CIS1730CP09

ANE-STENT

Stephanie Dominguez Andrade
Juan Sebastián Espinosa Torres
Jose Antonio Quintero Gómez
David Alonso Villamizar Lizcano

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
2018

CIS1730CP09
ANE-STENT

Autores:

Stephanie Dominguez Andrade
Juan Sebastián Espinosa Torres
Jose Antonio Quintero Gómez
David Alonso Villamizar Lizcano

MEMORIA DE TRABAJO DE GRADO REALIZADA PARA CUMPLIR UNO DE LOS REQUERIMIENTOS PARA EL TÍTULO DE INGENIERÍA DE SISTEMAS

Director

Ing. Leonardo Flórez Valencia, Ph.D.

Jurados del Trabajo de Grado

Ing. José Hernando Hurtado Rojas, M.Sc.

Ing. Jaime Andrés Pavlich Mariscal, Ph.D.

Sitio Web del Trabajo de Grado

http://pegasus.javeriana.edu.co/~CIS1730CP09

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
Mayo, 2018

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERÍA DE SISTEMAS

Rector de la Pontificia Universidad Javeriana

Jorge Humberto Peláez Piedrahita, S.J.

Decano de la Facultad de Ingeniería

Ing. Jorge Luis Sánchez Téllez, M.Sc.

Directora del Programa de Ingeniería de Sistemas

Ing. Mariela Josefina Curiel Huérfano, Ph.D.

Jefe del Departamento de Ingeniería de Sistemas

Ing. Efrain Ortiz Pabón, Ph.D.

Artículo 23 de la Resolución No. 1 de junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

AGRADECIMIENTOS

Le agradecemos a nuestros padres por habernos permitido estudiar en la Pontificia Universidad Javeriana y permitirnos llegar hasta acá. Gracias a Dios, a nuestros amigos, a nuestras parejas y a todas las personas cercanas que hicieron más ameno el tiempo en la carrera y en la ejecución del trabajo de grado.

También agradecemos a Leonardo Flórez-Valencia, nuestro director de trabajo de grado por apoyarnos con su tiempo, conocimiento y motivación en este proyecto.

Agradecemos al ingeniero Jaime Pavlich-Mariscal, por su apoyo durante las actividades de la clase de Planeación del Trabajo de Grado.

Agradecemos también a los profesores que nos han enseñado, y a la Pontificia Universidad Javeriana por todas las oportunidades, herramientas y espacios proporcionados.

CONTENIDO

FIGURAS

TABLAS

ABSTRACT

A proof of concept was developed, to evaluate the feasibility of developing a physics simulation of rigid and soft bodies, integrating *Bullet Physics* with *VTK*. Building upon the research of the *Thrombus-VPH* project.

RESUMEN

Se realizó una prueba de concepto, evaluando la factibilidad del desarrollo de una simulación de física de cuerpos rígidos y suaves, integrando *Bullet Physics* con *VTK*. Continuando así con la investigación del proyecto *Thrombus-VPH*.

I. INTRODUCCIÓN

Los *stents* son cilindros con paredes de malla, que se usan en tratamientos mínimamente invasivos para varias enfermedades cardiovasculares.

Existe una amplia variedad de *stents* en el mercado e incertidumbre sobre la influencia en el proceso de tratamiento de cada variación. Esto dificulta la selección del *stent* óptimo para cada caso médico específico.

Ane-stent es una prueba de concepto de una simulación física hecha en C++ que integra dos librerías de código abierto:

- *Bullet Physics*, una librería de física que permite simular cuerpos rígidos y suaves, con restricciones y en tiempo real.
- *VTK*, una librería de visualización de datos usada frecuentemente en aplicaciones que trabajan con imágenes médicas.

Ane-stent permite a sus usuarios crear visualizaciones de interacción física entre modelos en formatos *VTP* y *VTK*, combinando cuerpos rígidos y suaves con comportamiento específico para cada modelo.

La sección 2 habla del problema con la oportunidad encontrada para resolverlo y lista los objetivos que establecen el alcance del proyecto. La sección 3 describe el contexto del trabajo de grado mediante proyectos similares y aquel en que se inspira Ane-stent: *Thrombus-VPH*. En esta sección también se describen en profundidad las librerías mencionadas. La sección 4 lista los requerimientos y restricciones de la prueba de concepto. La sección 5 contiene el diseño completo de la prueba de concepto. La sección 6 habla de la metodología de desarrollo usada, los hitos propuestos y muestra el estado final alcanzado de la prueba de concepto. Posteriormente, las secciones 7 y 8 tienen los resultados y las conclusiones del proyecto respectivamente.

