar y Normalizar Imagen obtenida con función fBm2
Normalizar Imagen
Multiplicar Imágenes de Turbulence y fBm1
Aplicar UMBRAL al resultado de la multiplicación
Conversión de modelos de HSV a RGB.
H= Umbral (Turbulence x fBm1)
S= Turbulence
V= fBm2
Fin

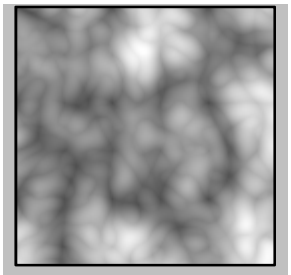
Parámetros

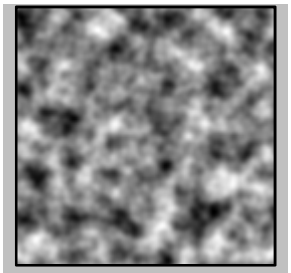
Escala=96

Octava inicial=2

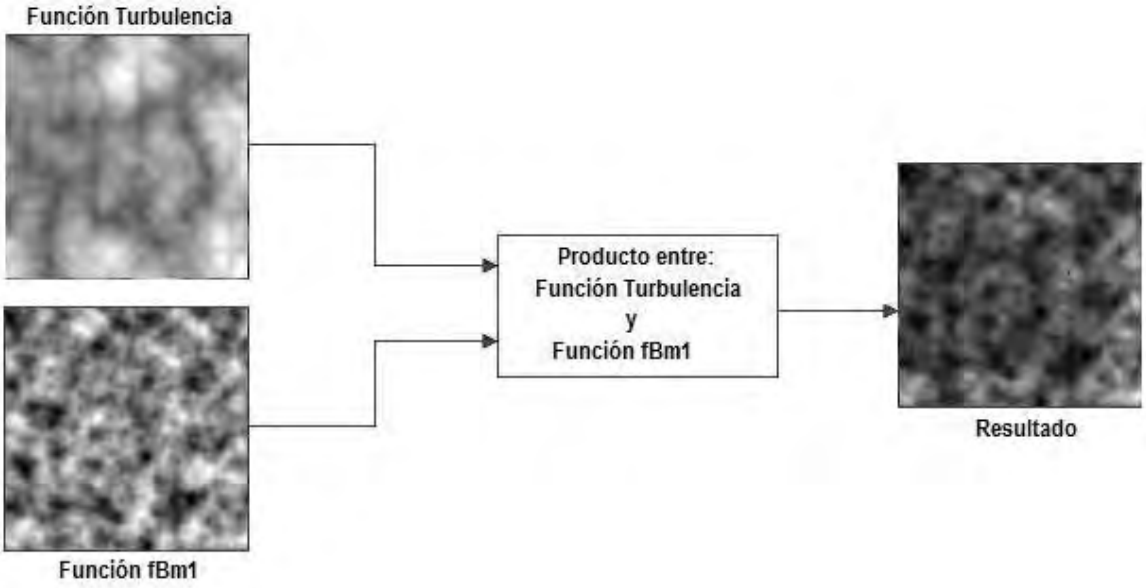
Número octavas=4.5

Textura

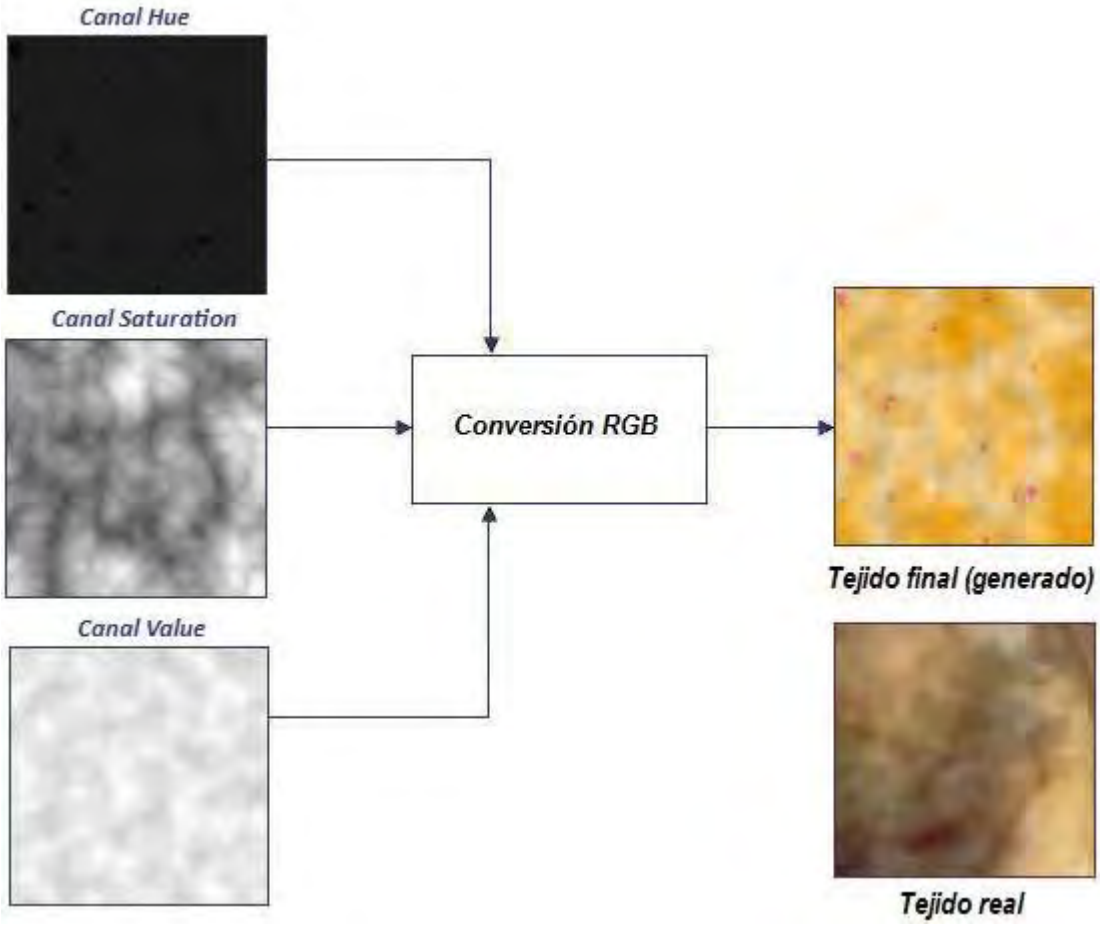


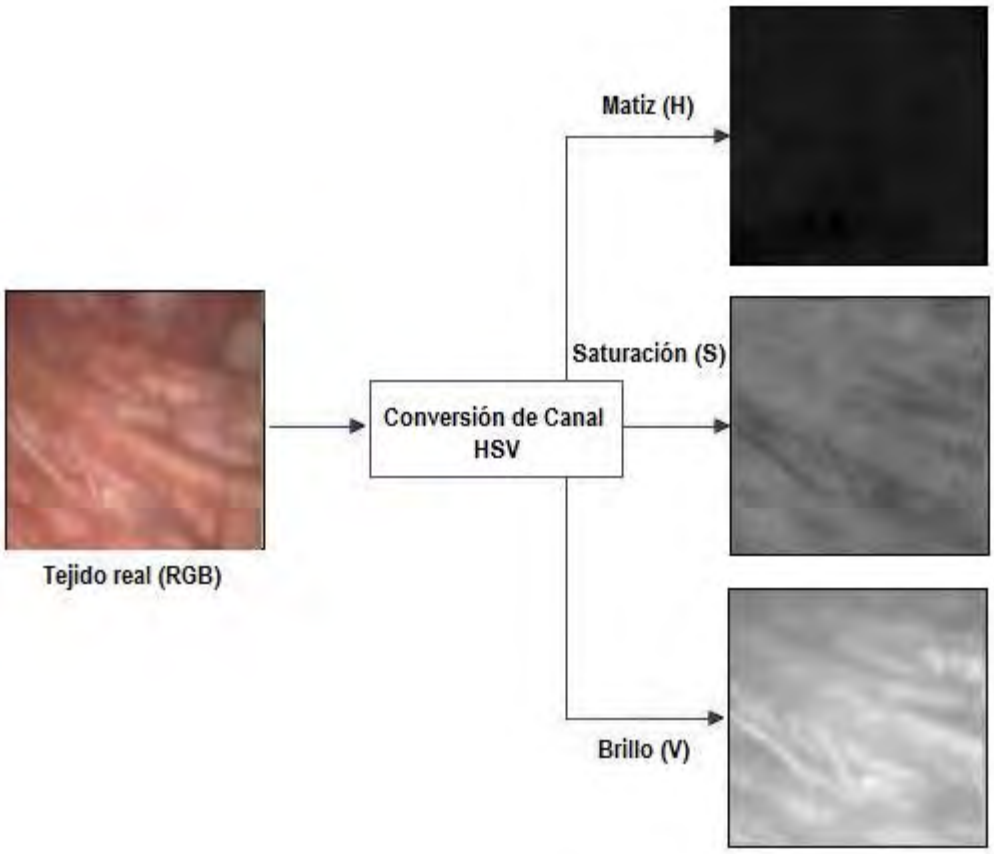
Parámetros	Textura
Escala=16 H=0.5 Lacunaridad=2 Número octavas=2.52	

Parámetros	Textura
Escala=16 H=1.12 Lacunaridad=2 Número octavas=3.52	


Función Turbulencia
Función fBm1
Producto entre:
Función Turbulencia
y
Función fBm1
Resultado


Resultado multiplicación
Umbral
(Resultado multiplicación)
Resultado umbral


Canal Hue
Canal Saturation
Canal Value
Conversión RGB
Tejido final (generado)
Tejido real


Matiz (H)
Saturación (S)
Conversión de Canal HSV
Tejido real (RGB)
Brillo (V)

3.3.3.1 Diagrama de flujo para la generación de textura del tejido cápsula

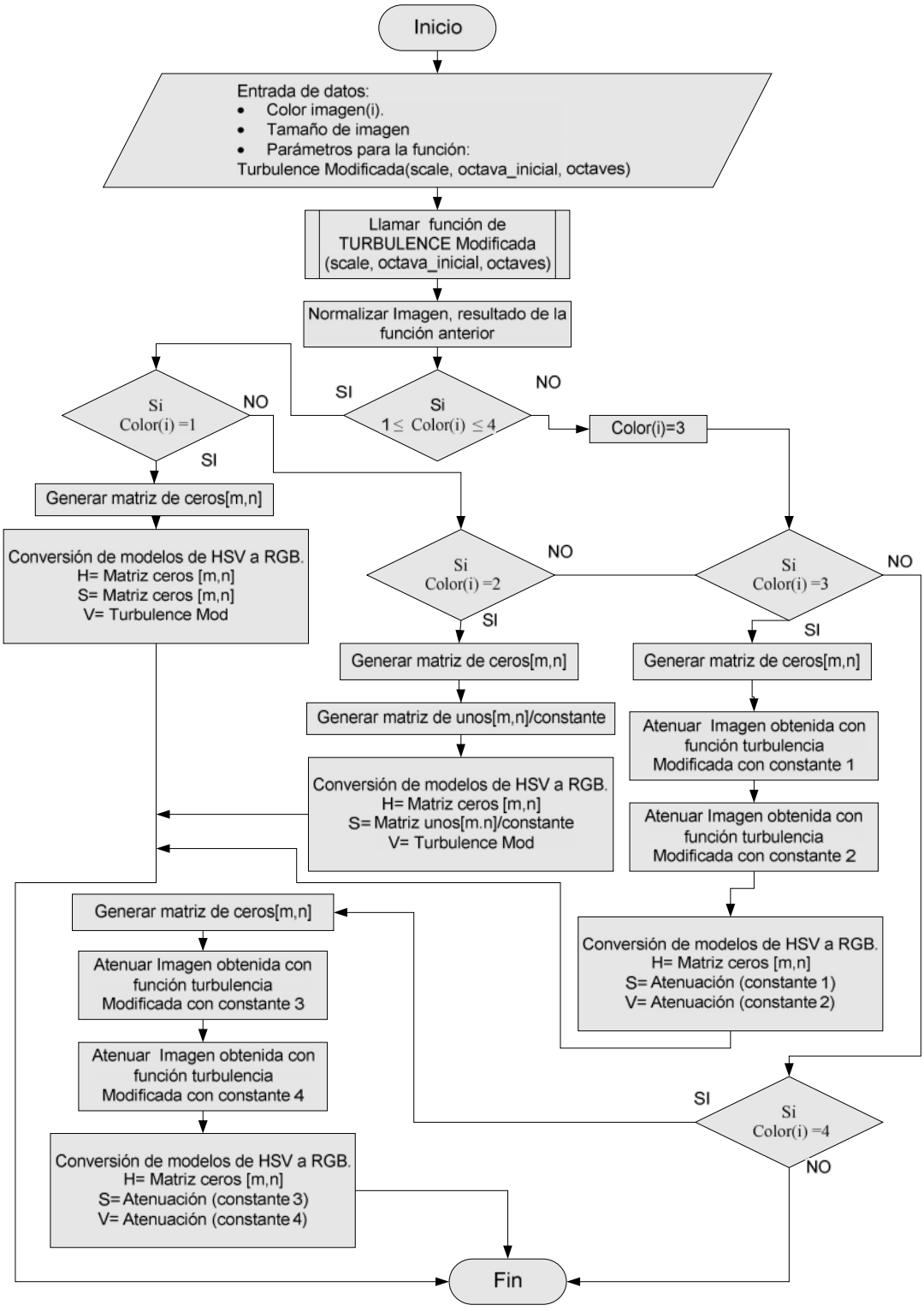
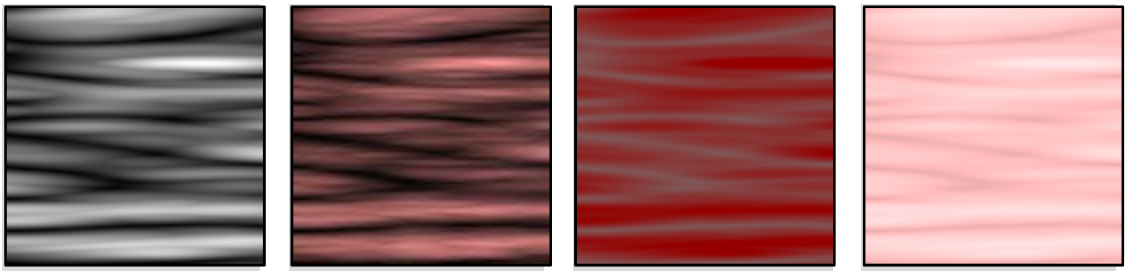
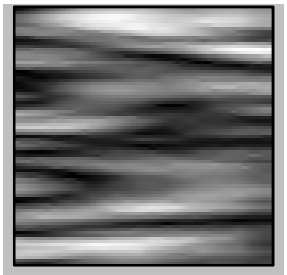
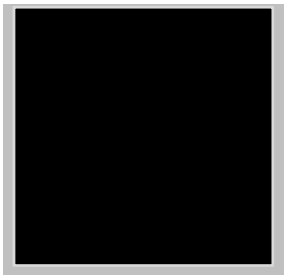


Diagrama Flujo →Tejido Rugoso

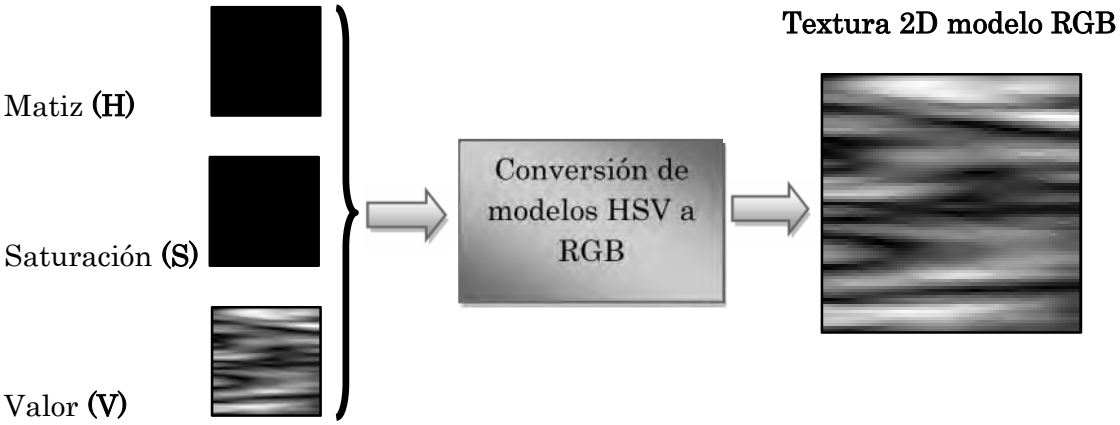


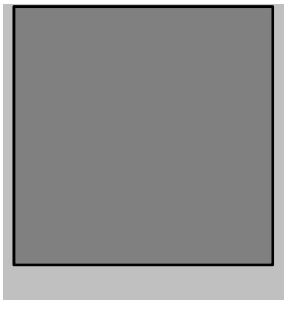
a) Color = 1 b) Color = 2 c) Color = 3 d) Color = 4

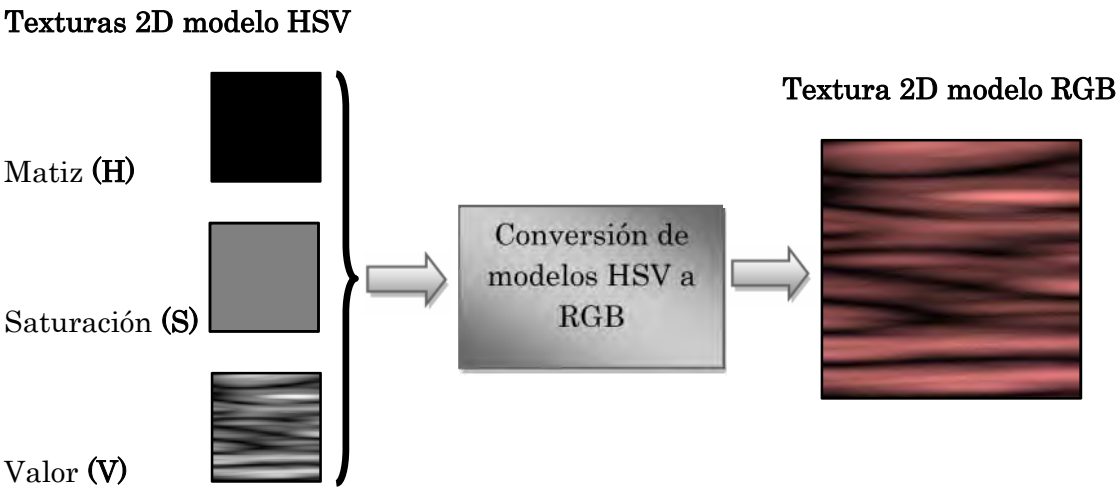
Parámetros	Textura
Escala=64 Octava inicial=3 Número octava=4.5	

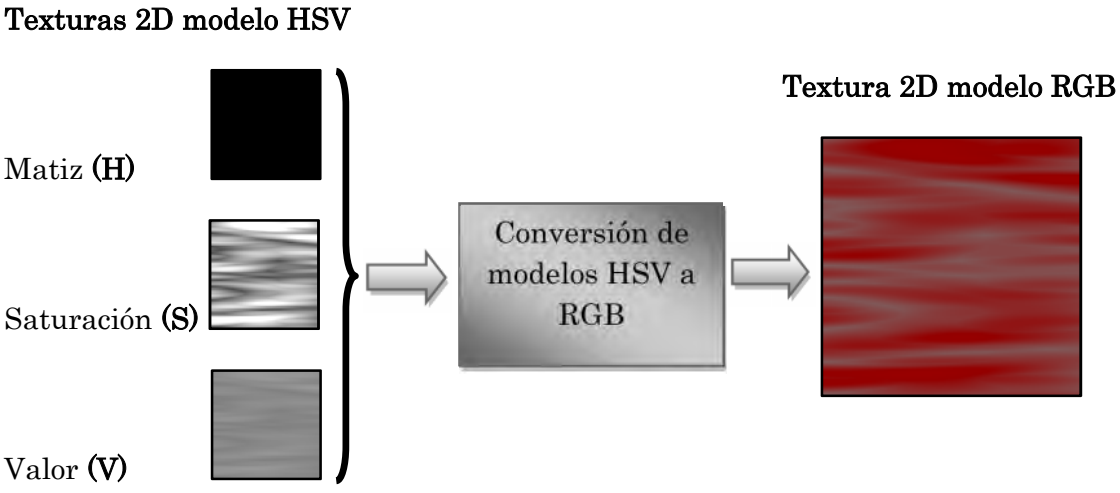
Parámetros	Textura
Matriz Ceros	

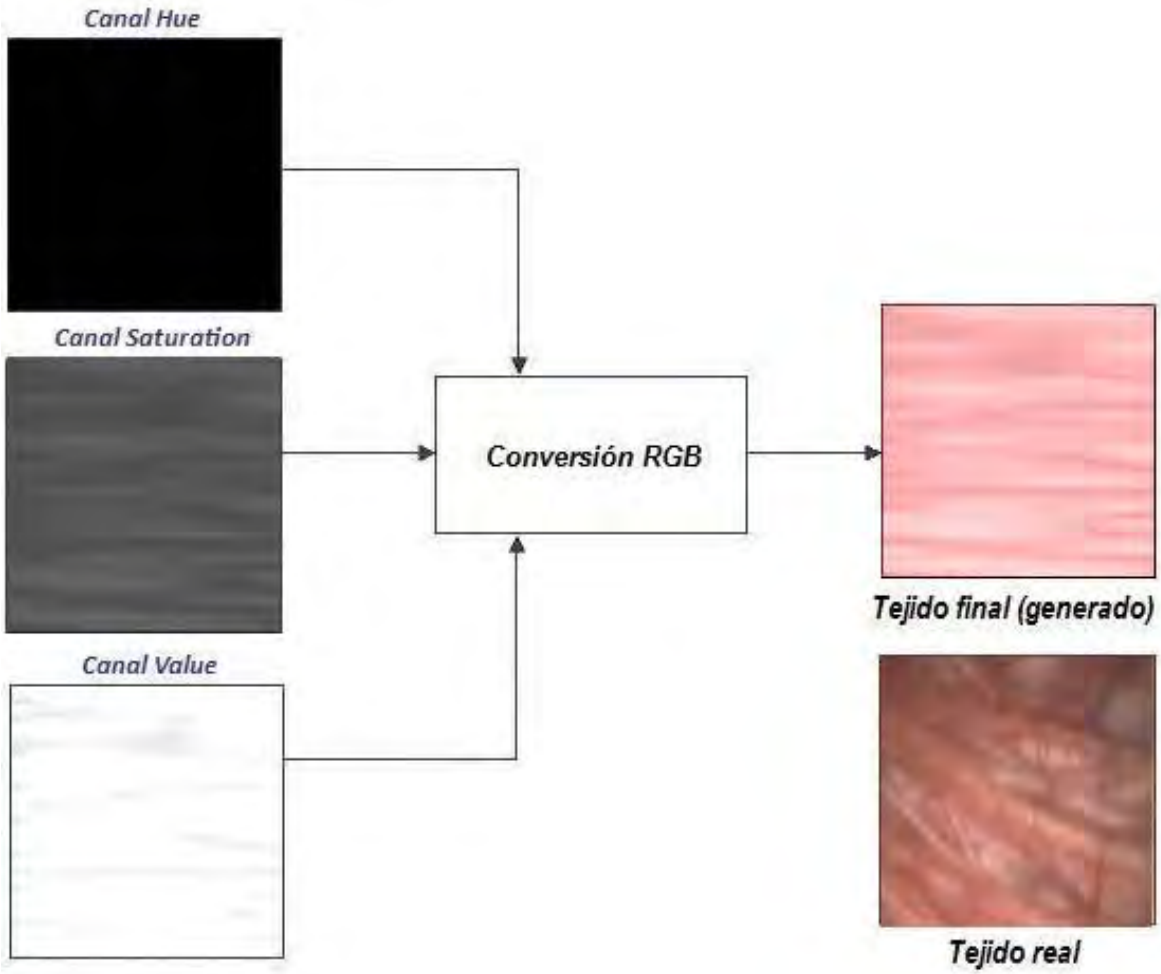
Texturas 2D modelo HSV



Parámetros	**Textura**
(Matriz Unitaria)/ Factor	


Texturas 2D modelo HSV
Textura 2D modelo RGB
Matiz (H)
Saturación (S)
Valor (V)
Conversión de modelos HSV a RGB


Texturas 2D modelo HSV
Textura 2D modelo RGB
Matiz (H)
Saturación (S)
Valor (V)
Conversión de modelos HSV a RGB


Canal Hue
Canal Saturation
Canal Value
Conversión RGB
Tejido final (generado)
Tejido real

a)

b)

En las imágenes *a y b,* encontramos problemas de continuidad en el mapeado ya que son evidentes las costuras sobre la geometría de la superficie 3D cuando se utilizan texturas 2D.

c) Se puede apreciar las ventajas del uso de una textura sólida (textura 3D), con la cual podemos obtener una vista interior del material, teniendo continuidad y por ende obtener una apariencia más real.

Teniendo como antecedentes las ventajas que ofrecen las texturas 3D, se considera la necesidad de obtener una textura volumétrica para ser utilizada en el simulador virtual de cirugía de próstata, ya que está englobará todas las características visuales que se necesitan para el simulador, es decir, se contará con una continuidad del tejido, con efectos sólidos, permitiendo al usuario llevar a cabo cortes y el efecto de cauterización sobre el tejido al momento de usar el resectoscopio, obteniendo resultados que no perderán la continuidad del material teniendo la posibilidad de realizar vistas del modelo desde su interior sin que con ello se pierda una apariencia visual real.

Para lograr esto se extenderán los resultados ya obtenidos y probados de las funciones procedurales en 2D, explicadas anteriormente en este mismo capítulo. Serán ahora generadas en tres dimensiones logrando con ello la obtención de una textura volumétrica unificada que contendrá las tres texturas: textura del tejido de la uretra (vascularidades), textura del tejido interno de la próstata (amarillo-pardo) y textura del tejido de la capsula de la próstata (apariencia rugosa); con las cuales se formarán los contornos de la glándula prostática.

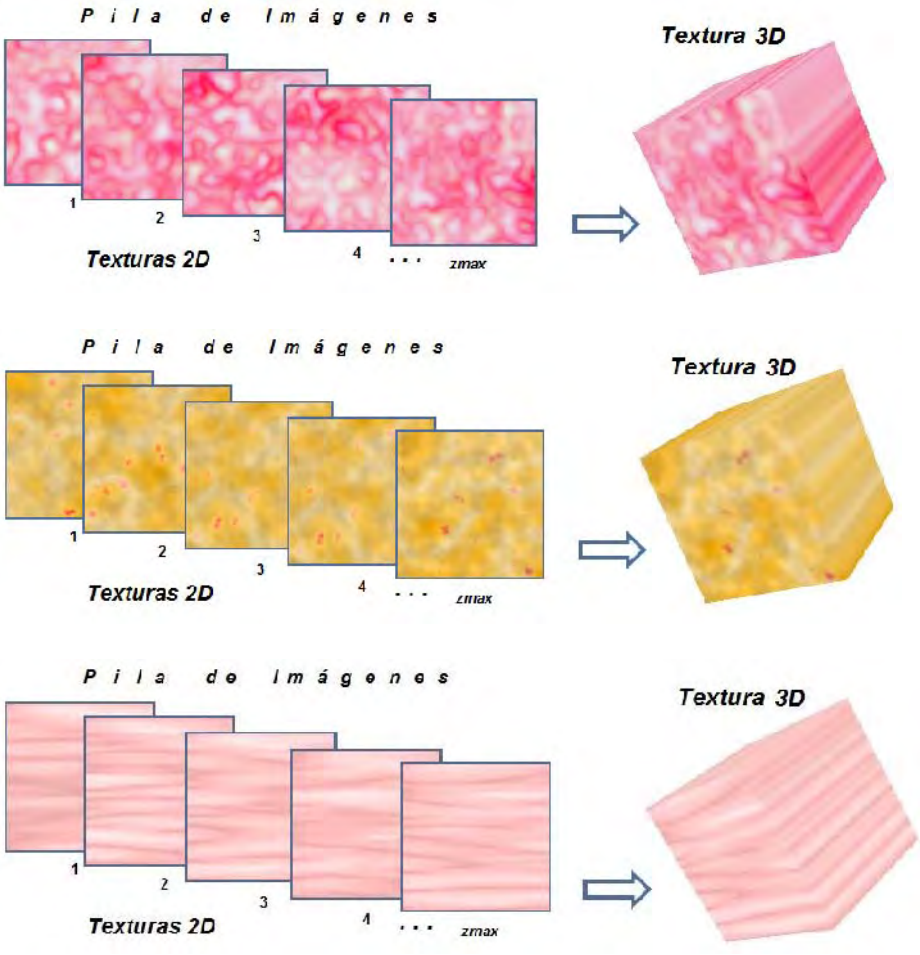
3.4.1 Procesos para la generación de la textura volumétrica

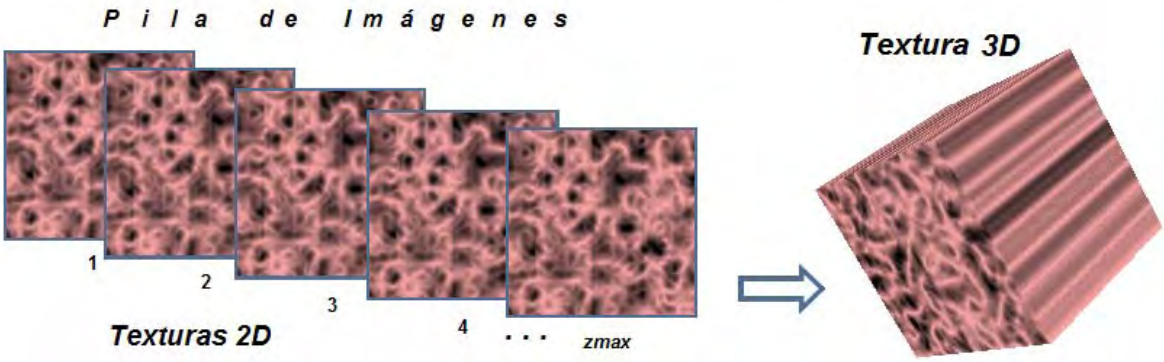
La generación de la textura 3D de la próstata se puede explicar principalmente con los siguientes pasos a realizar:

- Las funciones básicas 2D de *turbulencia, turbulencia modificada, movimiento Browniano fraccional (fBm)* y *terreno heterogéneo*, así como los valores de cada parámetro de estas, se seguirán utilizando de la misma forma que se utilizaron en la generación de texturas 2D. La diferencia radicará en la extensión de la coordenada "z" para la generación de la textura sólida.

- Al agregar el parámetro z en los algoritmos, se lograra obtener una profundidad en la textura que conceptualmente nos formara un cubo (x,y,z) del material. En el cual podremos "esculpir" los contornos de la próstata, obteniendo con ello el modelo 3D de la glándula prostática.

- Para la integración de las tres texturas generadas del tejido de la próstata, se utilizara una pila de imágenes de contornos de la glándula en blanco y negro; desarrollando un algoritmo capaz de agregar las texturas correspondientes en la región de la glándula, con lo cual se logrará el objetivo final: generar una textura volumétrica que integre las tres texturas de los tejidos prostáticos con las características y efectos del procedimiento RTU.


Pila de Imágenes
Texturas 2D
1
2
3
4
zmax
Textura 3D
Pila de Imágenes
Texturas 2D
1
2
3
4
zmax
Textura 3D
Pila de Imágenes
Texturas 2D
1
2
3
4
zmax
Textura 3D


P i l a d e I m á g e n e s
1
2
3
4
. . .
zmax
Texturas 2D
Textura 3D

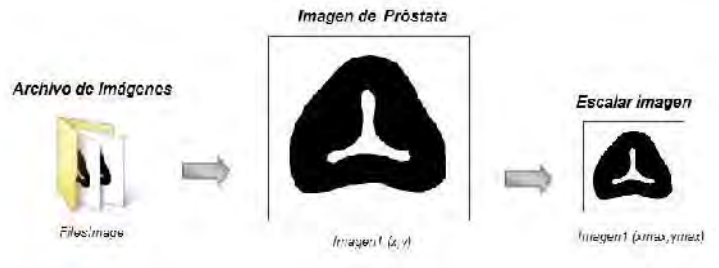
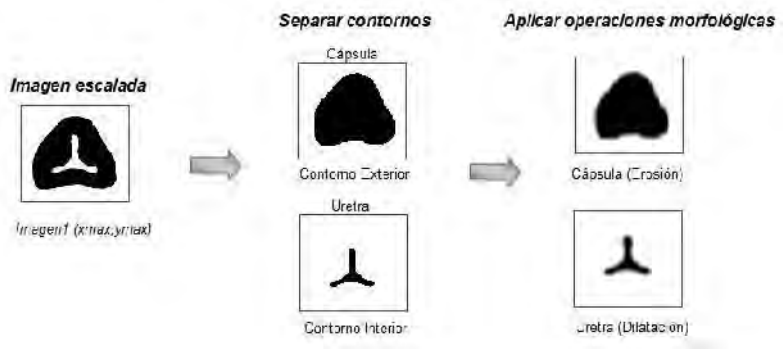


Imagen de Próstata
Archivo de Imágenes
Escalar imagen
Files:Image
Imagen1 (x,y)
Imagen1 (xmax,ymax)

14

15

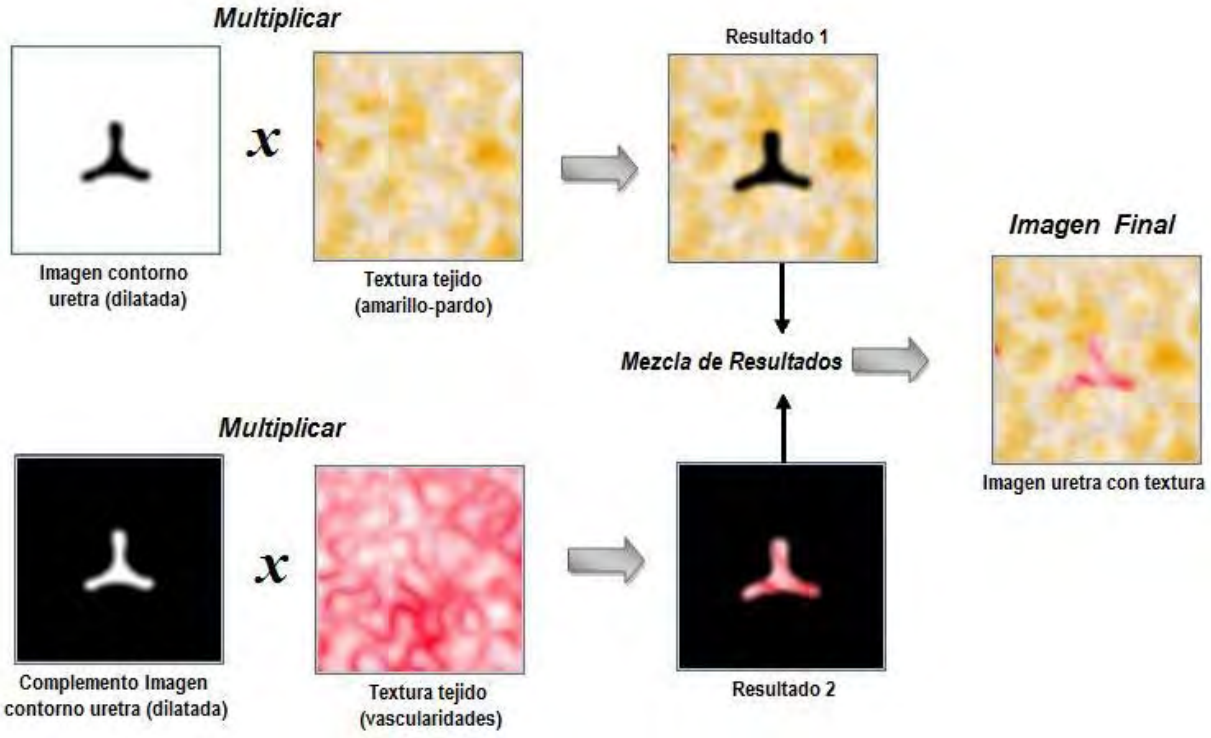

Separar contornos
Aplicar operaciones morfológicas
Imagen escalada
Cápsula
Contorno Exterior
Cápsula (Erosión)
Uretra
Contorno Interior
Uretra (Dilatación)
Imagen1 (xmax,ymax)

14

15

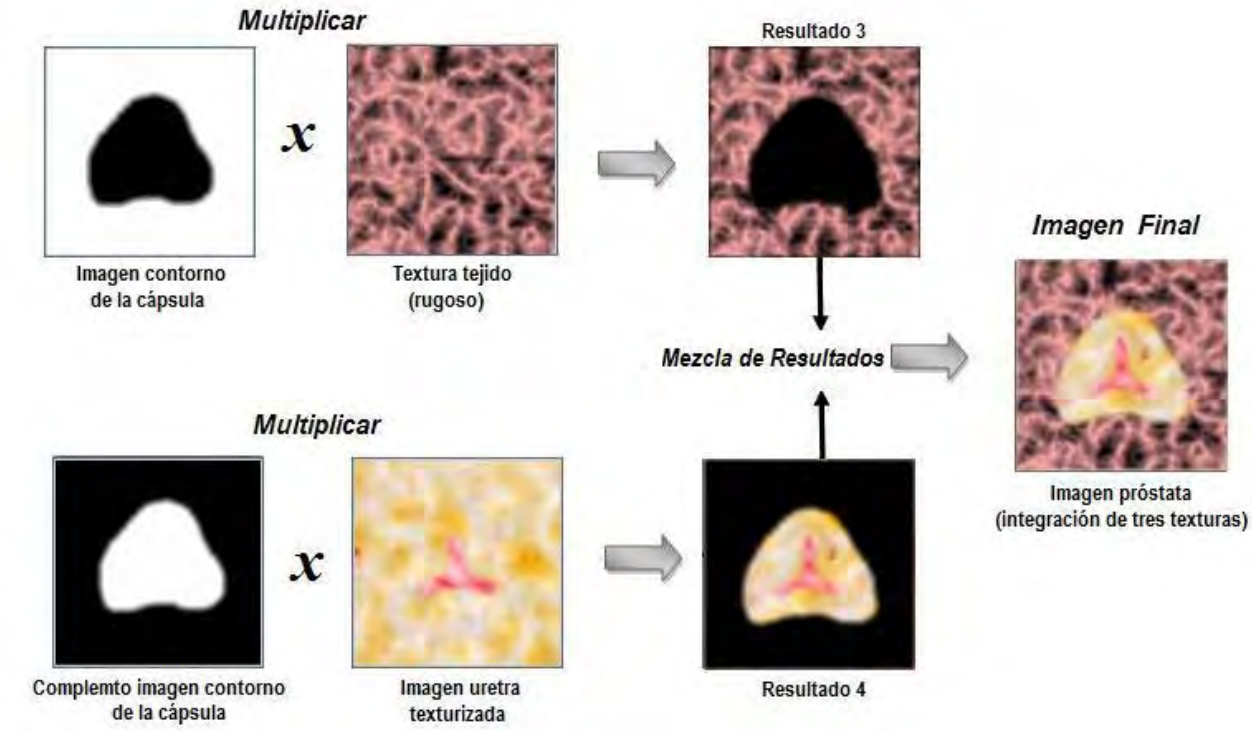
$$Texturizado1 \quad A \; x \; B$$
$$Texturizado2 \quad \overline{A} \; x \; C$$
$$Imagen\ uretra\ texturizada \quad Texturizado1 \quad Texturizado2$$

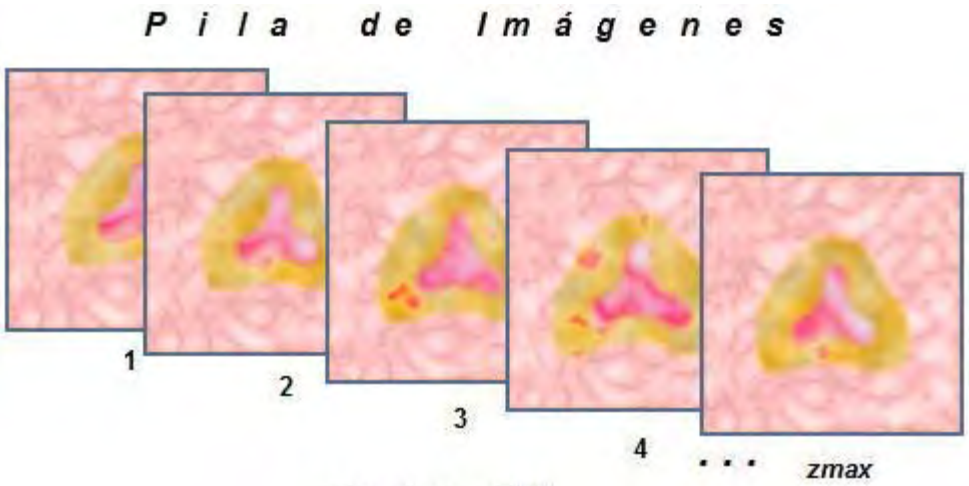
$$A \quad Imagen\ Contorno\ Uretra\ dilatada$$
$$B \quad Textura\ Interna\ Pr\ stata\ amarillo \quad pardo$$
$$C \quad Textura\ Uretra\ vascularidades$$



$$Texturizado3 \quad D \; x \; E$$
$$Texturizado4 \quad \overline{D} \; x \; F$$
$$Image\_Prostata\_Final \quad Texturizado3 \quad Texturizado4$$

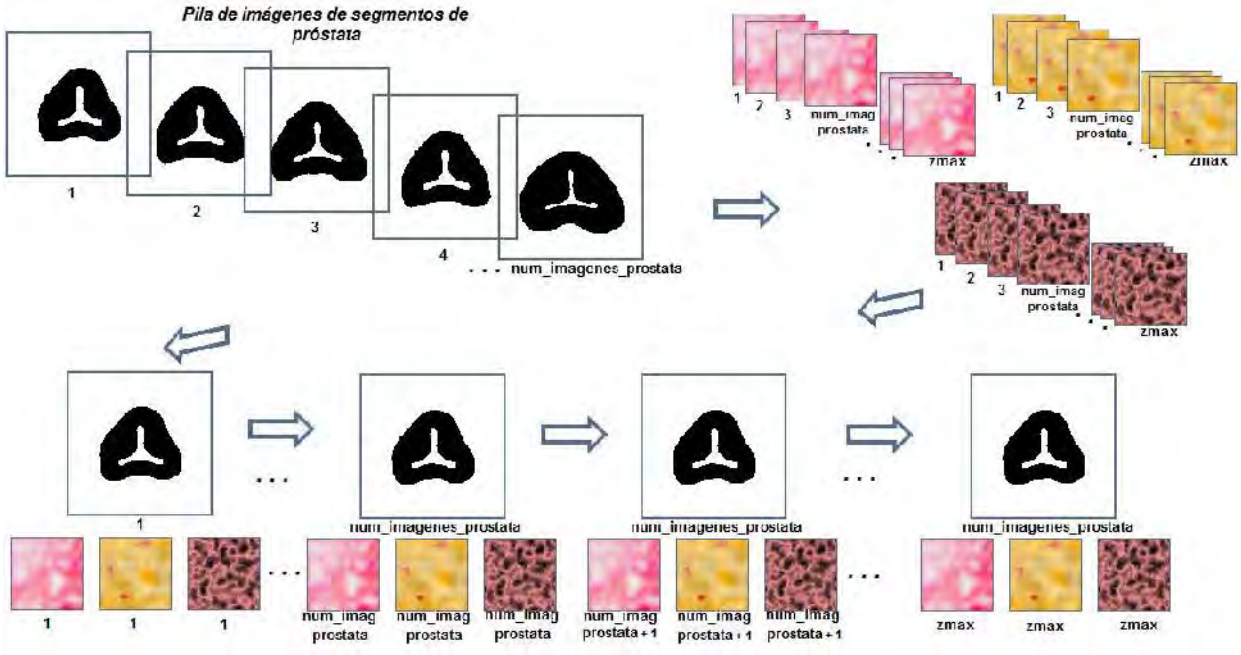
$$D \quad Imagen\_Contorno\_Pr\ stata\_erosionada$$
$$E \quad Textura\_c\ psula\_pr\ stata\_rugosa$$
$$F \quad Imagen\_uretra\_texturizada$$

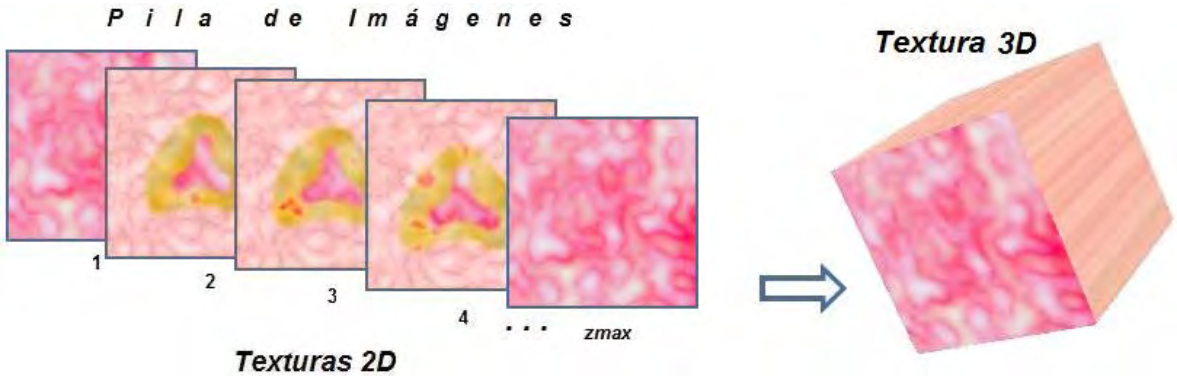




Texturas 2D

Textura 3D


Pila de imágenes de segmentos de próstata
1
2
3
4
num_imagenes_prostata
num_imag prostata
zmax
num_imagenes_prostata
num_imag prostata + 1


P i l a d e I m á g e n e s
1
2
3
4
. . .
zmax
Texturas 2D
Textura 3D

Para activar el mapeo de textura se realiza mediante la función **glEnable** eligiendo el parámetro **GL_TEXTURE_3D**.