En continuidad de esto, este trabajo de grado construye una prueba de concepto en la que se integran las tecnologías de *VTK* y *Bullet Physics* para evaluar la viabilidad del uso estas dos tecnologías en conjunto con el fin de responder a la problemática planteada.

El código fuente de la prueba de concepto se encuentra en el siguiente enlace: https://github.com/davl3232/Ane-stent

II. DESCRIPCIÓN GENERAL

1. Oportunidad, problema

1.1. Contexto del problema

Según Courbebaisse et al., "gracias a avances en tecnología, mejoras en visualización y un mercado creciente para dispositivos mínimamente invasivos, actualmente los practicantes de medicina tienen un conjunto de opciones de tratamiento más amplio para tomar una decisión objetiva adaptada de manera cercana a las condiciones específicas del paciente. Sin embargo, hay muchas preguntas sobre la influencia de estos dispositivos en el proceso de curación de aneurismas que permanecen abiertas, requiriendo un mejor y más profundo estudio de estos métodos, combinando aproximaciones numéricas y virtuales con datos específicos de cada paciente." (Courbebaisse, 2014)

Al ser este el contexto del problema del proyecto al cual se está tratando de continuar su investigación, también es el de esta prueba de concepto, sin embargo, el del presente está más específicamente orientado en mejorar lo que ya existe.

Ane-Stent extiende *Thrombus-VPH* y no otros proyectos relacionados (incluso entre aquellos que colaboran) porque además de trabajar junto con la Pontificia Universidad Javeriana, este proyecto tiene como enfoque crear simulaciones que puedan ser interactivas, es decir, se ejecutan en tiempo real para este fin a diferencia de otros

tipos de simulaciones que pueden tardar mayores cantidades de tiempo (debido a que se utilizan para otros fines). (Courbebaisse, 2014)

1.2. Formulación del problema

El trabajo de grado está inspirado en el proyecto que se ha llevado a cabo por Courbebaisse et al. El proyecto que se propone tiene como fin dar continuación a éste, agregando grosor a las paredes de los vasos sanguíneos y el *stent* como un objeto sólido y no una malla solamente.

1.3. Solución propuesta

Se propone investigar acerca de la aplicabilidad de la tecnología de simulación física *Bullet Physics* en el contexto del proceso de postura de un *stent* en los vasos sanguíneos.

Dicha simulación debería mostrar la deformación de un segmento de vaso sanguíneo (e.g. segmentos venas y arterias) y los cambios del flujo sanguíneos causados por la colocación del *stent*.

1.4. Justificación de la solución

Se espera que el resultado de este proyecto pueda ser aplicado en los vasos sanguíneos con aneurismas para aumentar el conocimiento en el proceso de tratamiento de los mismos. Así mismo, se puede entrenar a los médicos en proceso de aprendizaje para la realización de la cirugía de postura de *stent*.

La herramienta *Bullet Physics* brinda una oportunidad de investigación para la representación de la colocación de un *stent* en un vaso sanguíneo con aneurisma, también aumenta para el conocimiento en esta área de la medicina y la computación gráfica. De esta manera, se puede entrenar al médico en proceso de aprendizaje para la realización de la cirugía.

2. Descripción del proyecto

2.1. Objetivo general

Evaluar la factibilidad de implementar una simulación de la física de un segmento de vaso sanguíneo, usando *Bullet Physics*, en el contexto de una cirugía de postura de *stent*.

2.2. Objetivos específicos

- Definir una arquitectura basada en *Bullet Physics*.
- Elaborar una prueba de concepto de una simulación física de un aneurisma con *stent* usando *Bullet Physics*.
- Definir un criterio y usarlo para evaluar la factibilidad de la implementación de la prueba de concepto de la simulación, usando métricas calculadas a partir de mediciones obtenidas durante el desarrollo.
- Elaborar un informe del proyecto en el que se detallan los problemas y soluciones según lo especificado en la plantilla generada por el grupo.