Para la especificación de la imagen, se utiliza la función **glTexImage3D**, teniendo como parámetros: **GL_TEXTURE_3D**, **GLint** *level*, **GLint** *internalFormat*, **GLsizei** *width*, **GLsizei** *height*, **GLsizei** *depth*, **GLint** *border*, **GLenum** *format*, **GLenum** *type*, **const GLvoid** ***texels

En donde:

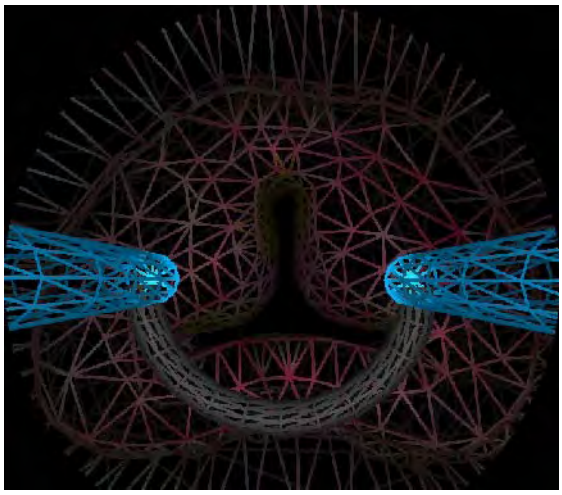
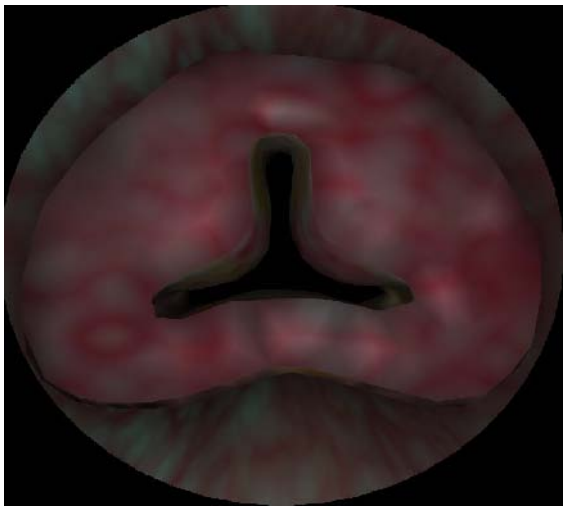
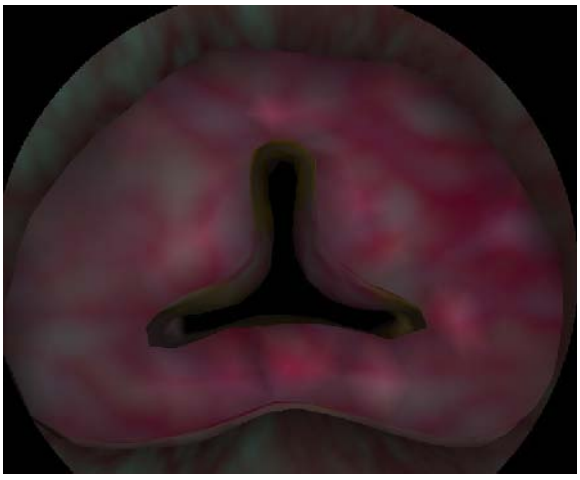
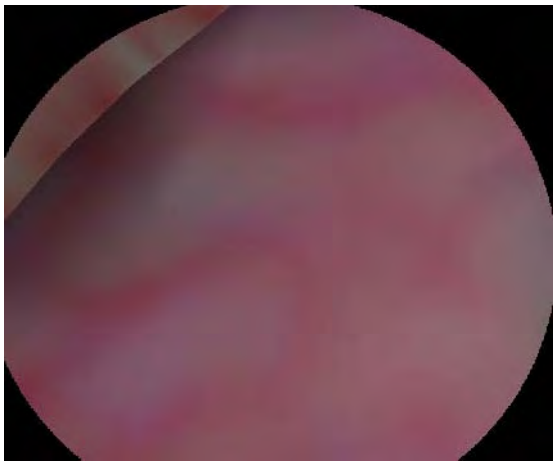
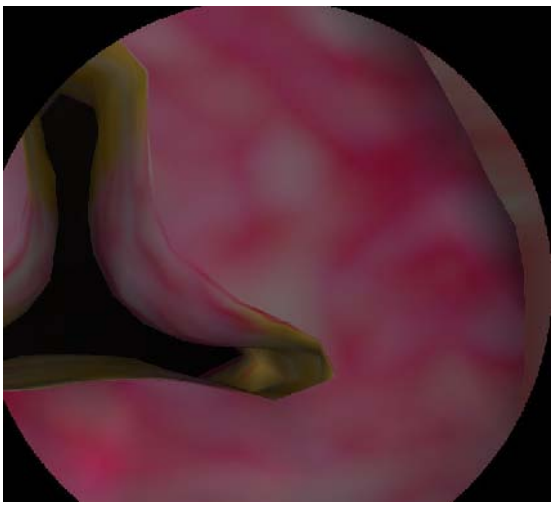
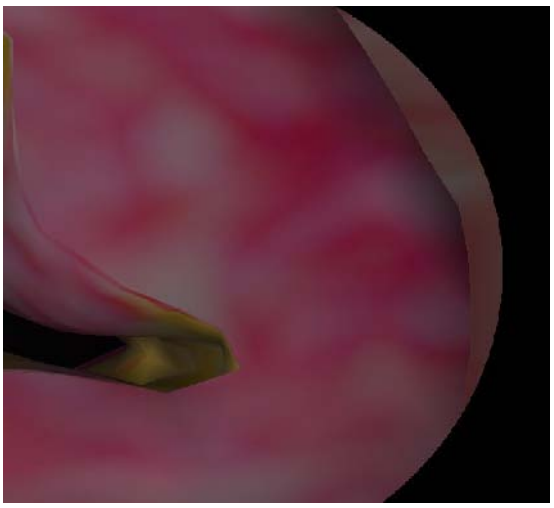
- *Level*: indica el nivel de detalle de la textura. Habitualmente tiene un valor 0.
- *InternalFormat*: indica el número de componentes de color. Color indexado=1, RGB=3, RGBA=4
- *Width*: indica el ancho de la imagen de textura, el valor del parámetro debe proporcionarse en potencia de 2.
- *Height:* indica el alto de la imagen de textura, el valor del parámetro debe proporcionarse en potencia de 2.
- *Depth:* indica la profundidad de la imagen de textura, el valor del parámetro debe proporcionarse en potencia de 2.
- *Border*: indica si se utilizará un borde en la textura con el valor 1 o 0. Usualmente es 0, sin borde.
- *Format:* formato de la información de píxel. Pueden ser: GL_COLOR_INDEX, GL_RGB, GL_RGBA, etc.
- *Type:* tipo de datos de cada píxel. Puede ser uno de los siguientes valores:
  - GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT, GL_INT o GL_FLOAT.
- *Texels*: Puntero al arreglo tridimensional en donde se almacenan los valores de la textura 3D.

Es importante mencionar que la función *glTexImage3D,* debe encontrarse antes de dibujar los polígonos sobre los que queremos aplicar la textura.

Finalmente para dibujar un objeto en la escena, se debe indicar a cada vértice de este la posición de textura que le corresponde. Esto se hará mediante la función **glTexCoord3f(GLfloat s, GLfloat t, GLfloat r).**

En donde (s,t,r) indicará la posición sobre el mapa de textura 3D. *(Shreiner, et al., 2007)*

Algunos resultados del mapeo de la textura volumétrica generada, sobre el simulador los podemos visualizar a continuación en la figura 3.32.

CAPÍTULO 4

4. PRUEBAS Y RESULTADOS OBTENIDOS EN LA GENERACIÓN DE TEXTURAS PROCEDURALES

En este capítulo se explicaran las pruebas y resultados de la generación de las texturas procedurales utilizando el software Matlab para el desarrollo del código.

Las texturas se generaron con las funciones base: *ruido de Perlin, función movimiento Browniano fraccional (fBm), turbulencia, turbulencia modificada y terreno heterogéneo*, como se explicó en el capítulo anterior.

Posteriormente se mostraran las tablas con las pruebas y resultados de cada función variando cada uno de los parámetros de las mismas para cada textura generada, logrando visualizar los cambios que sufre la imagen final obtenida.

Finalmente se mostraran algunas imágenes del simulador virtual de cirugía con la texturización de la síntesis de texturas generada en este trabajo.

4.1 Matlab: generación de texturas

La implementación de los algoritmos generadores de texturas procedurales bien pueden ser programados en lenguajes de alto nivel, o bien en tiempo real en una tarjeta aceleradora de gráficos. En este trabajo se busco minimizar tiempo en el periodo de pruebas y correcciones de las funciones básicas generadoras de textura, así como pre-calcular fuera de línea las texturas para evitar realizar cálculos innecesarios en el simulador en tiempo real. Por tal motivo se ha resuelto utilizar la herramienta de software Matlab que con ayuda de su toolbox[16] de procesamiento de imágenes acortará el tiempo de implementación haciendo más sencilla la realización de las pruebas necesarias.

La caja de herramientas de procesamiento de imágenes contiene un conjunto de las funciones más conocidas para trabajar con imágenes binarias, transformaciones geométricas, morfología y manipulación del color. Adecuando perfectamente la herramienta a las condiciones que necesitamos para modelar nuestras texturas.

4.2 Resultados del modelado de texturas procedurales al variar sus parámetros

Se realizaron las pruebas correspondientes a cada una de las funciones básicas presentes en la generación de cada una de las texturas. Quedando en el siguiente orden:

Textura del tejido de la uretra con vascularidades

- Función Terreno Heterogéneo(h, lacunaridad, offset, octavas)
- Función Turbulencia (escala, octava_inicial, octavas, factor)
- Función fBm (escala, h, lacunaridad, octavas)

Textura del tejido interno de la próstata con apariencia amarillo - pardo

- Función Turbulencia (escala, octava_inicial, octavas)
- Función fBm1 (escala, h, lacunaridad, octavas)
- Función fBm2 (escala, h, lacunaridad, octavas)

Textura del tejido de la cápsula de la próstata con apariencia rugosa

- Función Turbulencia modificada (escala, octava_inicial, octavas)

---

[16] **Toolbox:** MATLAB dispone de un amplio abanico de programas de apoyo especializados, denominados Toolboxes, que extienden significativamente el número de funciones incorporadas en el programa principal. Estos Toolboxes cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos el 'toolbox' de proceso de imágenes, señal, control, estadística, análisis financiero, matemáticas simbólicas, redes neurales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, etc.

En las siguientes secciones, se ejemplificarán los resultados más significativos que se obtuvieron al modificar los valores de los parámetros de cada función procedural básica, en cada textura generada. Esto nos fue realmente útil para poder conocer las variaciones que pueden existir en la generación de dichas texturas, pudiendo con ello conseguir una base de imágenes de texturas sintéticas, aplicables al modelo de cirugía virtual y ser capaz de presentar varias "texturizaciones", para lograr tener varios efectos sobre el simulador.

4.2.1 Textura del tejido de la uretra con vascularidades

Como se explico en el capítulo anterior, la generación de esta textura se hizo bajo la utilización de las tres funciones básicas: *terreno heterogéneo, turbulencia y movimiento Browniano fraccional (fBm).* A fin de conocer las variaciones que causa la modificación de los parámetros de entrada de cada función, se sintetizaron algunos de los resultados obtenidos, los cuales se mostrarán en las secciones siguientes.

4.2.1.1 Función terreno heterogéneo

Es importante recordar la utilización de esta función, en la generación de la textura de vascularidades, ya que ayudo a definir una zona dentro de la textura con una apariencia más "inflamada", simulando así una zona de un color más intenso, en la cual en un futuro servirá para disparar el efecto de la simulación de sangrado, cuando se toque dicha zona. Los parámetros de esta función procedural son:

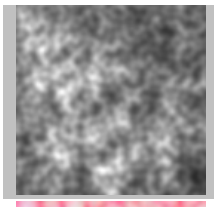
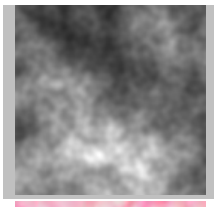
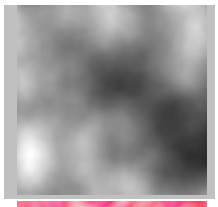
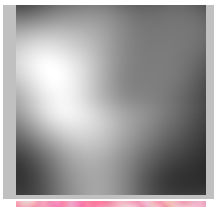
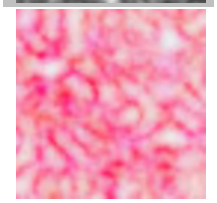
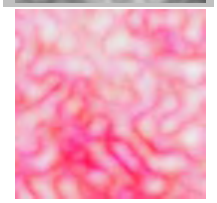
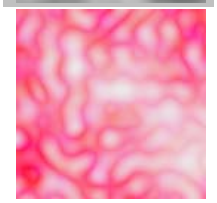
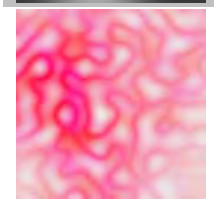
Funcion Terreno_heterogeneo (***h, lacunaridad, offset, num_octavas****)*

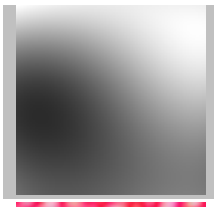
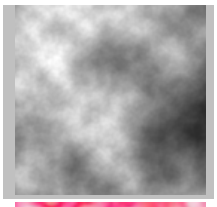
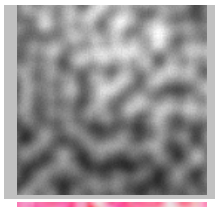
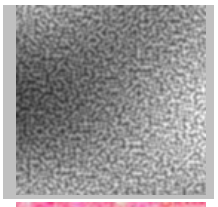
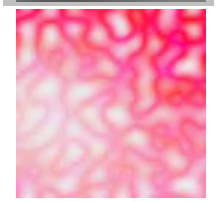
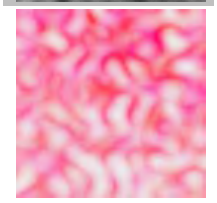
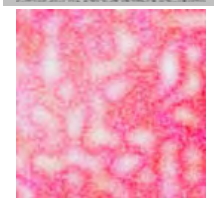
Los resultados que se presentan a continuación, se lograron al ir modificando el valor de cada uno de estos parámetros.

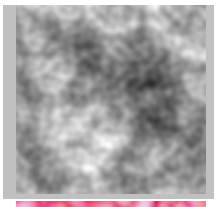
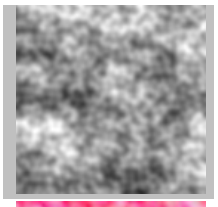
Parámetro H

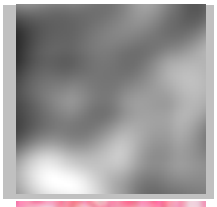
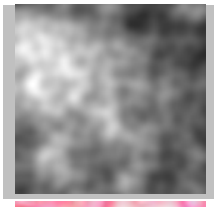
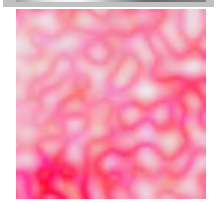
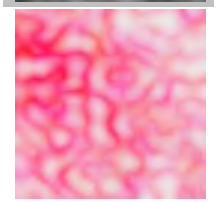
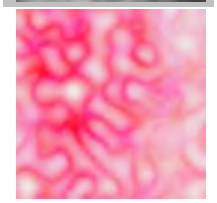
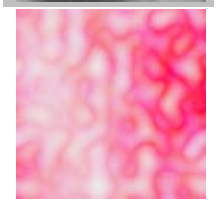
El parámetro H nos proporciona una "rugosidad" en la textura final generada. Cuando se cuenta con un valor pequeño en dicho parámetro, se obtendrá una textura con una apariencia demasiado "rugosa", por el contrario al tener un valor muy grande, se generará una textura mucho más "lisa".

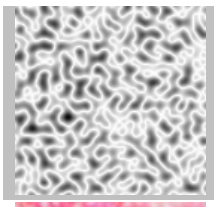
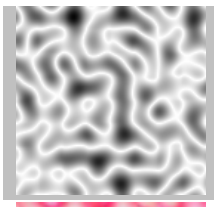
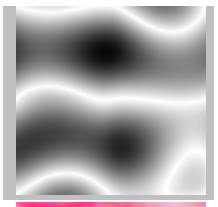
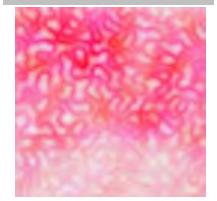
En la tabla 1, se muestran los resultados al variar el valor de *H* comenzando con un *valor en H = 0* hasta un valor en *H= 4.5*. Se puede apreciar primeramente el resultado de la función terreno heterogéneo en escala de grises al variar el parámetro H. Posteriormente se encuentra el resultado de la textura sintética final generada, apreciando las diferencias y los distintos efectos que podemos lograr al manipular este parámetro.

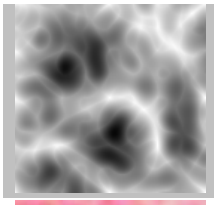
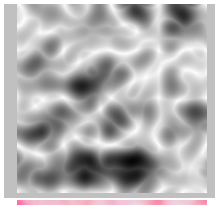
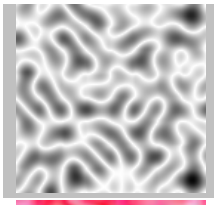
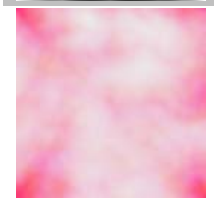
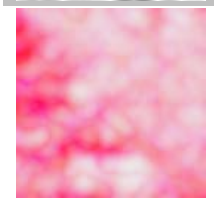
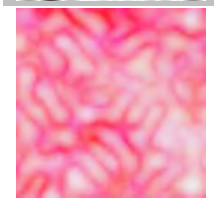
H	0	0.8	2.2	4.5
Textura función terreno heterogéneo				
Textura de tejido final generada (vascularidades)				

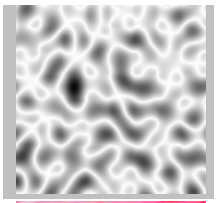
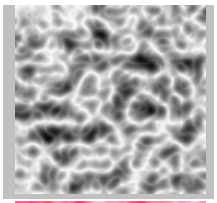
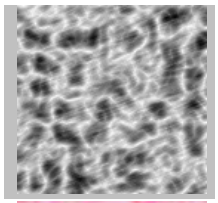
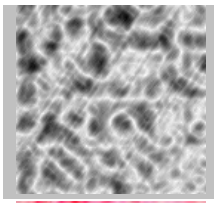
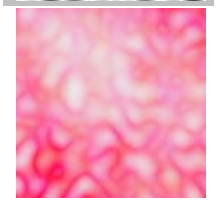
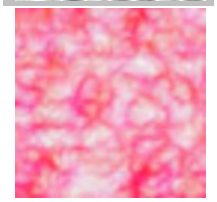
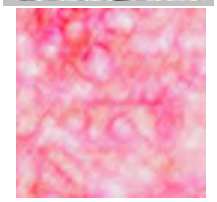
Lacunaridad	0.4	2.2	10	50
Textura función terreno heterogéneo				
Textura de tejido final generada (vascularidades)				

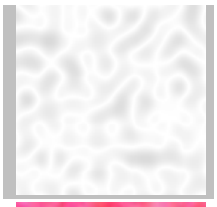
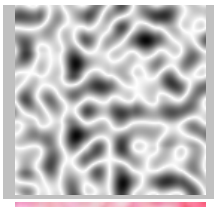
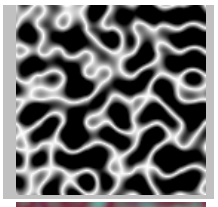
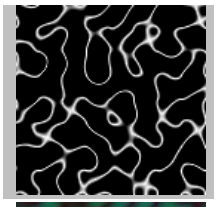
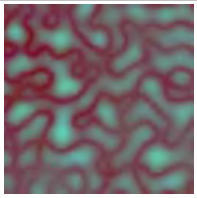
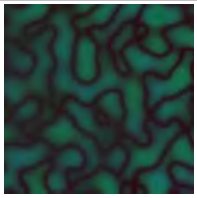
Offset	0.8	2.2	10	50
Textura función terreno heterogéneo				
Textura de tejido final generada (vascularidades)				

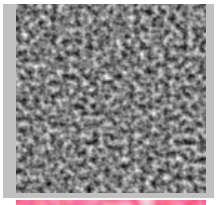
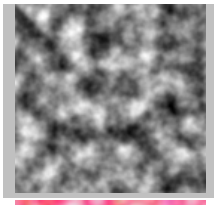
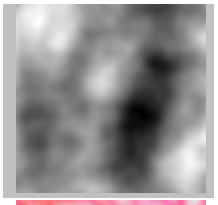
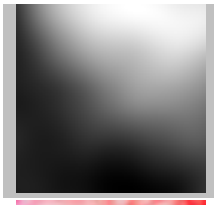
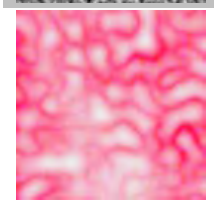
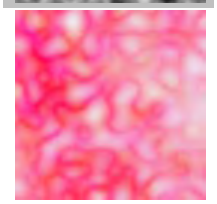
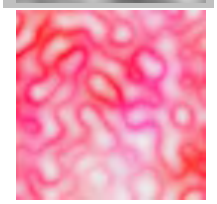
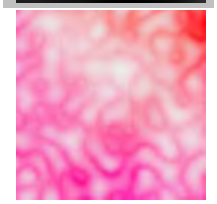
Octavas	1.5	3.67	10.25	12.14
Textura función terreno heterogéneo				
Textura de tejido final generada (vascularidades)				

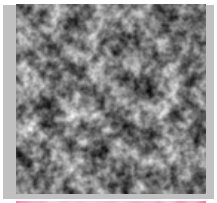
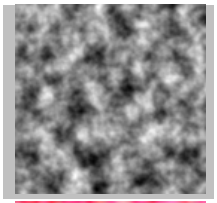
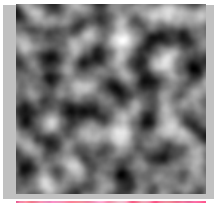
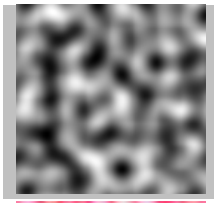
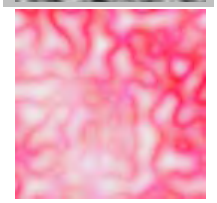
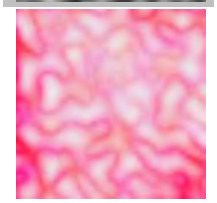
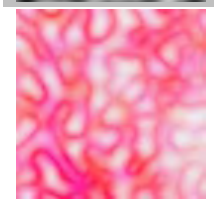
Escala	16	64	128	512
Textura función turbulencia				
Textura de tejido final generada (vascularidades)				

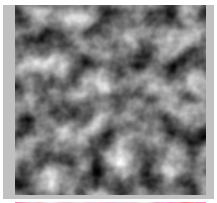
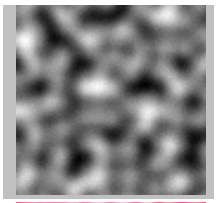
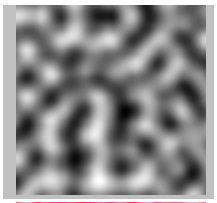
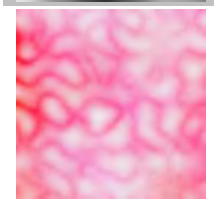
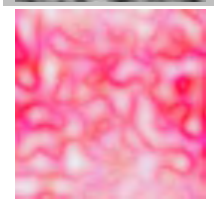
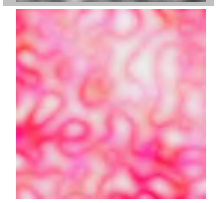
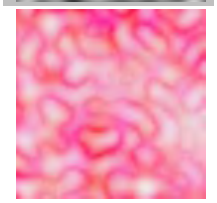
Octava inicial	1	2	3	4
Textura función turbulencia				
Textura de tejido final generada (vascularidades)				

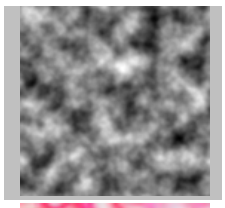
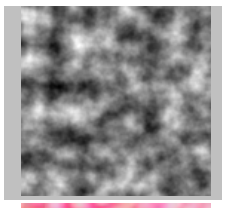
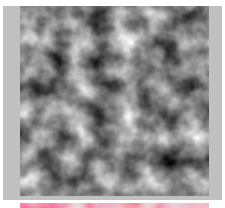
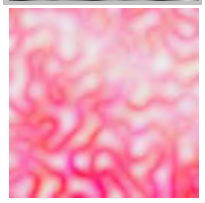
Octavas	4.2	5.42	8.76	12.45
Textura función turbulencia				
Textura de tejido final generada (vascularidades)				

Factor	0.5	3.2	10	30
Textura función turbulencia				
Textura de tejido final generada (vascularidades)				

Escala	4	16	64	256
Textura función fBm				
Textura de tejido final generada (vascularidades)				

H	0.2	0.8	2.2	10
Textura función fBm				
Textura de tejido final generada (vascularidades)				

Lacunaridad	0.4	2.2	10	50
Textura función fBm				
Textura de tejido final generada (vascularidades)				

Octavas	1.5	3.43	5.78	10.45
Textura función fBm				
Textura de tejido final generada (vascularidades)				

4.2.2.1 Función turbulencia

Esta función fue muy útil por contar con las características para ser la textura "base" del tejido interno de la próstata, ya que es la que cuenta con las condiciones visuales de una apariencia esponjosa parecida a la imagen médica real.

Es importante mencionar que a pesar de ser la función de turbulencia que se utilizó para agregar las vascularidades en la generación de la textura anterior; en este caso se utilizo por contener una apariencia visual muy parecida a la mostrada en la imagen médica real del tejido interno de la próstata.

Para lograr estas diferencias en el resultado de la generación de dichas texturas procedurales, solo basto con modificar los valores de sus parámetros de entrada, sobre todo sus parámetros de *octava inicial* y *número de octava*, logrando con esto obtener una textura procedural totalmente diferente. De hecho esto suele ser una de las ventajas más significativas de las funciones procedurales, ya que con solo manipular algunos de los parámetros de sus entradas podremos crear características y diferentes efectos aplicables a otros modelos virtuales de una misma función procedural básica.

Pues bien, los parámetros de esta función son:

*Funcion turbulencia (**escala, octava_inicial, num_octavas**)*

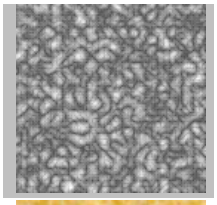
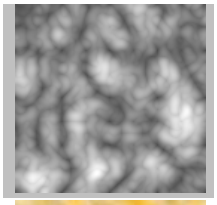
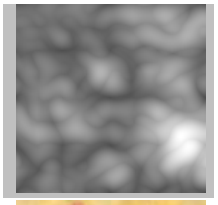
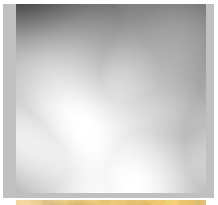
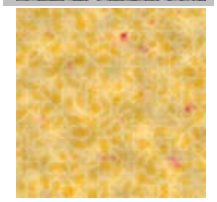
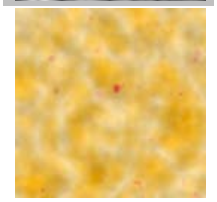
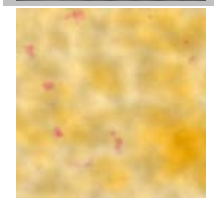
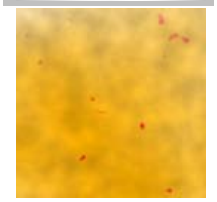
A continuación se desarrollaran los comentarios y los resultados obtenidos al variar el valor de cada parámetro de esta función.

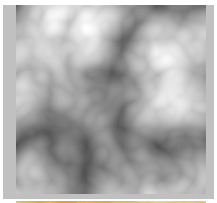
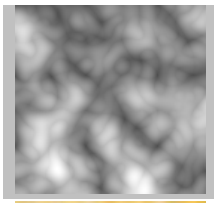
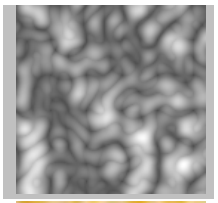
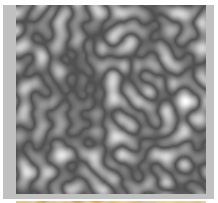
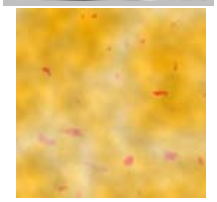
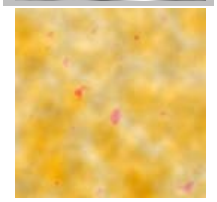
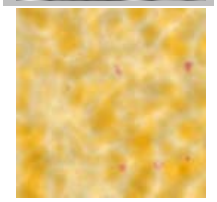
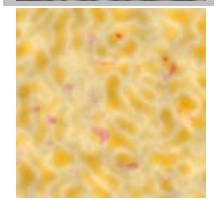
Parámetro ESCALA

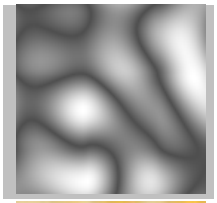
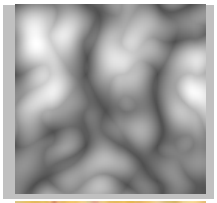
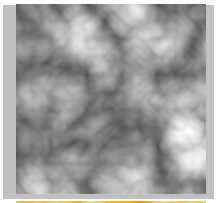
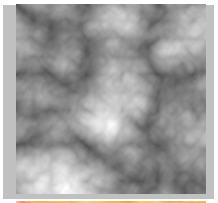
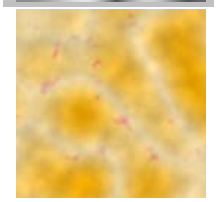
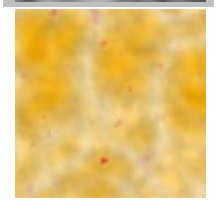
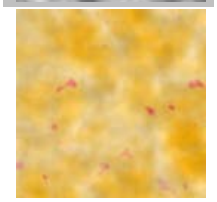
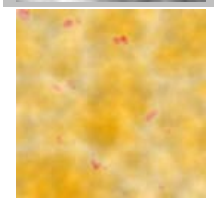
Este parámetro causa el efecto de zoom sobre la textura, ya que pareciera que se realiza un alejamiento o acercamiento sobre la misma. Como bien se menciono anteriormente, esta función nos da la característica "base" de la textura final generada.

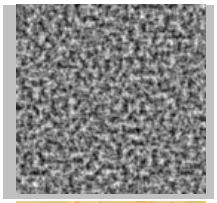
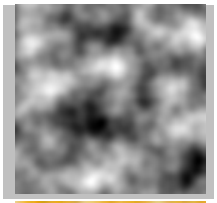
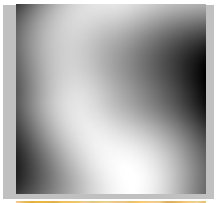
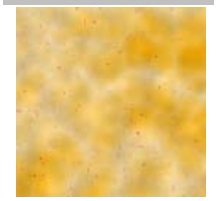
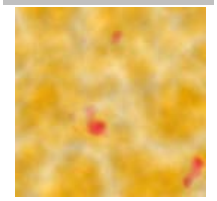
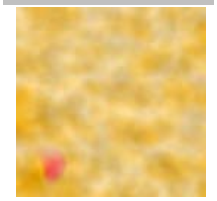
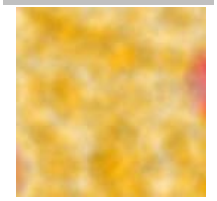
En la tabla 13, podemos apreciar los resultados que se obtienen al variar el valor de este parámetro. En el segundo renglón de la tabla 13 se pueden observar los resultados de la función básica de turbulencia en escala de grises, apreciando el efecto *zoom* con el incremento del valor.

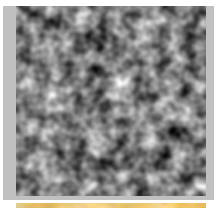
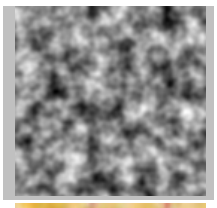
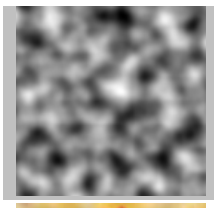
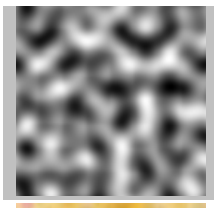
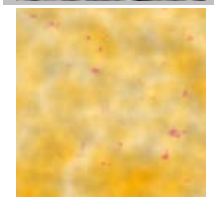
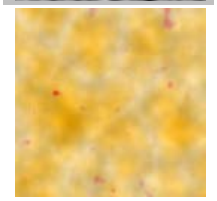
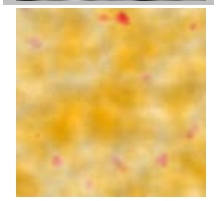
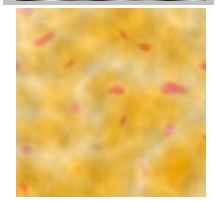
En el tercer renglón de la tabla 13, podemos apreciar el resultado de la textura final obtenida conservando la forma de la función procedural básica, añadiendo a está solo las tonalidades en blanco, amarillo y rojo.

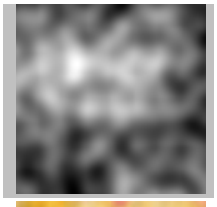
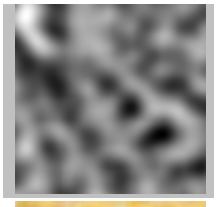
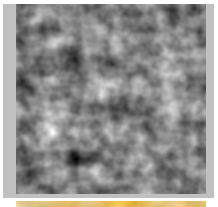
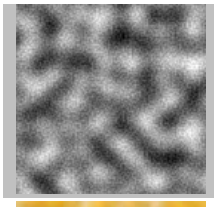
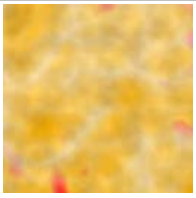
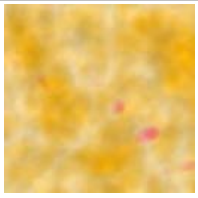
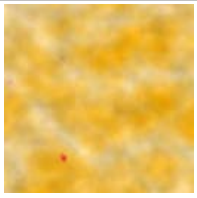
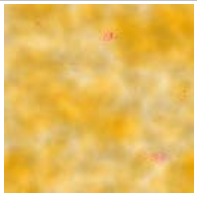
Escala	16	64	128	512
Textura función turbulencia				
Textura de tejido final generada (apariencia amarillo-pardo)				

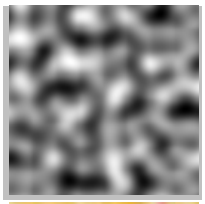
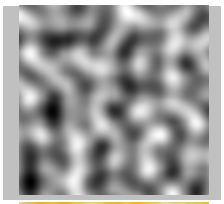
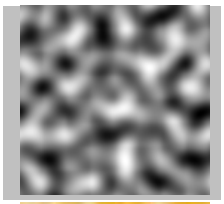
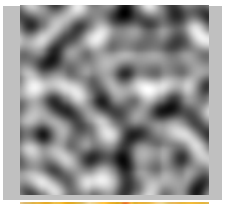
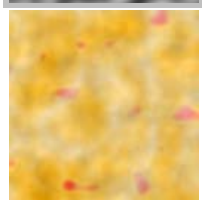
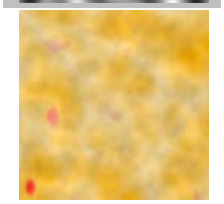
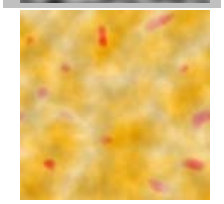
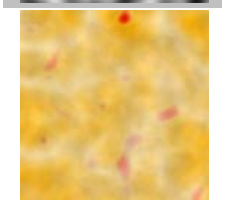
Octava inicial	1	2	3	4
Textura función turbulencia				
Textura de tejido final generada (apariencia amarillo-pardo)				

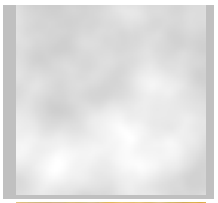
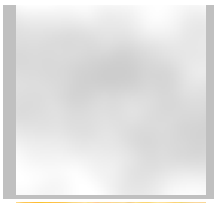
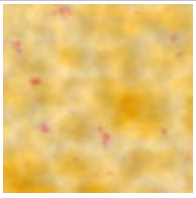
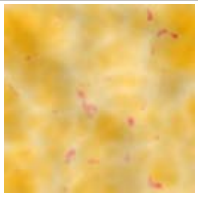
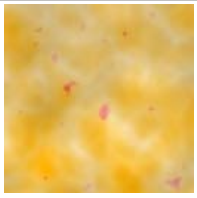
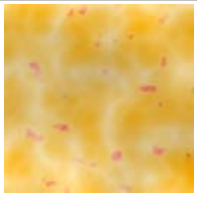
Octavas	2.38	3.43	5.63	10.54
Textura función turbulencia				
Textura de tejido final generada (apariencia amarillo-pardo)				

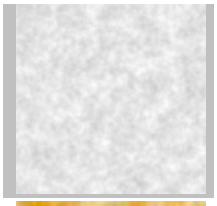
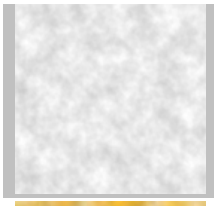
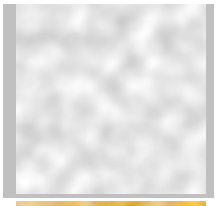
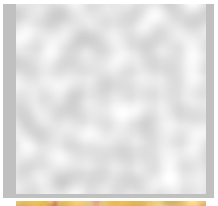
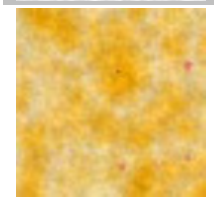
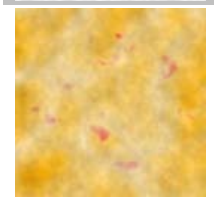
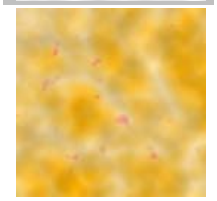
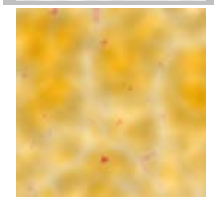
Escala	4	32	64	256
Textura función fBm1				
Textura de tejido final generada (apariencia amarillo-pardo)				

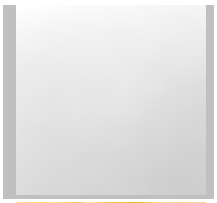
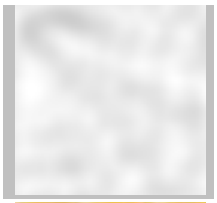
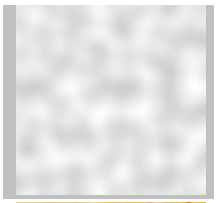
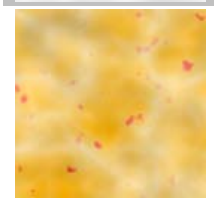
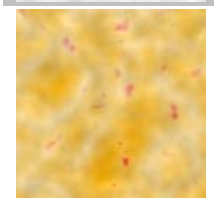
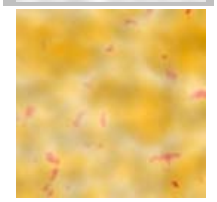
H	0	0.4	2.2	10
Textura función fBm1				
Textura de tejido final generada (apariencia amarillo-pardo)				

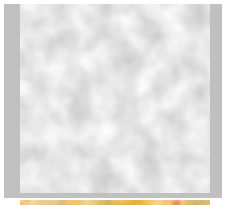
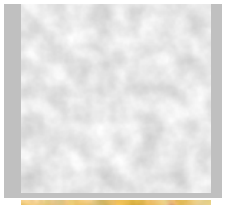
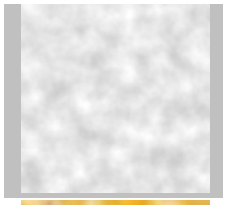
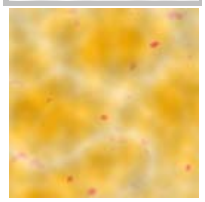
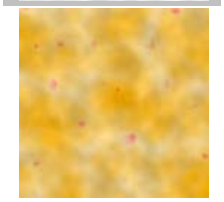
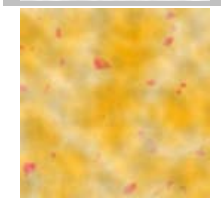
Lacunaridad	0.2	0.8	2.2	10
Textura función fBm1				
Textura de tejido final generada (apariencia amarillo-pardo)				

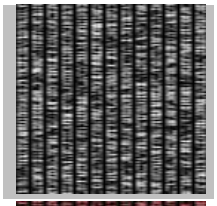
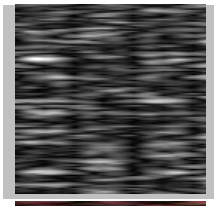
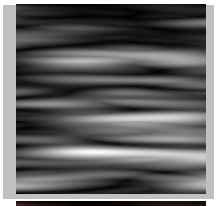
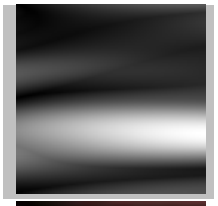
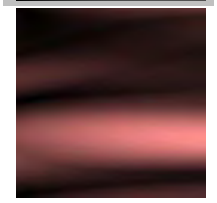
Octavas	1.5	2.84	5.63	10.82
Textura función fBm1				
Textura de tejido final generada (apariencia amarillo-pardo)				

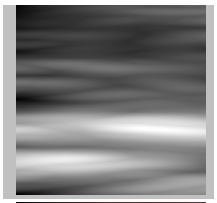
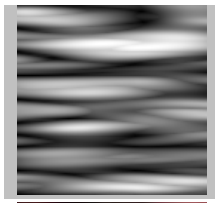
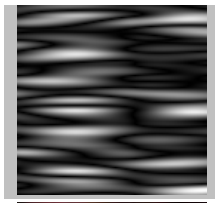
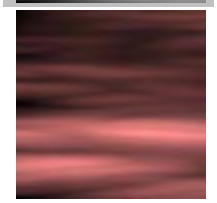
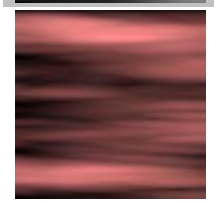
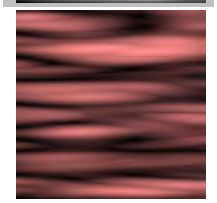
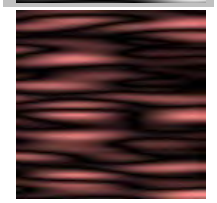
Escala	16	32	64	128
Textura función fBm2				
Textura de tejido final generada (apariencia amarillo-pardo)				

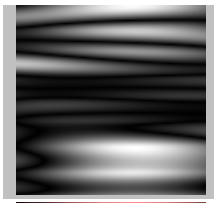
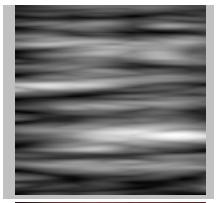
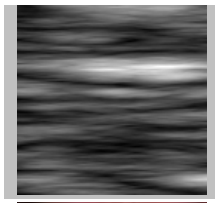
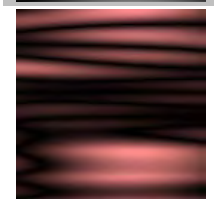
H	0	0.4	2.2	10
Textura función fBm2				
Textura de tejido final generada (apariencia amarillo-pardo)				

Lacunaridad	0.2	0.8	2.2	10
Textura función fBm2				
Textura de tejido final generada (apariencia amarillo-pardo)				

Octavas	1.5	2.84	5.63	10.82
Textura función fBm2				
Textura de tejido final generada (apariencia amarillo-pardo)				

Escala	4	16	64	256
Textura función turbulencia				
Textura de tejido final generada (apariencia rugosa)				

Octava inicial	1	2	3	4
Textura función turbulencia				
Textura de tejido final generada (apariencia rugosa)				

Octavas	3.75	5.75	8.43	12.55
Textura función turbulencia				
Textura de tejido final generada (apariencia rugosa)				