2.3. Entregables, estándares, y justificación

Entregable	Estándares asociados	Justificación
Poster		
SPMP	16326-2009 - ISO/IEC/IEEE Systems and Software Engineering--Life Cycle Processes--Project Management Software Metrics and Software Metrology	Se eligió este estándar en su versión de 2009 ya que es la más reciente con el fin de tomar los elementos a manera de plantilla que puedan adaptarse al desarrollo de este proyecto, también, tomando como referencia el SEBOK, se tiene que éste es el estándar que ahí se utiliza.

SRS	1016-2009 - IEEE Standard for Information Technology--Systems Design--Software Design Descriptions 830-1998 - IEEE Recommended Practice for Software Requirements Specifications	Se escogió el estándar del 2009 ya que especifica el contenido que deben tener las secciones de este, como plantilla se usará el del 1998 ya que, a pesar de estar obsoleto, especifica una plantilla.
SDD	29119-5-2016 - ISO/IEC/IEEE International Standard - Software and systems engineering -- Software testing. Partes 1-5	Se escogió este estándar a que los otros estándares encontrados eran obsoletos y anunciaban que habían sido actualizados al que se va a usar.
Documentación de pruebas	29119-5-2016 - ISO/IEC/IEEE International Standard - Software and systems engineering -- Software testing. Partes 1-5	El estándar fue escogido atendiendo la recomendación del profesor de la asignatura de planeación de trabajo de grado.
Código fuente	Google Style Guides	Google Style Guides provee una serie de parámetros y lineamientos que permiten organizar y el código para tener un control de la complejidad de éste, para así poseer

		un código legible, ordenado. Google Style Guides está disponible para múltiples lenguajes de programación que incluyen: C++, Java, Python, JavaScript, entre otros.
Post-mortem proyecto	A defined process for project post mortem review	Se escoge como guía para la creación de postmostem
Manual de uso, instalación y configuración	26514-2010 - IEEE Standard for Adoption of ISO/IEC 26514:2008 Systems and Software Engineering--Requirements for Designers and Developers of User Documentation	Se escoge con el fin de poder diseñar el contenido del manual de usuario. En primera instancia se optó por el estándar de la ISO, pero por accesibilidad al mismo, se escoge el estándar de la IEEE, ya que por medio de la plataforma de biblos de la Pontificia Universidad Javeriana, se puede acceder al mismo

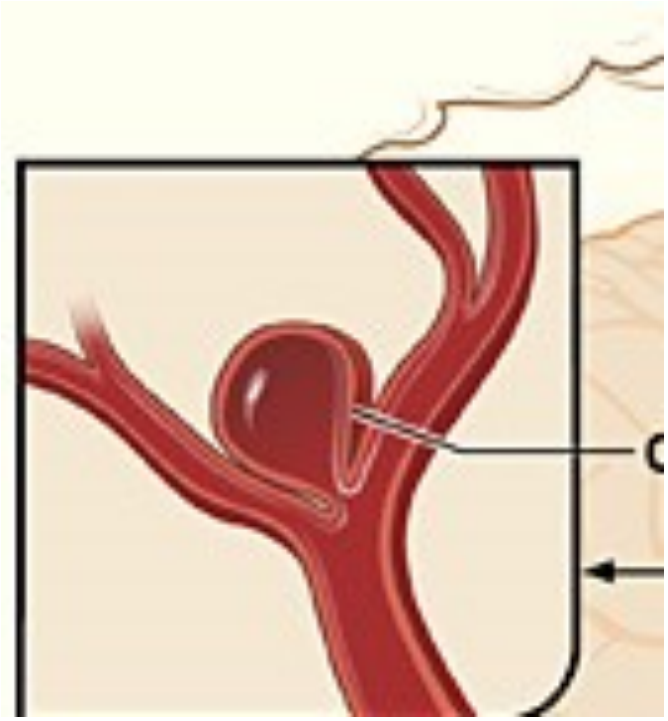
Tabla 1. Entregables, estándares, y justificación

III. CONTEXTO DEL PROYECTO

1. Trasfondo

Un vaso sanguíneo puede ser una vena o una arteria y tiene como fin transportar la sangre por todo el cuerpo (Science Daily, 2006). El flujo de la sangre genera presión sobre las paredes del vaso y que en algunos casos se puede salir de control y generar un aneurisma.

Un aneurisma es una deformación de un vaso sanguíneo, la cual causa una especie de globo que sale del mismo, como se puede ver en la figura 1, teniendo como consecuencia que, al estirar la pared de la vena o arteria, su grosor disminuye haciendo probable que se rompa generando sangrado interno.