32 x 32 pixeles

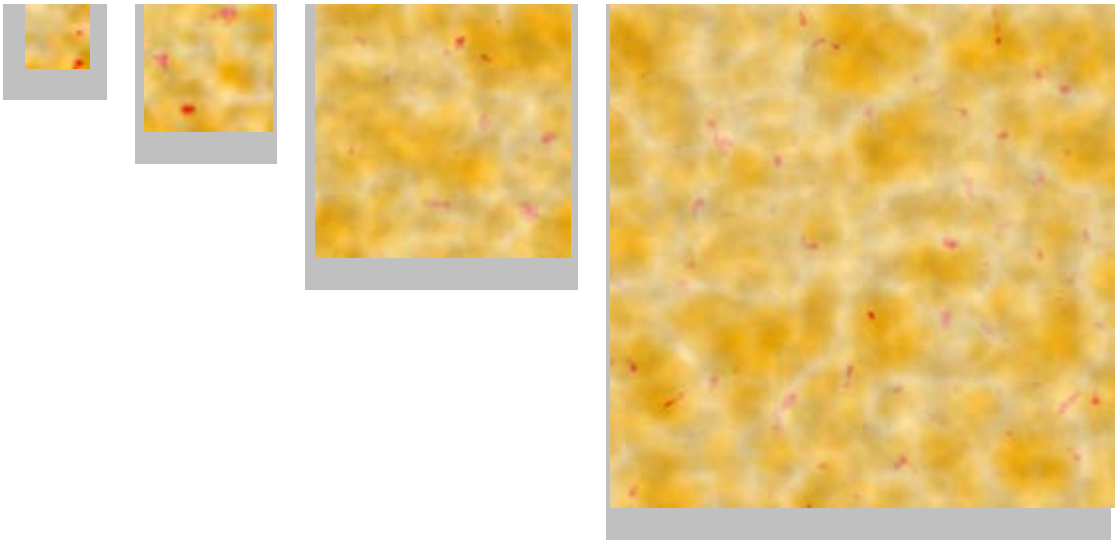
64 x 64 pixeles

128 x 128 pixeles

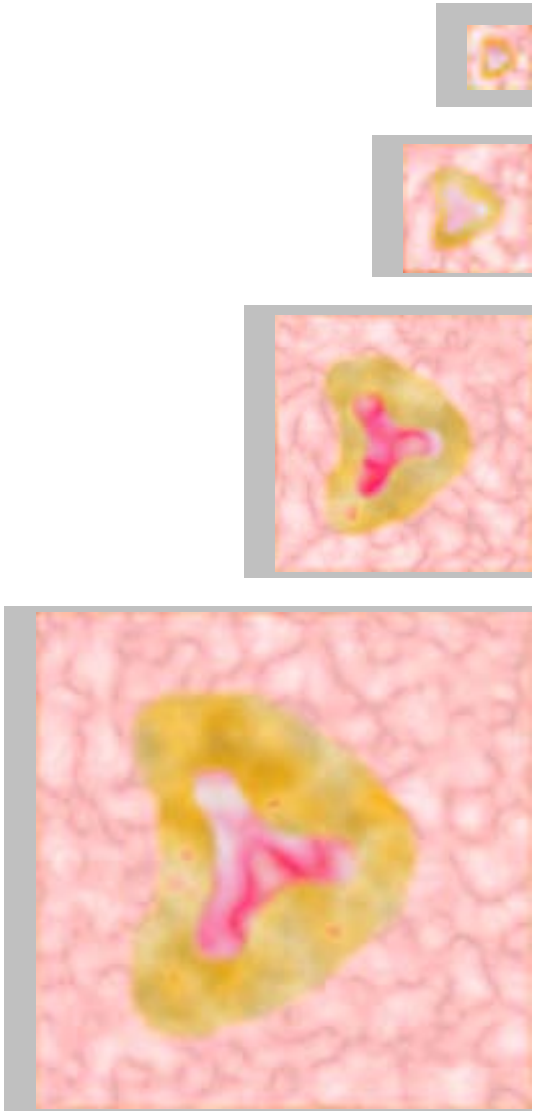
256 x 256 pixeles

Textura tejido (vascularidades)

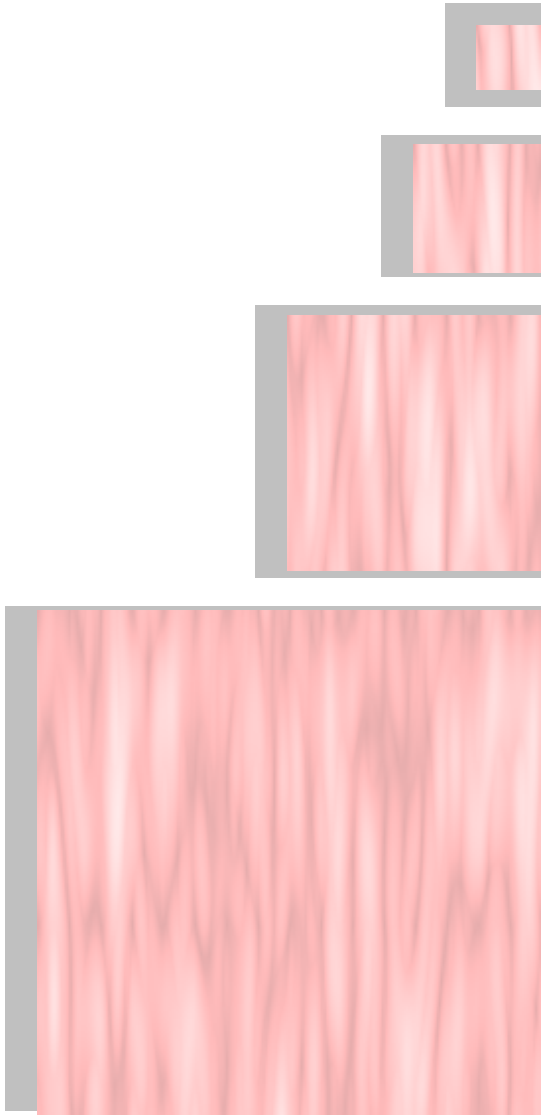
Textura tejido (apariencia amarilla)

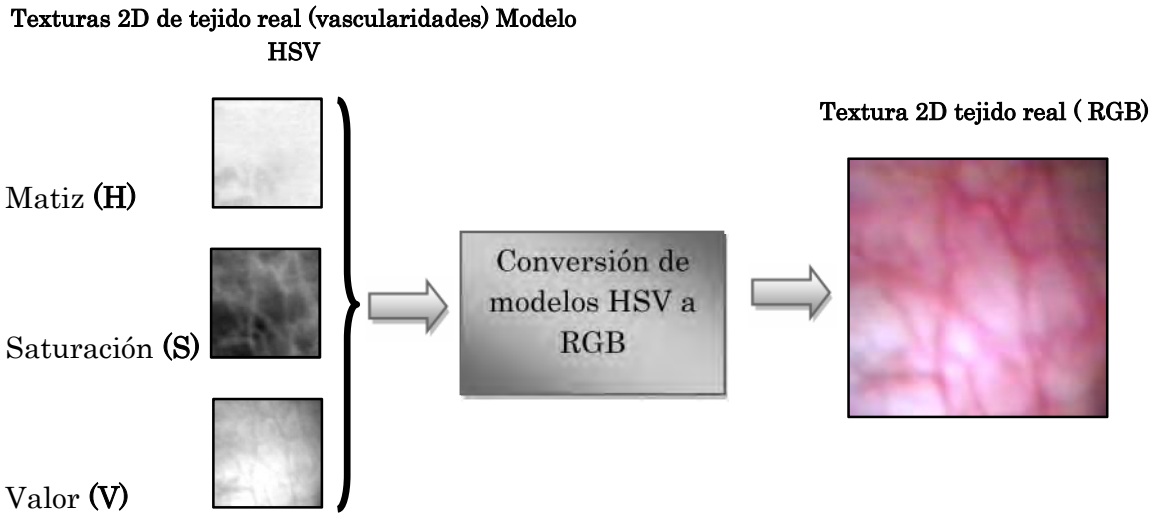


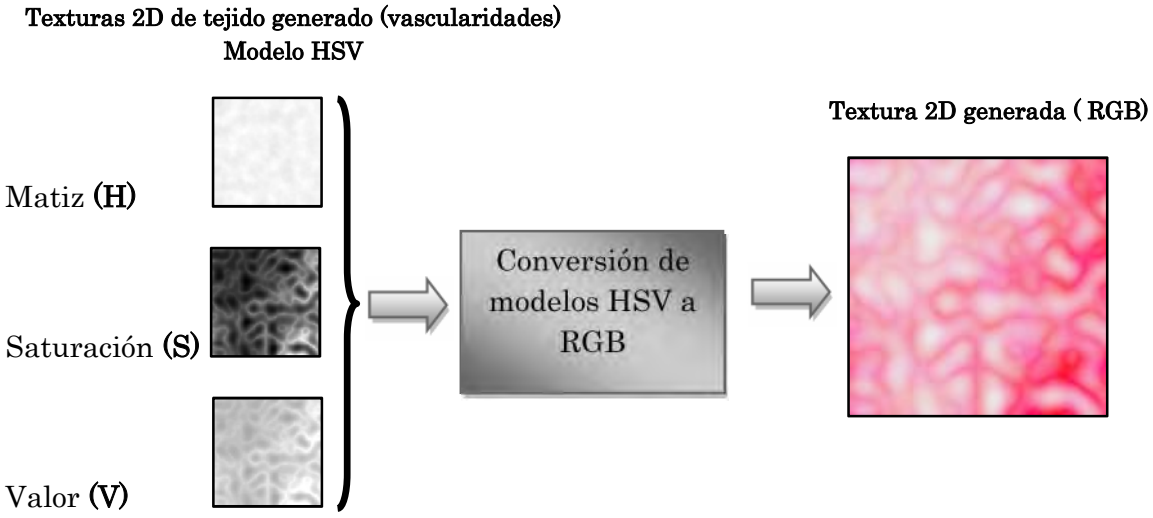
Textura próstata

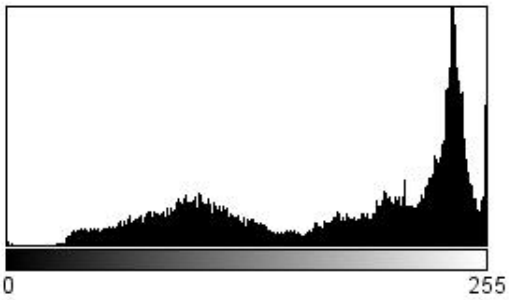


Textura tejido (apariencia rugosa)

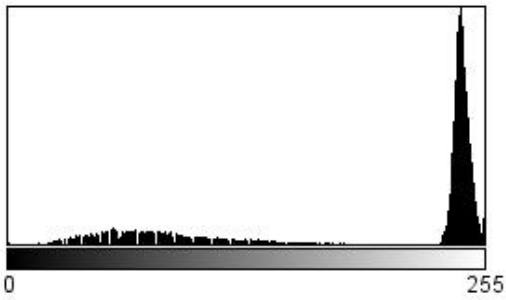



Texturas 2D de tejido real (vascularidades) Modelo HSV
Matiz (H)
Saturación (S)
Valor (V)
Conversión de modelos HSV a RGB
Textura 2D tejido real (RGB)

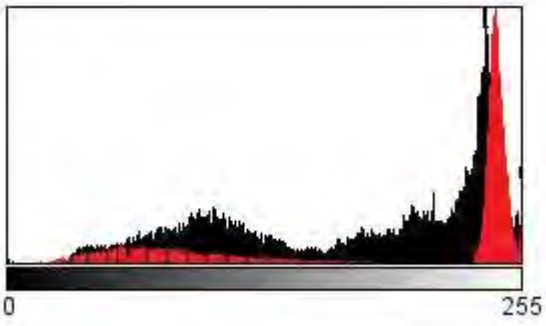

Texturas 2D de tejido generado (vascularidades) Modelo HSV
Matiz (H)
Saturación (S)
Valor (V)
Conversión de modelos HSV a RGB
Textura 2D generada (RGB)



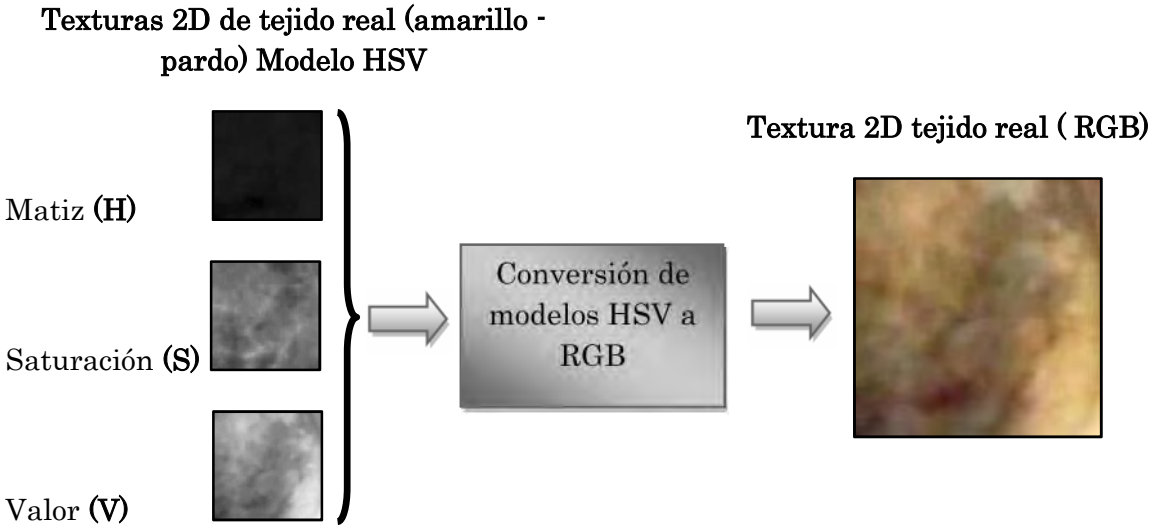
a) Histograma del modelo de color HSV correspondiente a la textura real del tejido de vascularidades.

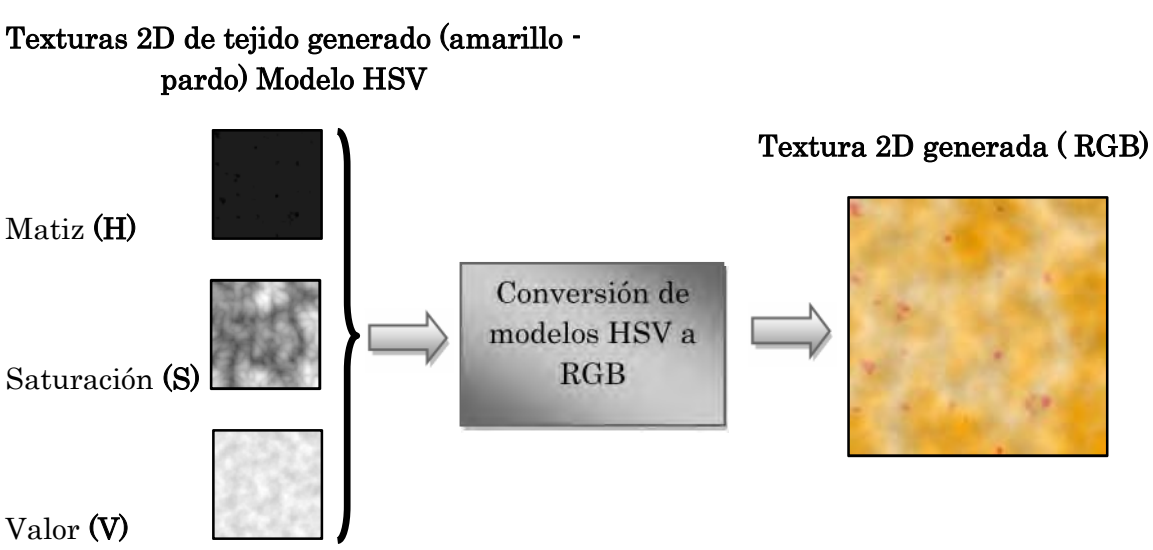


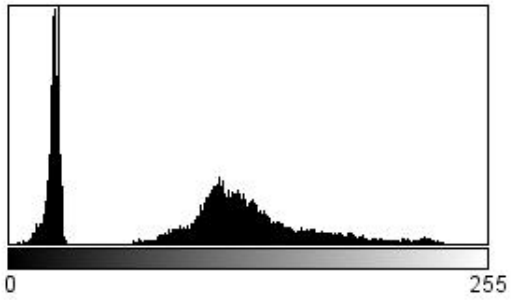
b) Histograma del modelo de color HSV correspondiente a la textura generada con apariencia de vascularidades.



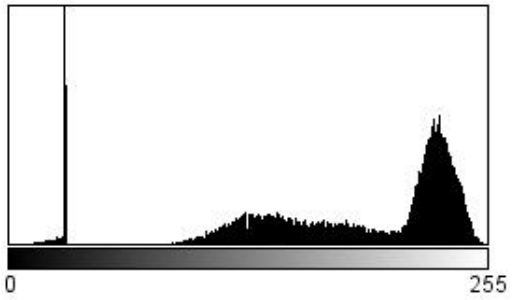
c) Resultado de la comparación de histogramas. En color negro: resultado del tejido real. En color rojo: resultado de la textura generada por computadora.


Texturas 2D de tejido real (amarillo - pardo) Modelo HSV
Matiz (H)
Saturación (S)
Valor (V)
Conversión de modelos HSV a RGB
Textura 2D tejido real (RGB)

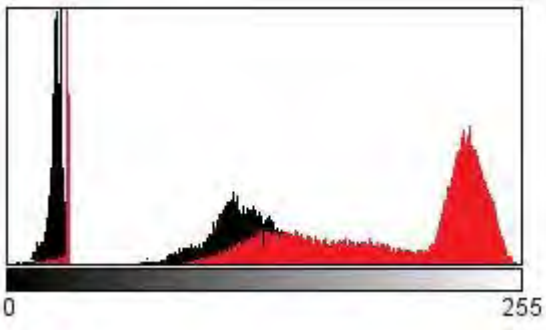

Texturas 2D de tejido generado (amarillo - pardo) Modelo HSV
Matiz (H)
Saturación (S)
Valor (V)
Conversión de modelos HSV a RGB
Textura 2D generada (RGB)



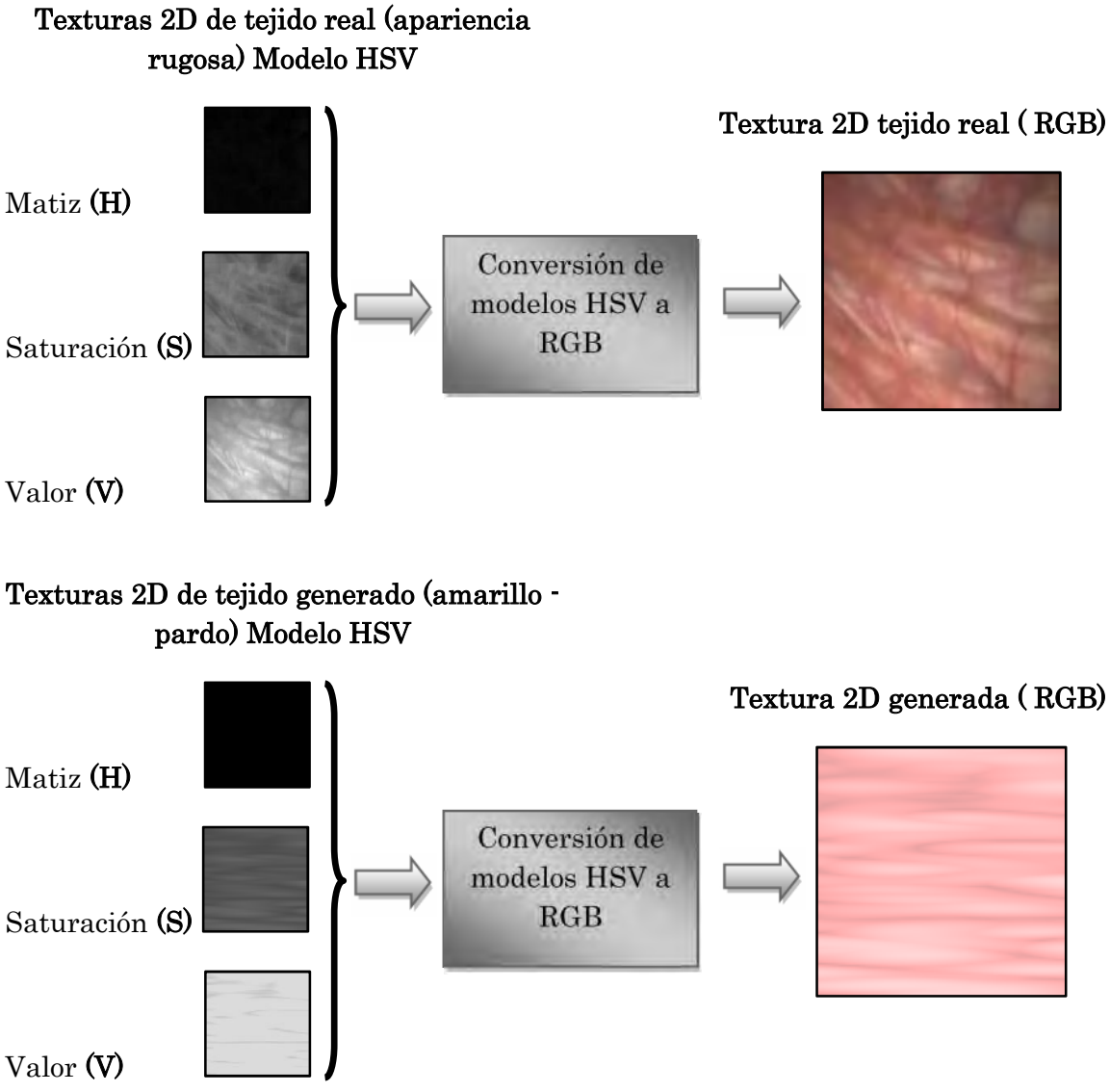
a) Histograma en modelo de color HSV correspondiente a la textura real del tejido (amarillo-pardo).

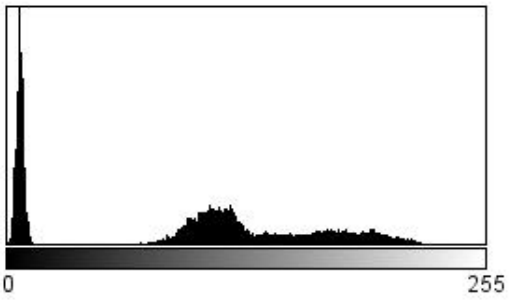


b) Histograma en modelo de color HSV correspondiente a la textura generada con apariencia (amarillo-pardo).



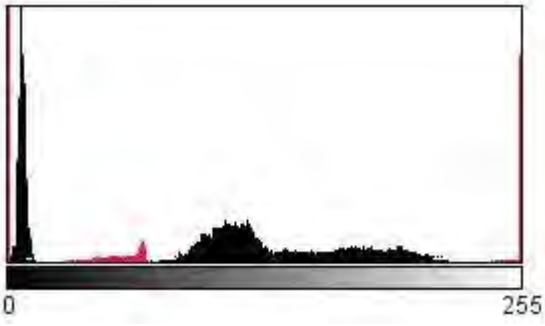
c) Resultado de la comparación de histogramas. En color negro: resultado del tejido real. En color rojo: resultado de la textura generada por computadora.


Texturas 2D de tejido real (apariencia rugosa) Modelo HSV
Matiz (H)
Saturación (S)
Valor (V)
Conversión de modelos HSV a RGB
Textura 2D tejido real (RGB)
Texturas 2D de tejido generado (amarillo - pardo) Modelo HSV
Matiz (H)
Saturación (S)
Valor (V)
Conversión de modelos HSV a RGB
Textura 2D generada (RGB)



a) Histograma en modelo de color HSV correspondiente a la textura real del tejido (apariencia rugosa).

b) Histograma en modelo de color HSV correspondiente a la textura generada con apariencia (apariencia rugosa).



c) Resultado de la comparación de histogramas. En color negro: resultado del tejido real. En color rojo: resultado de la textura generada por computadora.

4.5 Comparación de tiempos en la generación de texturas

Otro parámetro importante a considerar es el tiempo de generación de texturas. Para lo cual se hicieron diferentes pruebas con diferentes equipos, para conocer el tiempo que tomaba generar la síntesis de textura procedural.

El tiempo se midió considerando el tamaño del cubo de textura a generar, desglosado en horas, minutos y segundos. Obteniendo los siguientes resultados.

Equipo 1

Características Equipo: Hewlett-Packard Pavilion dv2500 Notebook PC

Procesador Intel (R) Core (TM)² Duo 1.50 GHz. RAM 2.00 GB, Sistema operativo de 32 bits Windows Vista Service Pack 1. Disco Duro: 160 GB.

Generación Imágenes 3D	Tamaño del cubo a generar (x,y,z)	Tiempo
Próstata Cerrada	16	00 : 00 : 04
Próstata Cerrada	32	00: 00 : 31
Próstata Cerrada	64	00 : 04 : 04
Próstata Cerrada	96	00 : 14 : 19
Próstata Cerrada	128	00 : 37 : 36

Equipo 2

Características Equipo: Dell. Pentium 4

Procesador Intel (R) Pentium(R) 4 CPU 3.00 GHz 2.99 GHz. RAM 1.00 GB, Sistema operativo Microsoft Windows XP Versión 2002. Service Pack 3. Disco Duro: 149 GB.

Generación Imágenes 3D	Tamaño del cubo a generar (x,y,z)	Tiempo
Próstata Cerrada	16	00 : 00 : 15
Próstata Cerrada	32	00 : 02 : 15
Próstata Cerrada	64	0 : 16 : 27
Próstata Cerrada	96	00 : 38 : 12
Próstata Cerrada	128	01 : 16 : 25

Equipo 3

Características Equipo: Dell. Intel Core

Procesador Intel (R) Core (TM)² CPU 6600 @ 2.40 GHz 2.39 GHz. RAM 1.00 GB, Sistema operativo de 32 bits Windows Vista Ultimate Service Pack 1. Disco Duro: 146 GB.

Generación Imágenes 3D	Tamaño del cubo a generar (x,y,z)	Tiempo
Próstata Cerrada	16	00 : 00 : 07
Próstata Cerrada	32	00 : 01 : 05
Próstata Cerrada	64	0 : 07 : 07
Próstata Cerrada	96	00 : 22 : 14
Próstata Cerrada	128	00 : 55 : 35

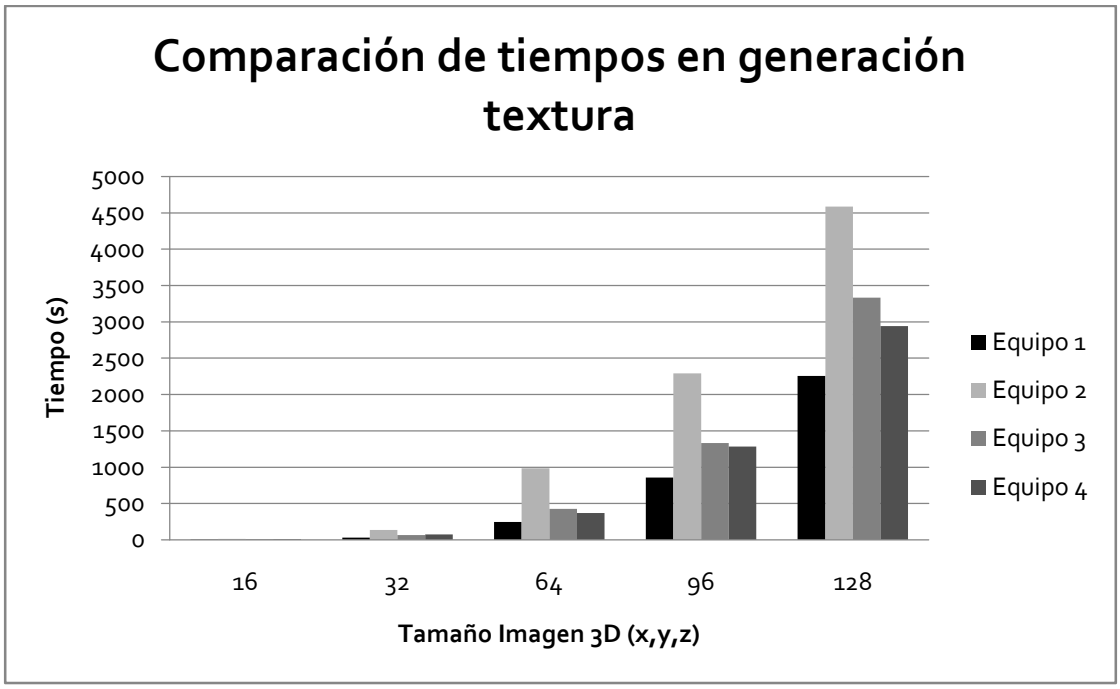
Equipo 4

Características Equipo: Sun Microsystems

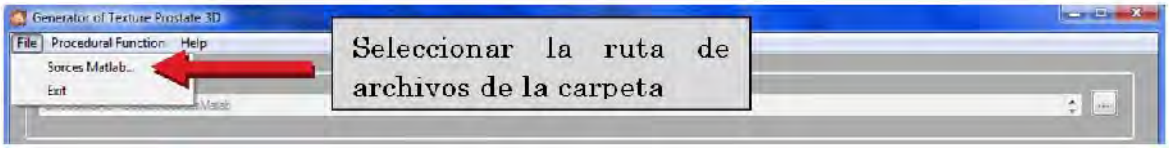
Procesador Dual-Core AMD Opteron™ CPU 1222; 3.00 GHz. RAM 4.00 GB, Sistema operativo Microsoft Windows XP Professional x 64 Edition, Service Pack 2. Disco Duro: 232 GB.

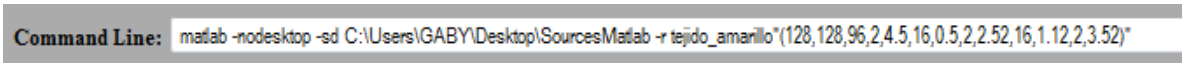
Generación Imágenes 3D	Tamaño del cubo a generar (x,y,z)	Tiempo
Próstata Cerrada	16	00 : 00 : 08
Próstata Cerrada	32	00 : 01 : 13
Próstata Cerrada	64	0 : 06 : 08
Próstata Cerrada	96	00 : 21 : 23
Próstata Cerrada	128	00 : 49 : 04

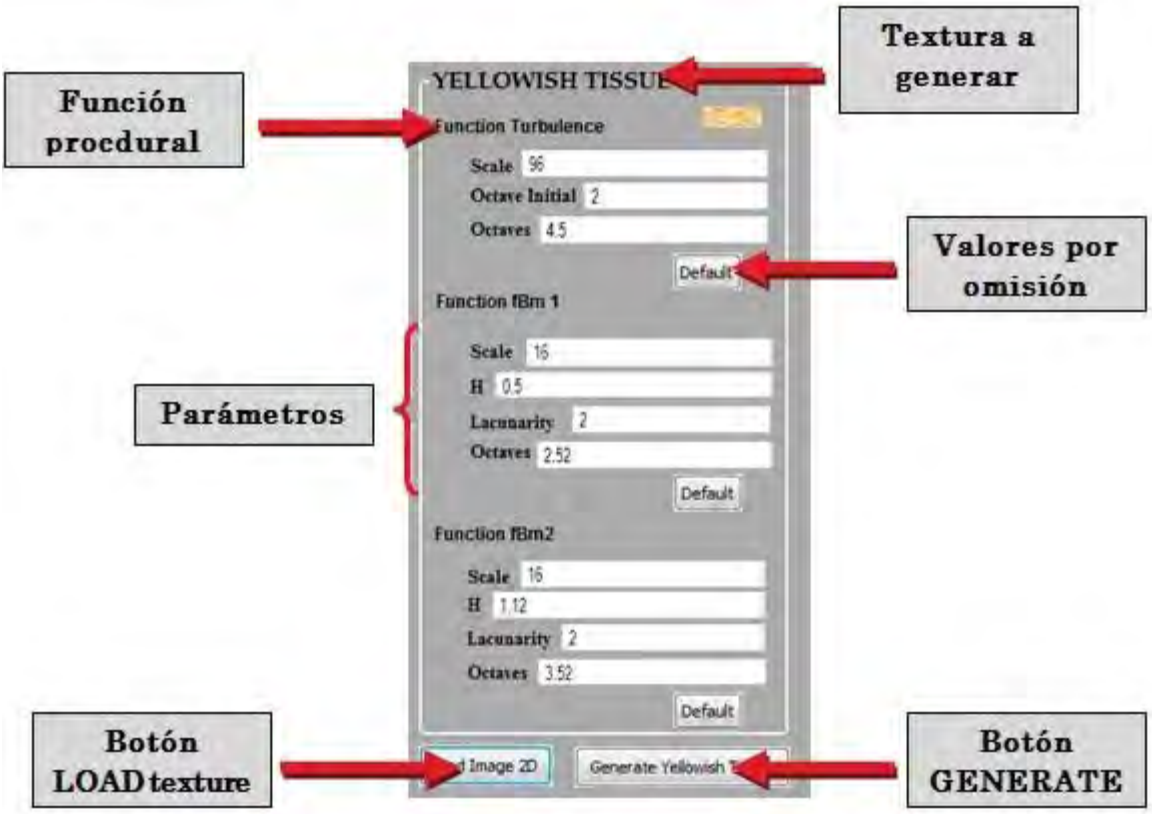
Dichas pruebas sirven para detectar y mejorar el código fuente en la generación de cada textura de tejido, logrando así que sea más efectiva la generación de la textura volumétrica final.

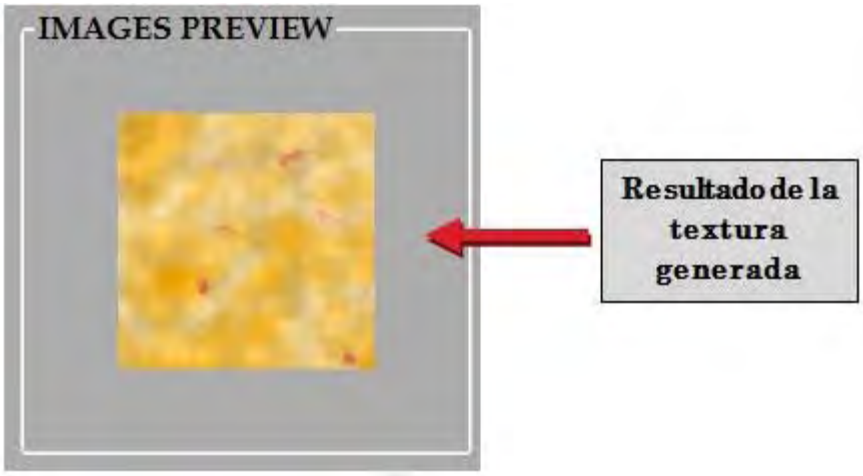

Comparación de tiempos en generación textura
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Tiempo (s)
16
32
64
96
128
Tamaño Imagen 3D (x,y,z)
Equipo 1
Equipo 2
Equipo 3
Equipo 4

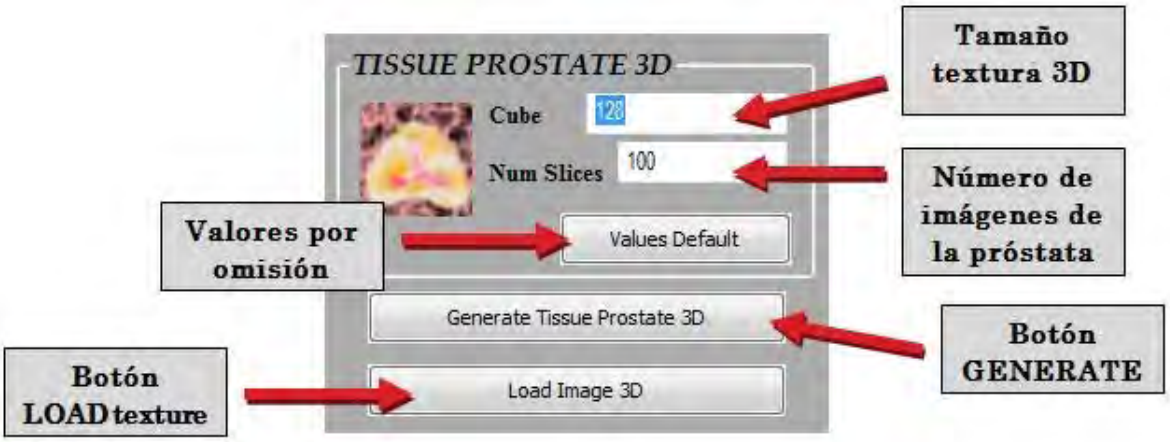

Generator of Texture Prostate 3D
File
Procedural Function
Help
Files
VEINS TISSUE
Function Hetero Terrain
H 1.2
Lacunarity 1
Offset 1.5
Octaves 4.63
Default
Function Turbulence
Scale 128
Octave Initial 4
Octaves 4.87
Factor 3.24
Default
Function fBm
Scale 16
H 1.12
Lacunarity 2
Octaves 3.52
Default
Load Image 2D
Generate Veins Tissue
YELLOWISH TISSUE
Function Turbulence
Scale 96
Octave Initial 2
Octaves 4.5
Default
Function fBm 1
Scale 16
H 0.5
Lacunarity 2
Octaves 2.32
Default
Function fBm2
Scale 16
H 1.12
Lacunarity 2
Octaves 3.52
Default
Load Image 2D
Generate Yellowish Tissue
ROUGHNESS TISSUE
Function Turbulence
Scale 64
Octave Initial 3
Octaves 4.5
Default
Color Gris
Load Image 2D
Generate Roughness Tissue
TISSUE PROSTATE 3D
Cube 128
Num Slices 100
Values Default
Generate Tissue Prostate 3D
Load Image 3D
IMAGES PREVIEW
Command Line: Code

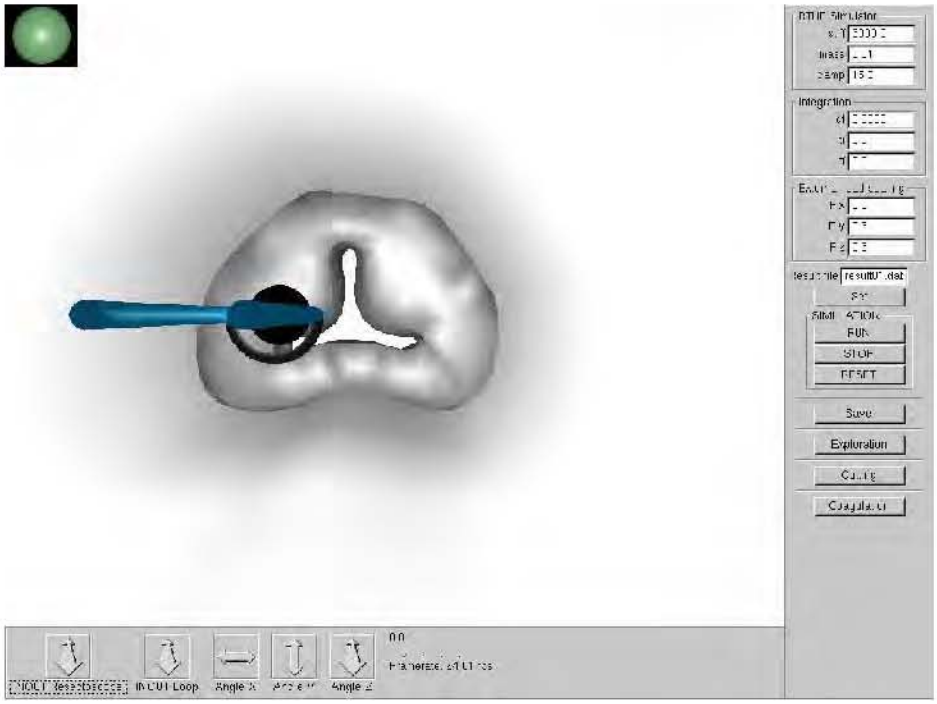

Generator of Texture Prostate 3D
File
Procedural Function
Help
Sorces Matlab...
Exit
Seleccionar la ruta de archivos de la carpeta

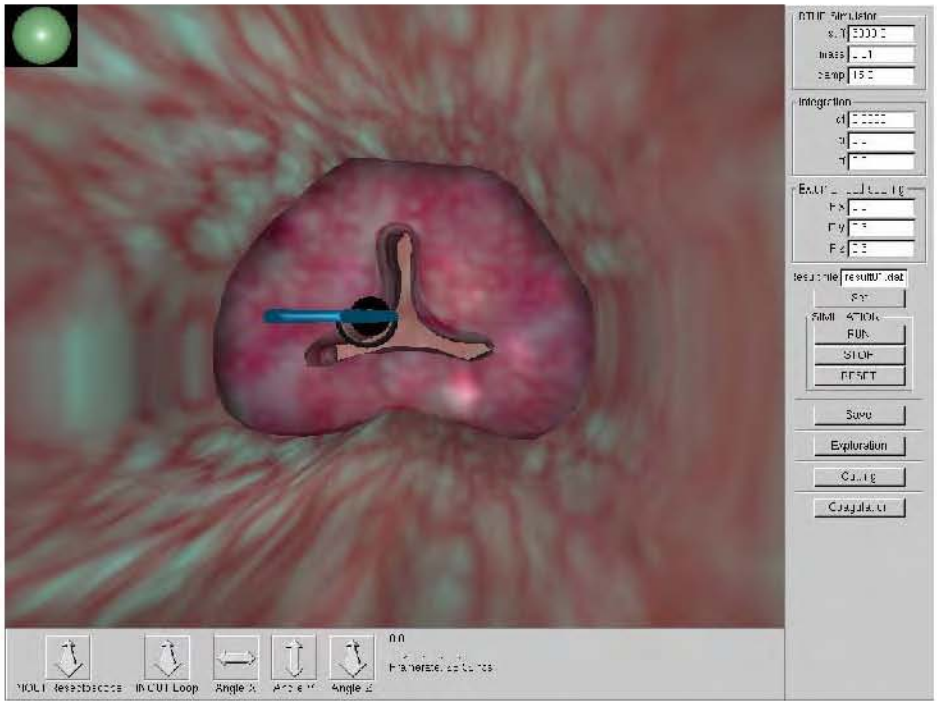

Command Line:
matlab -nodesktop -sd C:\Users\GABY\Desktop\SourcesMatlab -r tejido_amarillo"(128,128,96,2,4.5,16,0.5,2,2.52,16,1.12,2,3.52)"

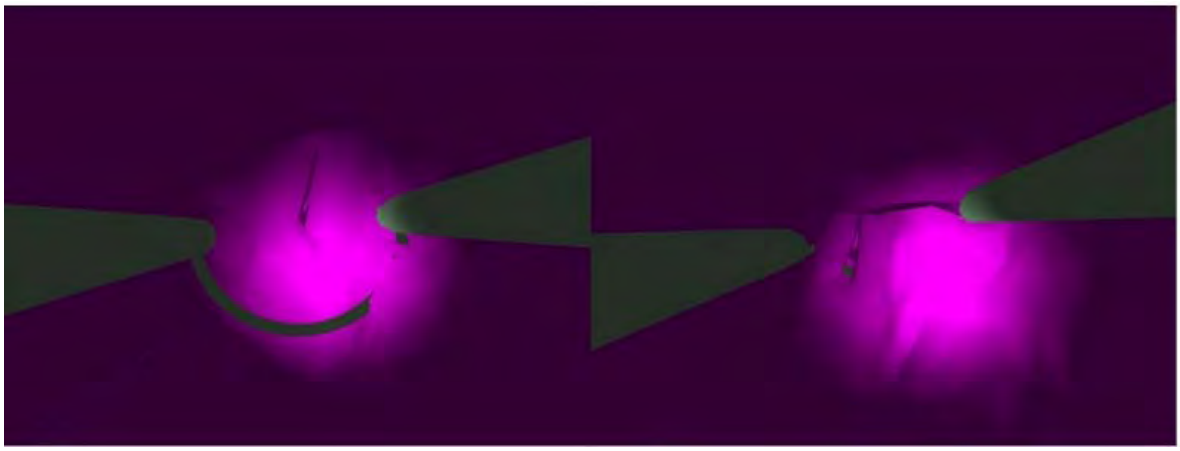

Textura a generar
Función procedural
YELLOWISH TISSUE
Function Turbulence
Scale 96
Octave Initial 2
Octaves 4.5
Default
Valores por omisión
Function fBm 1
Scale 16
H 0.5
Lacunarity 2
Octaves 2.52
Default
Parámetros
Function fBm2
Scale 16
H 1.12
Lacunarity 2
Octaves 3.52
Default
Botón LOAD texture
Image 2D
Generate Yellowish
Botón GENERATE

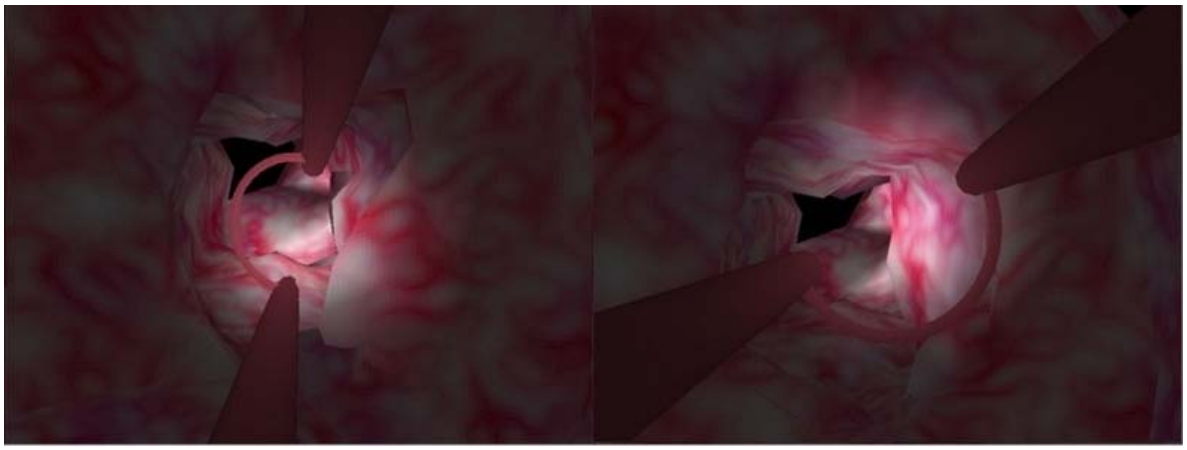

IMAGES PREVIEW
Resultado de la textura generada

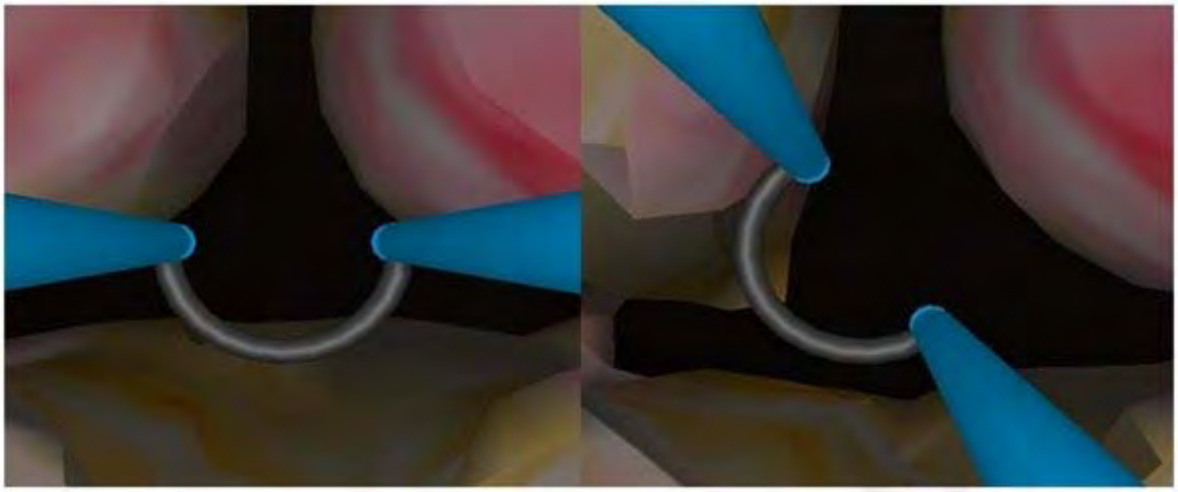

TISSUE PROSTATE 3D
Cube
128
Num Slices
100
Values Default
Generate Tissue Prostate 3D
Load Image 3D
Tamaño textura 3D
Número de imágenes de la próstata
Valores por omisión
Botón GENERATE
Botón LOAD texture