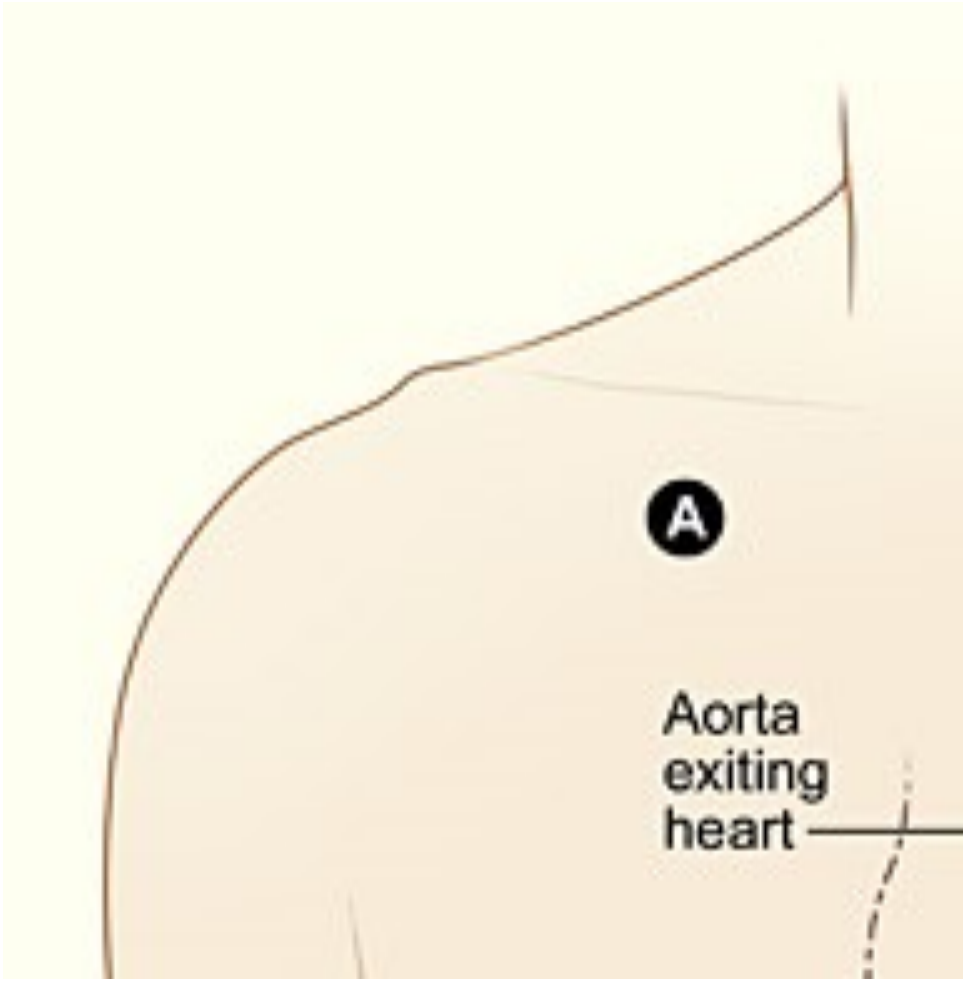


Figura 1. Aneurisma cerebral (izquierda). Aneurisma aórtico (derecha) por (NHLBI, 2018)

Hay múltiples tratamientos para esto, sin embargo, uno de ellos, que interesa para este proyecto, es el tratamiento usando un *stent*. Para su postura, se utiliza un micro catéter más allá del aneurisma y luego se despliega el dispositivo de desviación del flujo a través del cuello del aneurisma, en el vaso principal, este proceso se puede detallar en la figura 2. Inmediatamente el flujo en el aneurisma se reduce (John Hopkins Medicine, 2011) y causa que la sangre dentro del aneurisma se coagule re-forzando la pared afectada. (NHLBI, 2018)