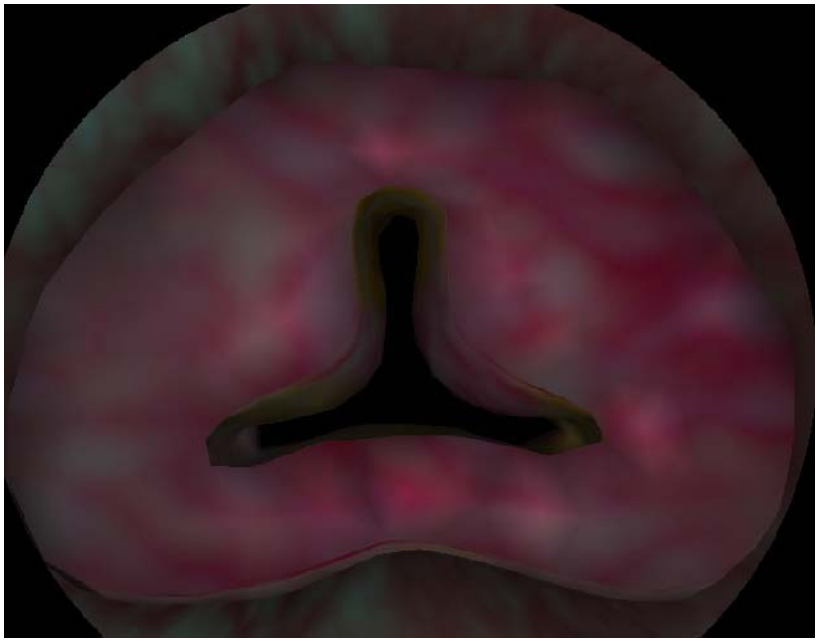

Integration
Save
Exploration
Coagulation
IN OUT Loop
Angle X

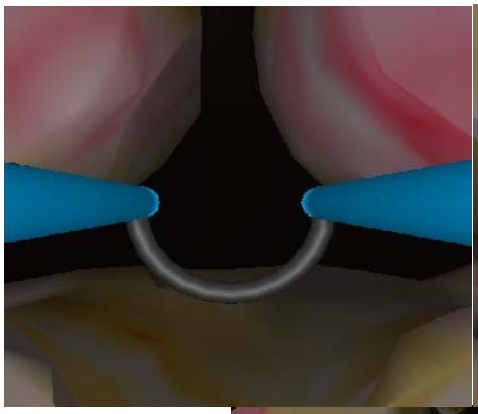

Integration
Save
Exploration
Coagulation
IN OUT Loop
Angle X

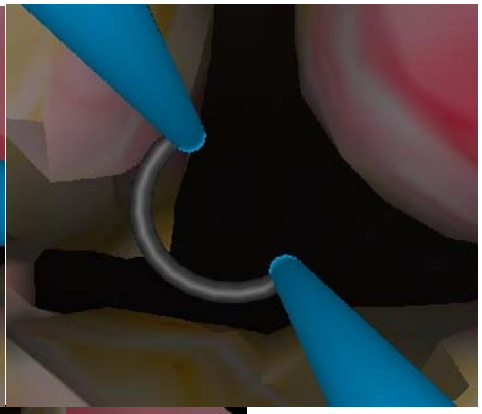


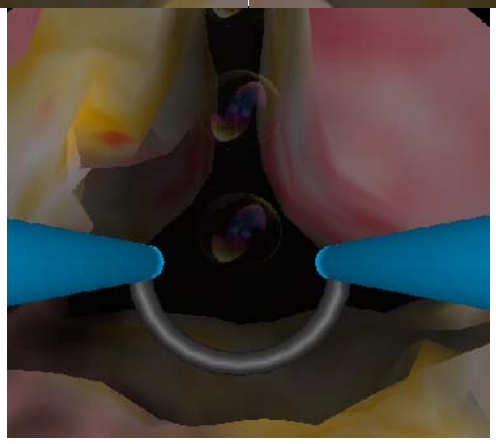












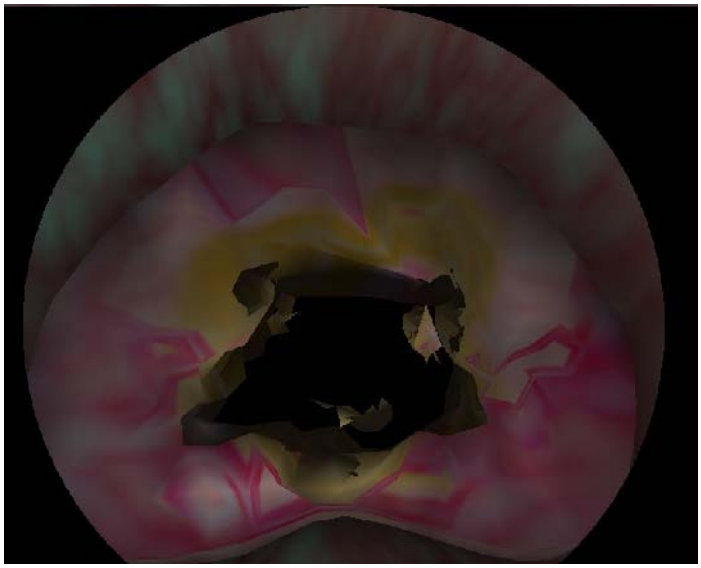
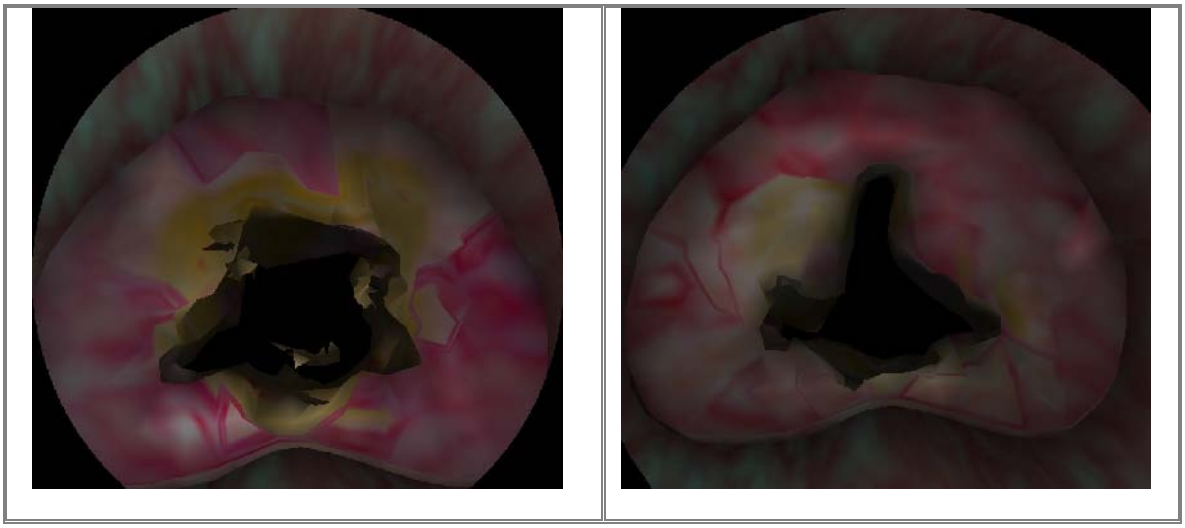


Figura 4.15 Imagen del simulador virtual RTU, una vez realizados algunos cortes.

Al verificar los resultados visuales de la figura anterior, podemos comentar que se ha logrado el objetivo principal de este trabajo de tesis. Ya que los efectos visuales que se aprecian en el simulador virtual de cirugía se acercan mucho a las condiciones reales que se presentan en el procedimiento de resección transuretral de próstata.



5

CAPÍTULO

5. CONCLUSIONES Y PERSPECTIVAS

La realización de este trabajo de tesis consistió en el desarrollo de un modelo generador de texturas procedurales con apariencia de tejido blando real, para ser utilizadas en un simulador de entrenamiento del procedimiento de resección transuretral de próstata (RTU) desarrollado en el laboratorio de Análisis de Imágenes y Visualización CCADET, UNAM. Dichas texturas sintéticas debían cumplir con ser visualmente muy parecidas al procedimiento real visto en previas imágenes médicas.

Una vez hecho la integración de funciones básicas para la generación de texturas procedurales, se realizaron algunas pruebas pudiendo llegar a las siguientes conclusiones.

5.1 Alcances generales

Es importante mencionar que la apariencia de las imágenes reales del tejido blando de la próstata, cuentan con tonalidades y efectos de apariencia irregular, con lo cual es necesario trabajar con elementos que nos faciliten la simulación para lograr el mismo efecto en las texturas generadas por computadora, la forma que se encontró más viable para lograr dichos efectos fue con el uso de métodos de texturas procedurales para su generación.

Se logro generar una textura con vascularidades, logrando que algunas de las vascularidades se visualicen de un color rojo más encendido. Lo cual nos da impresión de tener un tejido inflamado o con una mayor concentración de sangre, tal y como se muestran las imágenes medicas reales.

Se logro obtener un volumen de textura (textura 3D), ayudando notablemente en la apariencia visual del simulador, ya que nos da la impresión de contar con un material continúo, sin costuras o artefactos no deseados. Contando con ello a contar con un material "sólido" con el cual seremos capaces de hacer cortes transversales sobre el material no perdiendo la calidad de la textura.

El modelo de síntesis de texturas procedurales además de mejorar la apariencia visual sobre el simulador; entre otros efectos, también se logro obtener una apariencia visual diferente al realizar la cauterización del tejido, al agregar el matiz amarillo–pardo, el cual se visualiza en el modelo al realizar algunos cortes o cauterización del tejido.

Al utilizar funciones procedurales para la generación de dichas texturas, contamos con algunas de sus ventajas que es no tener un tamaño fijo, lo cual se pueden generar imágenes de diferentes tamaños adaptándolos a las dimensiones que se requieran sin perder la calidad de detalle en la textura.

Otra ventaja de la construcción de texturas procedurales, es que al contar con los algoritmos base de las funciones básicas para la generación de texturas (*turbulencia, turbulencia modificada, terreno heterogéneo y fBm*), se cuenta adicionalmente con una herramienta general para la generación de diferentes texturas, las cuales pueden ser aplicables a otros modelos virtuales o de simulación computarizada.

Una de las desventajas al usar este método es que muchas veces los resultados no pueden ser 100% predecibles, ocasionando que el obtener la textura que deseemos tendremos que hacerlo mediante técnicas estadísticas muchas de ellas a base de prueba y error, tarea que nos puede llevar mucho tiempo.

Otra desventaja en la generación de textura por medio de un método procedural, es que dependiendo de la complejidad de los cálculos puede llevarse mucho tiempo de proceso en el equipo de cómputo. Lo cual se recomienda utilizar una computadora con buena capacidad de procesamiento.

En general puedo mencionar que se logro el objetivo de este trabajo de tesis, generando una síntesis de texturas procedurales la cual agrego el detalle necesario para tener un simulador virtual del procedimiento de resección transuretral de próstata que cada vez más se va asemejando a las condiciones reales.

5.2 Trabajo a futuro

El trabajo a futuro por hacer sobre este sistema de simulación se puede comentar lo siguiente.

En términos de texturización se debe seguir trabajando en mejorar aún más los detalles del tejido como son las tonalidades, las formas y la apariencia de esponjosidad en algunos partes de la glándula (próstata), dicho efecto debe venir directamente de la imagen de textura en 3D.

Otro a trabajo a futuro consiste en la implementación de sangrado. Este deberá dispararse cuando se toque o corte con el elemento de resección alguna de las vascularidades inflamadas o congestionadas.

Como trabajo futuro también queda la implementación de los algoritmos y funciones procedurales bases para la generación de texturas en un lenguaje de alto nivel (C++), incorporando en el código fuente las librerías de la tarjeta gráfica, esto con la finalidad de que los cálculos de generación de texturas se lleven a cabo en ella, logrando así que la generación de texturas se procese más rápido y eficientemente, además de contar con la posibilidad de agregar más complejidad sin que esto sacrifica el tiempo de generación y sobre todo ahorrando espacio en memoria para almacenar las texturas pre-calculadas.

Se cuenta además con una GCI para ser usada en un futuro en la generación de una base de imágenes procedurales capaces de ser usadas a diferente modelos o casos clínicos en la laboratorio de Análisis de Imágenes y Visualización, CCADET UNAM.

Finalmente para contar con una herramienta más completa capaz generar una colección más amplia de imágenes de texturas, se deben de incluir más funciones procedurales. Teniendo con ello la gran ventaja de conocer la “función base” o “fórmula” que con otros programas no podemos conocer. Con lo cual podemos adecuarlos a las necesidades que se requieran y obtener exactamente la imagen que necesitemos.

A. Código fuente: Pila de Tejido de la Próstata en 3D

```
                              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                              %%%%%%%%%%%%%%%%% TISSUE PROSTATE 3D%%%%%%%%%%%%%%%%%%%%%%%%
                              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function  tissueProstate3D(cube, numslices,color_rugoso, var_h_heteroterrain,var_lacunarity_heteroterrain,
var_offset_heteroterrain, var_octaves_heteroterrain, veins_scale_turbulence, veins_inioctave_turbulence,
veins_octaves_turbulence, veins_factor_turbulence, veins_scale_fbm, veins_h_fbm, veins_lacunarity_fbm,
veins_octaves_fbm, yellow_scale_turbulence, yellow_octaves_turbulence, yellow_initoctave_turbulence,
yellow_scale_fbm1, yellow_h_fbm1, yellow_lacunarity_fbm1,  yellow_octaves_fbm1,yellow_scale_fbm2,
yellow_h_fbm2, yellow_lacunarity_fbm2, yellow_octaves_fbm2, roughness_scale_turbulence,
roughness_initoctave_turbulence, roughness_octaves_turbulence)

%INITIAL VALUES
t1=fix(clock);

%SIZE OF CUBE (X,Y,Z)
xmax = cube;
ymax = cube;
zmax = cube;

%FILES IMAGES
prefixfile='3D\\';
filesImage = 'ProstataAbierta\\Region%d.tif';
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% INITIATION OF NOISE AND SPECTRAL FUNCTIONS %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[p, g1, g2, g3] = init_noise;
spectralweights_heteroterrain = spectralWeights(var_h_heteroterrain, var_lacunarity_heteroterrain,
var_octaves_heteroterrain);   %TISSUE VARIATION (HeteroTerrain)
spectralweights_fbm_veins = spectralWeights(veins_h_fbm, veins_lacunarity_fbm, veins_octaves_fbm);   %GROUND
TISSUE (fbM)
spectralweights_fbm1_yellow = spectralWeights(yellow_h_fbm1, yellow_lacunarity_fbm1, yellow_octaves_fbm1);
%GROUND TISSUE (fbM)
spectralweights_fbm2_yellow = spectralWeights(yellow_h_fbm2, yellow_lacunarity_fbm2, yellow_octaves_fbm2);
%GROUND TISSUE (fbM)
%spectralweights_fbm_roughness = spectralWeights(roughness_h_fbm, roughness_lacunarity_fbm,
roughness_octaves_fbm);   %GROUND TISSUE (fbM)

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PRECOMPUTES THE ROTATION MATRICES FOR ANTIALIASING %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:veins_octaves_turbulence,
    if i < 6, RAUX=zeros(3,3); RAUX(1,1)=1.0; RAUX(2,2)=1.0; RAUX(3,3)=1.0;
    else RAUX=rand(3);
    end;
    RAUX(1,4)=0.0; RAUX(2,4)=0.0; RAUX(3,4)=0.0;
    RAUX(4,1)=0.0; RAUX(4,2)=0.0; RAUX(4,3)=0.0; RAUX(4,4)=1.0;
    R2(:,:,i)=RAUX;
end
for i=1:veins_octaves_fbm,
    if i < 6, RAUX=zeros(3,3); RAUX(1,1)=1.0; RAUX(2,2)=1.0; RAUX(3,3)=1.0;
    else RAUX=rand(3);
    end;
    RAUX(1,4)=0.0; RAUX(2,4)=0.0; RAUX(3,4)=0.0;
    RAUX(4,1)=0.0; RAUX(4,2)=0.0; RAUX(4,3)=0.0; RAUX(4,4)=1.0;
    R3(:,:,i)=RAUX;
end
for i=1:yellow_octaves_turbulence,
    if i < 6, RAUX=zeros(3,3); RAUX(1,1)=1.0; RAUX(2,2)=1.0; RAUX(3,3)=1.0;
    else RAUX=rand(3);
    end;
    RAUX(1,4)=0.0; RAUX(2,4)=0.0; RAUX(3,4)=0.0;
    RAUX(4,1)=0.0; RAUX(4,2)=0.0; RAUX(4,3)=0.0; RAUX(4,4)=1.0;
    R4(:,:,i)=RAUX;
end
```

```
for i=1:yellow_octaves_fbm1,
    if i < 6, RAUX=zeros(3,3); RAUX(1,1)=1.0; RAUX(2,2)=1.0; RAUX(3,3)=1.0;
    else RAUX=rand(3);
    end;
    RAUX(1,4)=0.0; RAUX(2,4)=0.0; RAUX(3,4)=0.0;
    RAUX(4,1)=0.0; RAUX(4,2)=0.0; RAUX(4,3)=0.0; RAUX(4,4)=1.0;
    R5(:,:,i)=RAUX;
end
for i=1:yellow_octaves_fbm2,
    if i < 6, RAUX=zeros(3,3); RAUX(1,1)=1.0; RAUX(2,2)=1.0; RAUX(3,3)=1.0;
    else RAUX=rand(3);
    end;
    RAUX(1,4)=0.0; RAUX(2,4)=0.0; RAUX(3,4)=0.0;
    RAUX(4,1)=0.0; RAUX(4,2)=0.0; RAUX(4,3)=0.0; RAUX(4,4)=1.0;
    R6(:,:,i)=RAUX;
end
for i=1:roughness_octaves_turbulence,
    if i < 6, RAUX=zeros(3,3); RAUX(1,1)=1.0; RAUX(2,2)=1.0; RAUX(3,3)=1.0;
    else RAUX=rand(3);
    end;
    RAUX(1,4)=0.0; RAUX(2,4)=0.0; RAUX(3,4)=0.0;
    RAUX(4,1)=0.0; RAUX(4,2)=0.0; RAUX(4,3)=0.0; RAUX(4,4)=1.0;
    R7(:,:,i)=RAUX;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% COMPUTES THE 3D DENSITY FUNCTION(TISSUE 3D) %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for z=1:zmax,

    z
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%% VEINS TISSUE (EXTERNAL) %%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Function_heteroterrain3D = tex_heteroterrainSlice( xmax, ymax, zmax, z, var_h_heteroterrain,
var_lacunarity_heteroterrain, var_octaves_heteroterrain, var_offset_heteroterrain,
spectralweights_heteroterrain, p, g3 );
    Function_heteroterrain3D = 0.2*ones(size(Function_heteroterrain3D))+Function_heteroterrain3D; %%ATTENUATION
FUNCTION
    Function_heteroterrain3D =(1/max(max(Function_heteroterrain3D)))*Function_heteroterrain3D; %NORMALIZE
FUNCTION

    Function_turb_veins3D = tex_turbSlice(xmax, ymax, zmax, z, veins_scale_turbulence,
veins_octaves_turbulence, veins_inioctave_turbulence, p, g3, R2);
    Function_turb_veins3D =(Function_turb_veins3D*(0.3*(veins_factor_turbulence))); %MULTIPLY FUNCTION BY A
CONSTANTE
    Function_turb_veins3D = ones(size(Function_turb_veins3D))-Function_turb_veins3D; %INVERT COLOR OF FUNCTION
    Function_turb_veins3D =(1/max(max(Function_turb_veins3D)))*Function_turb_veins3D;

    Function_fBm_veins3D = tex_fBmSlice(xmax, ymax, zmax, z, veins_scale_fbm, veins_h_fbm,
veins_lacunarity_fbm, veins_octaves_fbm, p, g3, spectralweights_fbm_veins, R3);
    Function_fBm_veins3D =0.92*ones(size(Function_fBm_veins3D))+0.1*Function_fBm_veins3D; %ATTENUATION FUNCTION
    Function_fBm_veins3D =(1/max(max(Function_fBm_veins3D)))*Function_fBm_veins3D; %NORMALIZE FUNCTION

    Result_Heteroterrain_x_Turbulence=Function_heteroterrain3D.*Function_turb_veins3D; %MULTIPLY FUNCTION
HETEROTERRAIN_3D BY FUNCTION TURBULENCE_3D
       Function_Value=0.92*ones(size(Result_Heteroterrain_x_Turbulence))+0.1*Result_Heteroterrain_x_Turbulence;

       %CONVERTION OF HSV TO RGB
    HSV=[];
    HSV(:,:,1)=Function_fBm_veins3D; HSV(:,:,2)=Result_Heteroterrain_x_Turbulence; HSV(:,:,3)=Function_Value;
    TEX=hsv2rgb(HSV);

       %SEPARATE CANALS OF COLOR (RGB)
    TISSUEURETHRA_R(:,:,z) = TEX(:,:,1);
    TISSUEURETHRA_G(:,:,z) = TEX(:,:,2);
    TISSUEURETHRA_B(:,:,z) = TEX(:,:,3);



    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%% YELLOWISH TISSUE  %%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Function_turb_yellow3D = tex_turbSlice(xmax, ymax, zmax, z, yellow_scale_turbulence,
```

```
yellow_octaves_turbulence, yellow_initoctave_turbulence, p, g3, R4);
    Function_turb_yellow3D=0.4*ones(size(Function_turb_yellow3D))+0.9*Function_turb_yellow3D;
    Function_turb_yellow3D=(1/max(max(Function_turb_yellow3D)))*Function_turb_yellow3D; %NORMALIZE FUNCTION

    Function_fBm1_yellow = tex_fBmSlice(xmax, ymax, zmax, z, yellow_scale_fbm1, yellow_h_fbm1,
yellow_lacunarity_fbm1, yellow_octaves_fbm1, p, g3, spectralweights_fbm1_yellow, R5);
    Function_fBm1_yellow = ones(size(Function_fBm1_yellow)) - Function_fBm1_yellow; %INVERT COLOR OF FUNCTION
    Function_fBm1_yellow=(1/max(max(Function_fBm1_yellow)))*Function_fBm1_yellow; %NORMALIZE FUNCTION

    Result_Turbulence_x_fBm_Yellow = Function_turb_yellow3D.*Function_fBm1_yellow; %MULTIPLY FUNCTION
TURBULENCE BY FUNCTION fBm
    Result_Threshold_Turb_x_fBm = threshold2(Result_Turbulence_x_fBm_Yellow,30); %APPLY THRESHOLD AT MULTIPLY
FINAL OF FUNCTION TURBULENCE BY FUNCTION fBm1

    Function_fBm2_yellow = tex_fBmSlice(xmax, ymax, zmax, z, yellow_scale_fbm2, yellow_h_fbm2,
yellow_lacunarity_fbm2, yellow_octaves_fbm2, p, g3, spectralweights_fbm2_yellow, R6);
    Function_fBm2_yellow = 0.8*ones(size(Function_fBm2_yellow))+0.2*Function_fBm2_yellow; % ATTENUATION
FUNCTION
    Function_fBm2_yellow=(1/max(max(Function_fBm2_yellow)))*Function_fBm2_yellow; %NORMALIZE FUNCTION

       %CONVERTION OF HSV TO RGB
    HSV=[];
    HSV(:,:,1)=Result_Threshold_Turb_x_fBm; HSV(:,:,2)=Function_turb_yellow3D; HSV(:,:,3)=Function_fBm2_yellow;
    TEX=hsv2rgb(HSV);

    %SEPARATE CANALS OF COLOR (RGB)
    TISSUEPROSTATE_R(:,:,z) = TEX(:,:,1);
    TISSUEPROSTATE_G(:,:,z) = TEX(:,:,2);
    TISSUEPROSTATE_B(:,:,z) = TEX(:,:,3);


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%% ROUGHNESS TISSUE     %%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Function_roughness_3D = tex_turbSliceAnisotropic(xmax, ymax, zmax, z, roughness_scale_turbulence,
10*roughness_scale_turbulence, roughness_scale_turbulence, roughness_octaves_turbulence,
roughness_initoctave_turbulence, p, g3, R7);
    Function_roughness_3D=(1/max(max(Function_roughness_3D)))*Function_roughness_3D; %NORMALIZE FUNCTION

    %MATRIX_ZEROS=zeros(size(Function_roughness_3D)); %GENERATE A MATRIX OF ZEROS
    %MATRIX_ONES=ones(size(Function_roughness_3D))/1; %GENERATE A MATRIX OF ONES
    %MATRIX_SCALE_GRAY=ones(size(Function_roughness_3D))/2.5;   %GENERATE A MATRIX OF GRAY SCALE

       %ADD THE VALOR DEFAULT IF THE STRING DOESN´T TRY THE VALUE OF COLOR
    if (color_rugoso > 4) || (color_rugoso <= 0)
    color_rugoso=4;
    end


       %SELECTION COLOR FINAL OF THE CAPSULE TEXTURE
       %COLOR= 1; FINAL COLOR OF THE TEXTURE= GRAY
       %COLOR= 2; FINAL COLOR OF THE TEXTURE= PINK
       %COLOR= 3; FINAL COLOR OF THE TEXTURE= LIGHT PINK
    if color_rugoso==1
       %CONVERTION OF HSV TO RGB
       MATRIX_ZEROS=zeros(size(Function_roughness_3D)); %GENERATE A MATRIX OF ZEROS
       Function_roughness_3D=ones(size(Function_roughness_3D))-Function_roughness_3D; %INVERT COLOR OF FUNCTION
        HSV=[];
        HSV(:,:,1)=MATRIX_ZEROS; HSV(:,:,2)=MATRIX_ZEROS; HSV(:,:,3)=Function_roughness_3D;
        TEX=hsv2rgb(HSV);

    %SEPARATE CANALS OF COLOR (RGB)
        TISSUECAPSULE_R(:,:,z) = TEX(:,:,1);
        TISSUECAPSULE_G(:,:,z) = TEX(:,:,2);
        TISSUECAPSULE_B(:,:,z) = TEX(:,:,3);

    elseif color_rugoso==2
       %CONVERTION OF HSV TO RGB
       Function_roughness_3D=ones(size(Function_roughness_3D))-Function_roughness_3D; %INVERT COLOR OF FUNCTION
       MATRIX_ZEROS=zeros(size(Function_roughness_3D)); %GENERATE A MATRIX OF ZEROS
    %MATRIX_ONES=ones(size(Function_roughness_3D))/1; %GENERATE A MATRIX OF ONES
       MATRIX_SCALE_GRAY=ones(size(Function_roughness_3D))/2.5;   %GENERATE A MATRIX OF GRAY SCALE
        HSV=[];
        HSV(:,:,1)=MATRIX_ZEROS; HSV(:,:,2)=MATRIX_SCALE_GRAY; HSV(:,:,3)=Function_roughness_3D;
        TEX=hsv2rgb(HSV);
```

```
       %SEPARATE CANALS OF COLOR (RGB)
        TISSUECAPSULE_R(:,:,z) = TEX(:,:,1);
        TISSUECAPSULE_G(:,:,z) = TEX(:,:,2);
        TISSUECAPSULE_B(:,:,z) = TEX(:,:,3);

    elseif color_rugoso==3

       Function_Canal_Saturation= 0.3*ones(size(Function_roughness_3D))+ 0.9*Function_roughness_3D;
    Function_Canal_Value= 0.5*ones(size(Function_roughness_3D))+ 0.1*Function_roughness_3D;
       MATRIX_ZEROS=zeros(size(Function_roughness_3D)); %GENERATE A MATRIX OF ZEROS
    %Function_roughness_3D=Function_roughness_3D.*MATRIX_SCALE_GRAY; %MULTIPLY FUNCTION TURBULENCE MODIFICATE
BY MATRIX_SCALE_GRAY

    %Roughness_fBm_3D = tex_fBmSlice(xmax, ymax, zmax, z, roughness_scale_fbm, roughness_h_fbm,
roughness_lacunarity_fbm, roughness_octaves_fbm, p, g3, spectralweights_fbm_roughness, R8);
    %Roughness_fBm_3D=1.95*ones(size(Roughness_fBm_3D))+0.1*Roughness_fBm_3D; % ATTENUATION FUNCTION
    %Roughness_fBm_3D=(1/max(max(Roughness_fBm_3D)))*Roughness_fBm_3D; %NORMALIZE FUNCTION

       %CONVERTION OF HSV TO RGB
    HSV=[];
    HSV(:,:,1)=MATRIX_ZEROS; HSV(:,:,2)=Function_Canal_Saturation; HSV(:,:,3)=Function_Canal_Value;
    TEX=hsv2rgb(HSV);

       %SEPARATE CANALS OF COLOR (RGB)
    TISSUECAPSULE_R(:,:,z) = TEX(:,:,1);
    TISSUECAPSULE_G(:,:,z) = TEX(:,:,2);
    TISSUECAPSULE_B(:,:,z) = TEX(:,:,3);

       elseif color_rugoso==4

       Function_Canal_Saturation= 0.25*ones(size(Function_roughness_3D))+ 0.1*Function_roughness_3D;
    Function_Canal_Value= 0.9*ones(size(Function_roughness_3D))+ 0.5*Function_roughness_3D;
       MATRIX_ZEROS=zeros(size(Function_roughness_3D)); %GENERATE A MATRIX OF ZEROS
    %Function_roughness_3D=Function_roughness_3D.*MATRIX_SCALE_GRAY; %MULTIPLY FUNCTION TURBULENCE MODIFICATE
BY MATRIX_SCALE_GRAY

    %Roughness_fBm_3D = tex_fBmSlice(xmax, ymax, zmax, z, roughness_scale_fbm, roughness_h_fbm,
roughness_lacunarity_fbm, roughness_octaves_fbm, p, g3, spectralweights_fbm_roughness, R8);
    %Roughness_fBm_3D=1.95*ones(size(Roughness_fBm_3D))+0.1*Roughness_fBm_3D; % ATTENUATION FUNCTION
    %Roughness_fBm_3D=(1/max(max(Roughness_fBm_3D)))*Roughness_fBm_3D; %NORMALIZE FUNCTION

       %CONVERTION OF HSV TO RGB
    HSV=[];
    HSV(:,:,1)=MATRIX_ZEROS; HSV(:,:,2)=Function_Canal_Saturation; HSV(:,:,3)=Function_Canal_Value;
    TEX=hsv2rgb(HSV);

       %SEPARATE CANALS OF COLOR (RGB)
    TISSUECAPSULE_R(:,:,z) = TEX(:,:,1);
    TISSUECAPSULE_G(:,:,z) = TEX(:,:,2);
    TISSUECAPSULE_B(:,:,z) = TEX(:,:,3);
    end
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ROTATION OF FINAL CANALS OF COLOR (RGB) ROUGHNESS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for z=1:zmax,
    for y=1:ymax,
        for x=1:xmax,
            AUX_R(x,z,y) = TISSUECAPSULE_R(x,y,z);
            AUX_B(x,z,y) = TISSUECAPSULE_B(x,y,z);
            AUX_G(x,z,y) = TISSUECAPSULE_G(x,y,z);

         end;
    end;
end;
TISSUECAPSULE_R = AUX_R;
TISSUECAPSULE_G = AUX_G;
TISSUECAPSULE_B = AUX_B;

%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% INTEGRATE THE THREE TEXTURE WITH THE IMAGES %%%%
%%%%    PREVIOUS GENERATE OF THE PROSTATE      %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% VERIFICATE NUM OF IMAGES OF THE PROSTATE
offset = round((zmax - numslices))-2;
%ASSING ZERO WHEN ZMAX <= NUM OF IMAGES OF THE PROSTATE
if offset < 0, offset = 0;
else
%ASSING ONE WHEN ZMAX > NUM OF IMAGES OF THE PROSTATE
 offset=1;
end;
%LOOP FOR ADD THE TEXTURE TO THE IMAGES OF THE PROSTATE
for z=1:zmax,
        if z > offset & z<=zmax-offset %VERIFICATE RANGE THE NUM OF THE IMAGES
        zaux = z; %AUX THAT EXAMINE THE IMAGES ONE BY ONE
             if z<=numslices-1
             file = sprintf(filesImage,zaux-1); % ADD THE ROUTE OF THE FILE
             else
             file = sprintf(filesImage,numslices-1); % ADD THE ROUTE OF THE FILE
             end;
        IMAGEN_REAL = imread(file); % READ THE IMAGES OF THE PROSTATE IN B/W
        IMAGEN_REAL = imresize(IMAGEN_REAL, [xmax ymax], 'nearest' ); % SCALE THE IMAGES (X,Y)

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%% SEPARATION OF CONTOUR OF THE PROSTATE  %%%%%
        %%%%%%%%%%%        IN THE IMAGE                      %%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        IMAGEN_REAL=double(IMAGEN_REAL); % CHANGE VALUES OF IMAGES
        IMAGEN_REAL_COMPLE=ones(size(IMAGEN_REAL))-IMAGEN_REAL; %INVERT COLOR
        CONTORNO_EXT=imfill(IMAGEN_REAL_COMPLE); % CONTOUR EXTERIOR
        CAPSULE=ones(size(CONTORNO_EXT))-CONTORNO_EXT; %INVERT COLOR

        CONTORNO_INT= xor(IMAGEN_REAL,CAPSULE); %%%CONTOUR INTERIOR
        URETHRA=ones(size(CONTORNO_INT))-CONTORNO_INT; % %INVERT COLOR


        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%% EROTION CONTOUR EXTERIOR %%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        se = strel('ball',1,1); % METHOD FOR EROTION, FUNCTION OF MATLAB
        CAPSULE_EROTION = imdilate(CAPSULE,se); % EROTION IMAGE
        h = ones(5,5) / 25; % PARAMETER FOR THE FUNCTION FILTER, VALUE PROPER OF MATLAB
        CAPSULE_ERO_FILTRE= imfilter(CAPSULE_EROTION,h); % FILTER IMAGE
        CAPSULE_ERO_FILTRE=(1/max(max(CAPSULE_ERO_FILTRE)))*CAPSULE_ERO_FILTRE; %NORMALIZE IMAGE


        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%% DILATE CONTOUR INTERIOR %%%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        se1 = strel('rectangle', [5 5]); % METHOD FOR EROTION, FUNCTION OF MATLAB
        URETHRA_DILATE = imerode(URETHRA,se1); % DILATE IMAGE
        h = ones(5,5) / 25; % PARAMETER FOR THE FUNCTION FILTER, VALUE PROPER OF MATLAB
        URETHRA_DILATE_FIL= imfilter(URETHRA_DILATE,h); % FILTER IMAGE
        URETHRA_DILATE_FIL=(1/max(max(URETHRA_DILATE_FIL)))*URETHRA_DILATE_FIL; %NORMALIZE IMAGE


        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%% INTEGRATE OF TEXTURE WITH THE CONTOURS %%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        URETHRA_DILATE_FIL_COMPLE = ones(size(URETHRA_DILATE_FIL))-URETHRA_DILATE_FIL; %IMAGE COMPLEMENT

              % THE MULTIPLICATION OF CONTOUR BY THE  YELLOW TEXTURE PLUS THE MULTIPLICATION OF CONTOUR IMAGE
COMPLEMENT BY THE VEINS TEXTURE FOR EVERY CANAL OF COLOR
        TISSUE_UREandPROS_R(:,:,zaux) = URETHRA_DILATE_FIL_COMPLE.*TISSUEURETHRA_R(:,:,zaux) +
URETHRA_DILATE_FIL.*TISSUEPROSTATE_R(:,:,zaux);
        TISSUE_UREandPROS_G(:,:,zaux) = URETHRA_DILATE_FIL_COMPLE.*TISSUEURETHRA_G(:,:,zaux) +
URETHRA_DILATE_FIL.*TISSUEPROSTATE_G(:,:,zaux);
        TISSUE_UREandPROS_B(:,:,zaux) = URETHRA_DILATE_FIL_COMPLE.*TISSUEURETHRA_B(:,:,zaux) +
URETHRA_DILATE_FIL.*TISSUEPROSTATE_B(:,:,zaux);


        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%% INTEGRATE THE FINAL TEXTURE (CAPSULE) %%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        CAPSULE_ERO_FILTRE_COMPLE = ones(size(CAPSULE_ERO_FILTRE))-CAPSULE_ERO_FILTRE; %IMAGE COMPLEMENT

              %INTEGRATE THE FINAL ROUGHNESS TEXTURE WITH THE RESULT THE  TEXTURIZATE IMAGES
```

```
        TISSUE_R(:,:,zaux) = CAPSULE_ERO_FILTRE.*TISSUECAPSULE_R(:,:,zaux) +
CAPSULE_ERO_FILTRE_COMPLE.*TISSUE_UREandPROS_R(:,:,zaux);
        TISSUE_G(:,:,zaux) = CAPSULE_ERO_FILTRE.*TISSUECAPSULE_G(:,:,zaux) +
CAPSULE_ERO_FILTRE_COMPLE.*TISSUE_UREandPROS_G(:,:,zaux);
        TISSUE_B(:,:,zaux) = CAPSULE_ERO_FILTRE.*TISSUECAPSULE_B(:,:,zaux) +
CAPSULE_ERO_FILTRE_COMPLE.*TISSUE_UREandPROS_B(:,:,zaux);

        %SEPARATE CANALS OF COLOR (RGB) OF THE FINAL IMAGE OBTAINED
        TEX(:,:,1) = TISSUE_R(:,:,zaux);
        TEX(:,:,2) = TISSUE_G(:,:,zaux);
        TEX(:,:,3) = TISSUE_B(:,:,zaux);

    else
              %SEPARATE CANALS OF COLOR (RGB) OF THE FINAL IMAGE OBTAINED
         TEX(:,:,1)= TISSUEURETHRA_R(:,:,z);
         TEX(:,:,2)= TISSUEURETHRA_G(:,:,z);
         TEX(:,:,3)= TISSUEURETHRA_B(:,:,z);
   end;

       % SAVE IMAGE FINAL
    file=sprintf('%s%d.bmp',prefixfile,z);
    imwrite(TEX, file);
end;
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% TIME OF RUN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

hora1= t1(1,4);
min1=t1(1,5);
segundo1=t1(1,6);

t2=fix(clock);
hora2= t2(1,4);
min2=t2(1,5);
segundo2=t2(1,6);

horafinal=hora2-hora1;
minfinal=min2-min1;

if segundo2 > segundo1
    segfinal=segundo2 - segundo1;
else
    segfinal=segundo1-segundo2;
end

if minfinal < 0
    var1= (horafinal*60) + (minfinal);
    var2= (var1/60); var2=fix(var2);
    var3= var1 - (var2*60);
    horafinal=var2;
    minfinal=var3;
end
sprintf ('%d : %d : %d',horafinal,minfinal,segfinal)
exit;
```

BIBLIOGRAFÍA Y REFERENCIAS WEB

Abbou Claude y Dubernard Jean-Michel, *Cirugía de la próstata. Técnicas quirúrgicas, urología. España : Elsevier Masson, 2007.*

Ahearn Luke, *3D game textures. USA : Focal Press, 2006.*

Apodaca Anthony A., Larry Gritz, Ronen Barzel, *Advanced RenderMan: Creating CGI for Motion Pictures. USA : Morgan Kaufmann, 1999.*

Arámbula Cosio F., Padilla Castañeda M., Sevilla Martínez P., *Computer assisted prostate surgery training. International Journal of Humanoid Robotics. Mexico, 2006. - Vol. 3.*

Azarang Mohammad R. y Dunna Eduardo García, *Simulación y análisis de modelos estocásticos. México: McGraw-Hill, 1996.*

Azcárraga Dr. Gustavo*, Urología. México : Mendez Editores, 2000.*

Blender User Guide, *Texture. Septiembre 2004. http://www.blender.org/documentation/htmlI/c4607.html*

Capilla Pascual, *Fundamentos de colorimetría. España : Universidad de Valencia, 2002 .*

CEC Electromedicina, *Manual Electrobisturi URO 400. Argentina : CEC.*

Chen C. H., Pau L. F. y Wang S. P., *Handbook of pattern recognition & computer vision. USA : World Scientific, 1998.*

Clemente Ramos L.M., Rueda F. Ramasco y Platas A., *Síndrome de Reabsorción Post-Resección Transuretral (R.T.U.) de Próstata: Revisión de aspectos fisiopatológicos, diagnóstico y terapéuticos. España : Actas Urológicas Españolas, 2001. - 25 : Vol. 1.*

Ebert David, Musgrave F. Kenton y Peachey, *Texturing & Modeling: A Procedural Approach. USA : Morgan Kaufmann, 2002.*

Elias Hugo *Perlin Noise. 2003. http://freespace.virgin.net/hugo.elias/models/m_perlin.htm*

ERCO-LightScout Guide, *Sombreado y textura. 1997. http://www.erco.com/guide_v2/guide_2/simulation_95/shading_2678/es/es_shading_intro_1.htm.*

Ferguson R. Stuart, *Practical algorithms for 3D computer graphics. USA : A K Peters, 2001.*

Foley James D, Van Dam A., Hughes S.K., *Introducción a la graficación por computador. Delawere, USA : Addison-Wesley Iberoamericana, 1996.*

Gudlaugsson Bjarki, *Procedural Content Generation. USA : Publication, 2006.*

Hearn Donald y Baker M. Pauline, *Gráficas por computadora. México : Prentice Hall Hispanoamericana, 1995.*

Heinz-Otto Peitgen, Hartmut Jürgens, Dietmar Saupe, *Chaos and fractals: new frontiers of science. USA : Springer, 2004.*

Jagnow Robert Carl, *Stereological Techniques for Synthesizing Solid Textures. USA : Massachusetts Institute of Technology, 2004.*

Mandelbrot Benoit, *La Geometria fractal de la naturaleza. Barcelona, España : Tusquets, 1997.*

Mayo Clinic, *Transurethral resection of the prostate (TURP). 2000. - http://www.mayoclinic.com/health/turp/MY00633.*

McConnell Jeffrey J., *Computer Graphics. Theory into practice. EUA : Jones & Bartlett Publishers, 2006.*

Noise Perlin, *Noise and Turbulence. 1997. http://mrl.nyu.edu/~perlin/doc/oscar.html#noise.*

Padilla Castañeda M. y Arámbula Cosío F.*, Deformable for turp sugery simulation . Computer and Graphics 28, 2004.*

Perlin Ken, *Making Noise. Diciembre de 1999. - http://www.noisemachine.com/talk1/.*

Poissant Yves, *Musgrave. Materials Plugins. 2002. http://www.ypoart.com/downloads/Musgrave.htm.*

Resnick Robert, *Física Vol. 1. México : CECSA, 1999.*

Shirley Peter y Ashikhmin Michael, *Fundamentals of computer graphics. USA : A k Peters, 2005.*

Shreiner Dave, Woo Mason y Neider Jackie, *The official Guide to Learning OpenGL. Versión 2.1. Massachusetts, USA : Addison-Wesley, 2007.*

Stewart Ian, *The New Mathematics of Chaos. USA : Penguin Press, 1997.*

Stoelting Robert K. y Dierdorf Stephen F., *Anestesia y enfermedad coexistente. España : Elsevier, 2003. - Vol. Cuarta.*

Texture Procedural, *Procedural Texture. 2006. http://www.timaxmedia.com/html/help/Layer_Type__Procedural_Texture.htm.*

Turcotte Donald Lawson, *Fractals and chaos in geology and geophysics. USA : Cambridge University Press, 1997.*

Turk Greg, *Generating Textures on Arbitrary Surfaces. EUA : (A Thesis Proposal) , 1991.*

Universidad Nacional del Sur Dpto. Ciencias e Ingeniería de la Computación, *Computación Gráfica. 2008. - http://cs.uns.edu.ar/cg/clases.htm.*

Valdéz Soriano David, *Generación de modelos de mallas de tetraedros para simuladores quirúrgicos. México DF : Tesis Profesional. Facultad de Ingeniería. UNAM, 2008.*

Van Leigh, *LightWave 3D 8 Texturing. USA : Wordware Publishing, Inc., 2004.*

Wein Alan J. y Kavoussi Louis R., *Urologia de Campbell & Walsh. USA : Editorial Medica Panamericana, 2008.*