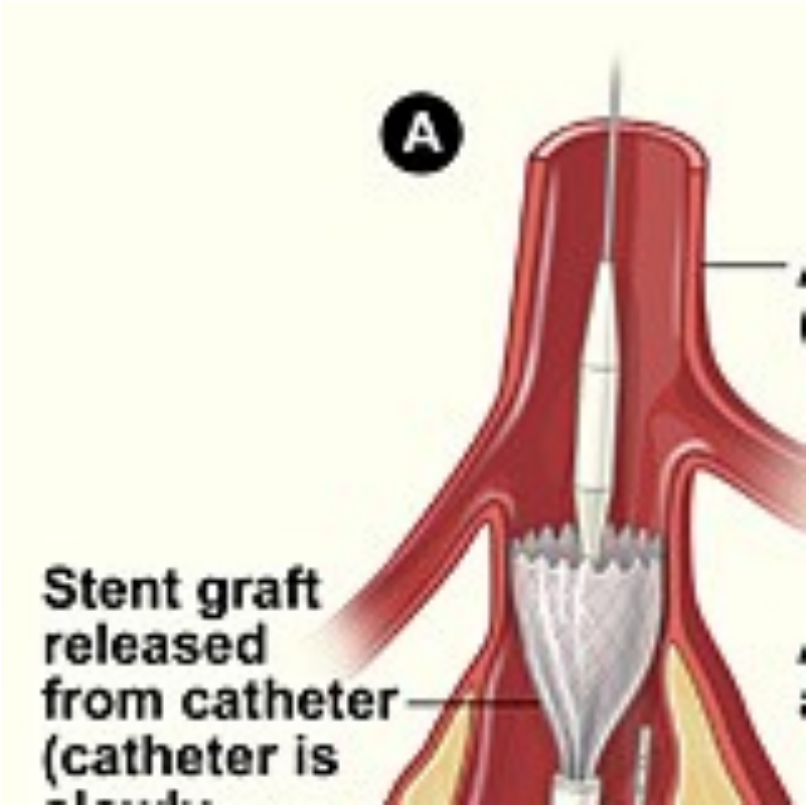


Figura 2. Aplicación del stent en una arteria aorta por (NHLBI, 2018)

Este método dio a luz al proyecto *Thrombus-VPH*, el cual es consciente que se necesita mucha investigación aún, como por ejemplo ¿cuál es el *stent* más apropiado para el vaso sanguíneo a tratar?, ya que los *stents* tienen muchos materiales y formas. Adicionalmente, habría un gran beneficio en que los médicos se capaciten para hacer este procedimiento en un medio virtual, puesto que equivocarse de este modo no es tan peligroso como lo sería en un paciente vivo.

Por otro lado, una simulación física es un conjunto de ecuaciones diferenciales que describen la evolución de variables en el tiempo (myPhysicsLab, 2009). Thrombus tiene como fin poder crear una simulación física de la operación para la aplicación del *stent* en un vaso sanguíneo y a pesar de que ya se tiene una primera versión de la simulación, se quiere mejorar, se quiere hacer algo que permita más rendimiento, interacción y hacerlo más real.

El objetivo de hacerlo más real implica que el comportamiento tenga en cuenta tantas variables como se puedan especificar a la hora de describir un vaso sanguíneo y un *stent*. (Courbebaisse, 2014)

Es en este punto que entra el presente proyecto, el cual tiene por finalidad hacer la simulación más realista, añadiendo variables a los objetos en escena que permitirían ver los objetos como cuerpos suaves, los cuales tienen la propiedad de ser deformables.

A la hora de pensar en hacer una simulación física, es necesario aplicar un modelo físico, el cual se aplicaría en dicha simulación. Un modelo físico es una conceptualización de sistemas físicos que obedecen la teoría (Rolleri, 2013) y se tiene una herramienta que aplica estos modelos, la librería *Bullet Physics*, la cual es de código abierto escrito en C++.

Como se mencionó anteriormente, este trabajo de grado haría parte de un proyecto más grande que es *Thrombus-VPH*, por lo que la intención no es empezar sin ninguna base previa, pero tampoco es terminar todo. Por ejemplo, la representación del flujo de sangre requiere que la simulación soporte dinámica de fluidos, siendo esta "la rama de la ciencia aplicada que se ocupa del movimiento de líquidos y gases" (Eckbert, 2006), lo cual es un aspecto que en este proyecto no se abarcará.

Por otro lado, se tiene planeado poner a disposición código abierto bajo una licencia zLib, la cual permite a cualquiera el uso y redistribución del software para cualquier fin, incluyendo fines comerciales, con la condición de atribuir el crédito apropiado a los autores originales de este (zLib, 2017) para que el proyecto pueda ser continuado permitiendo expandirlo con los aspectos científicos no tenidos en cuenta, como por ejemplo, agregar la dinámica de fluidos.

2. Análisis de contexto

2.1. Blood flow simulation within stented aneurysms (Thrombus-VPH, 2013)

Como parte del proyecto *Thrombus-VPH* se hizo una simulación de un vaso sanguíneo con aneurisma con un *stent* desviación de flujo dentro, con el objetivo de observar la efectividad del *stent* para desviar flujo fuera de la cavidad del aneurisma.

Este trabajo resuelve los problemas de simulación de la física del vaso sanguíneo con *stent*.

Se simuló la dinámica del flujo de sangre dentro del vaso con el *stent* usando métodos de enrejado de Boltzmann o "LBM" (Lattice Boltzmann Methods). Resulta extraña la elección de esta técnica, cuando los métodos más usados para este tipo de simulaciones son soluciones a las ecuaciones de Navier-Stokes, pero según Perumal & K. Dass, aunque LBM ocupa más espacio en memoria, es más paralelizable gracias a que su cómputo es completamente local, y permite obtener resultados más detallados. (Perumal & K. Dass, 2015)

El vaso sanguíneo y el *stent* se simularon como restricciones estáticas, es decir, modelos tridimensionales que no se mueven ni cambian de forma a lo largo de la simulación, ignorando sus propiedades de elasticidad. Esta simplificación hace que la simulación de resultados con una amplia diferencia con la realidad, por tanto, no funciona para apoyar la toma de decisiones médicas, donde la vida del paciente está en riesgo.

En la simulación se consideran casos de pacientes específicos, lo cual puede ayudar a proveer una decisión médica más precisa.

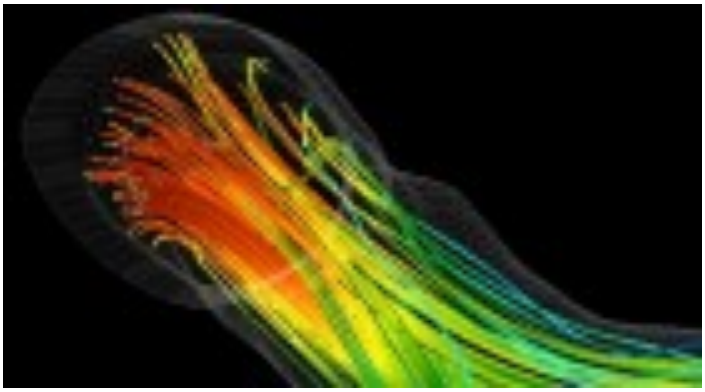


Figura 3. Demostración de *Thrombus-VPH* por (Thrombus-VPH, 2013)

2.2. Real-time surgery simulation of intracranial aneurysm clipping with patient -specific geometries and haptic (Fenz & Dirnberger, 2015)

Los autores Fenz y Dirnberger realizaron una simulación en tiempo real de una cirugía intracraneal sobre un aneurisma en pacientes con retroalimentación háptica.

Este trabajo resuelve los problemas de:

- La creciente popularidad del tratamiento endovascular de aneurisma intracraneal lleva una escasez de situaciones quirúrgicas simples para operaciones de recorte, dejando principalmente casos complejos, que llegan a ser un desafío para cirujanos experimentados.
- Hacer una implementación que permita simular el procedimiento quirúrgico en tiempo real, buscando el método que mejor se acople a las necesidades del usuario.

Con estos problemas, surge que la complejidad geométrica de los aneurismas cerebrales y sus relativos vasos delgados constaría de al menos 10.000 elementos. La primera aproximación se definió usando un número finitos de elementos en modelos de elasticidad no lineales, lo que permitiría que en largas deformaciones sea más precisas la simulación, pero con más demanda en tiempos computación. Por lo que el rendimiento en tiempo real no era suficiente.

La segunda aproximación que se usó fue usar un modelo elástico lineal, el cual, gracias a la complejidad que proporcionaba, funcionó para el prototipo final.

El resultado de este proyecto fue una simulación física (ver figura 4) de una cirugía en la que se acorta un aneurisma intracraneal con retroalimentación en tiempo real, debilitó el proyecto fue el uso de procesadores (hardware), No apto para la simulación y al usar un modelo lineal no es tan preciso.

Un problema que se identifica es que el proyecto se llevó a cabo basándose en la experiencia de los desarrolladores del proyecto, con la cual no se cuenta por los integrantes este grupo.

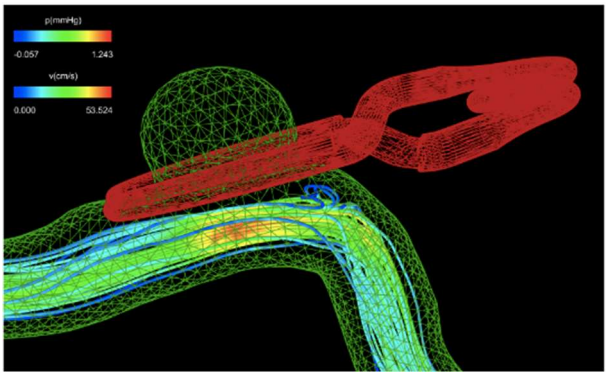


Figura 4. Acorte de aneurisma intracraneal por (Fenz & Dirnberger, 2015)

IV. ANÁLISIS DEL PROBLEMA

1. Requerimientos

Dentro del alcance del proyecto se definieron una lista de requerimientos, para el desarrollo de la prueba de concepto. En esta sección se definen los más prioritarios definidos en el documento de especificación de requerimientos (Ver especificación de requerimientos) (Ver SRS)

1.1. Requerimientos funcionales

Para la realización del proyecto se definieron los siguientes requerimientos funcionales (tabla 2) como prioritarios.

ID Completo	Requerimiento funcional	Requerimientos asociados (ID)
RF-001	El sistema debe permitir al usuario cargar el modelo en 3D del segmento de vaso sanguíneo.	6,7,8,9,22,23,24,25,32,34
RF-010	El sistema debe interactuar con el usuario.	Todos
RF-012	El sistema debe usar la física de cuerpos suaves de *Bullet Physics*.	Todos
RF-013	El sistema debe usar un motor de renderizado.	22,23,24,25,32,34,36
RF-014	El sistema debe usar la física de cuerpos rígidos de *Bullet Physics*.	Todos
RF-018	El sistema debe integrar los modelos físicos de los elementos de la escena.	Todos
RF-027	El sistema debe permitir al usuario observar el resultado de la simulación renderizada.	Todos
RF-030	El sistema debe permitir al usuario especificar el grosor del vaso sanguíneo mediante una textura.	1,6,7,8,9,22,23,24,25,32,34,35

RF-038	El sistema debe permitir renderizar el resultado de la simulación.	Todos

Tabla 2. Requerimientos funcionales

1.2. Requerimientos no funcionales

Para los requerimientos no funcionales, se tuvieron en cuenta aspectos de diseño, de rendimiento y de facilidad sobre el uso del sistema, estos son, los que se encuentran a continuación en la tabla 3.

ID	Requerimiento no funcional	Atributo de calidad (FURPS+)
RNF-009	El sistema debe poder ejecutarse en GNU Linux Ubuntu 16 LTS.	+Implementación
RNF-015	El sistema debe ser desplegado como aplicación standalone.	Soportabilidad
RNF-019	El sistema debe ser utilizable por estudiantes a menos de 2 semestres de graduarse.	Usabilidad
RNF-024	El sistema debe ser desarrollado en C++.	+Implementación
RNF-036	El sistema debe poder ejecutarse en OSX El Capitán	+Implementación

Tabla 3. Requerimientos no funcionales

2. Restricciones

Las restricciones del sistema son las siguientes:

- Las nuevas actualizaciones que se realicen al proyecto deben estar el lenguaje de programación C++.
- El programa solo puede ser compilado y ejecutado en los sistemas operativos: Linux Ubuntu 16.06 LTS y OSX Capitán.

V. DISEÑO DE LA SOLUCIÓN

1. Integración Bullet Physics con *VTK*

Para poder cumplir con los requerimientos de la prueba de concepto, se integran dos librerías:

- *Bullet Physics*: Una librería de física en tiempo real.
- *VTK*: Una librería de visualización de datos.

Ambas librerías pueden usarse desde C++, pueden compilarse e incluirse en proyectos fácilmente mediante CMake y funcionan en los sistemas operativos que se escogieron para la prueba de concepto: Ubuntu Linux 16 LTS y MacOS High Sierra.

1.1. Flujo de datos

Se inicializa *Bullet Physics* con una escena que puede tener varios objetos rígidos y suaves, luego se va actualizando en un ciclo cuadro por cuadro el estado de la simulación. Cada vez que *Bullet Physics* calcula un cuadro, el estado resultante se renderiza con *VTK*.

Pasar el estado de *Bullet Physics* a *VTK* es un problema diferente en cuerpos rígidos y en cuerpos suaves.

Paso de estado en cuerpos suaves

Implica pasar una matriz de transformación de 4x4 cada vez que *Bullet Physics* llama un *callback* registrado previamente. Dicho *callback* no se llama cuando no hay cambios en el estado de la simulación.

Paso de estado en cuerpos rígidos

Implica pasar todos los vértices del colisionador de *Bullet Physics* a *VTK* en cada iteración del ciclo que actualiza la escena.

Diferencias como estas sugieren un diseño que trate diferente a cuerpos rígidos y suaves.

2. Ingeniería inversa de cuerpos suaves de *Bullet Physics*

Debido a la carencia de documentación del modelo de cuerpos suaves de *Bullet Physics*, fue necesario buscar la información sobre este en varias fuentes:

- Tutoriales en la Wiki de *Bullet Physics*.
- Los foros oficiales de *Bullet Physics*.
- Comentarios del archivo SoftBody.h que forma parte del código fuente de *Bullet Physics*.

La información encontrada no fue suficiente, por lo que se llevaron a cabo dos actividades de ingeniería inversa para entender mejor el modelo de cuerpos suaves subyacente.

3. Análisis de la pila de llamados de funciones

Con el fin de hacer ingeniería inversa para dar con el modelo, se hizo seguimiento de la pila de llamados del método *stepSimulation* con la herramienta para *debug* lldb. Para esto se hacían saltos en las funciones que se contenían y cuando se llegaba a una función que no llamaba más funciones, se imprimía la pila de ejecución en este punto, allí se podía ver lo siguiente:

```
frame #0: 0x00000001001e872c main`btBoxShape::getAabb(btTransform const&, btVector3&, btVector3&) const
[inlined] btTransformAabb(halfExtents=0x000000010407e2f0, margin=0.0399999991, t=0x000000010407e480,
aabbMinOut=0x00007ffeefbfc2d0, aabbMaxOut=0x00007ffeefbfc2c0) at btAabbUtil2.h:184
   frame #1: 0x00000001001e872c main`btBoxShape::getAabb(this=0x000000010407e2c0,
t=0x000000010407e480, aabbMin=0x00007ffeefbfc2d0, aabbMax=0x00007ffeefbfc2c0) const at
btBoxShape.cpp:33
   frame #2: 0x0000000100182280d main`btCollisionWorld::updateSingleAabb(this=0x0000000104004ba0,
colObj=0x000000010407e470) at btCollisionWorld.cpp:151
   frame #3: 0x0000000100182c57 main`btCollisionWorld::updateAabbs(this=0x0000000104004ba0) at
btCollisionWorld.cpp:204
   frame #4: 0x0000000100182d61
main`btCollisionWorld::performDiscreteCollisionDetection(this=0x0000000104004ba0) at btCollisionWorld.cpp:
222
   frame #5: 0x00000001000b2ef8
main`btDiscreteDynamicsWorld::internalSingleStepSimulation(this=0x0000000104004ba0,
timeStep=0.0166666675) at btDiscreteDynamicsWorld.cpp:496
   frame #6: 0x0000000100349065
main`btSoftRigidDynamicsWorld::internalSingleStepSimulation(this=0x0000000104004ba0,
timeStep=0.0166666675) at btSoftRigidDynamicsWorld.cpp:94
   frame #7: 0x00000001000b2d20 main`btDiscreteDynamicsWorld::stepSimulation(this=0x0000000104004ba0,
timeStep=0.0166666675, maxSubSteps=10, fixedTimeStep=0.0166666675) at btDiscreteDynamicsWorld.cpp:456
   frame #8: 0x000000010003be50 main`Scene::UpdatePhysics(this=0x000000010410d350, deltaTime=(__rep_ =
0.016666666666666666)) at Scene.cpp:173
   frame #9: 0x000000010003bdab main`Scene::Update(this=0x000000010410d350, deltaTime=(__rep_ =
0.016666666666666666)) at Scene.cpp:168
   frame #10: 0x000000010003f065 main`vtkTimerCallback::Execute(this=0x0000000104146960,
(null)=0x0000000104069890, eventId=25, (null)=0x00000001041437f8) at Scene.cpp:58
   frame #11: 0x0000000103bdb303 libvtkCommonCore-8.1.1.dylib`vtkSubjectHelper::InvokeEvent(unsigned long,
void*, vtkObject*) + 949
   frame #12: 0x00007fff41be7739 Foundation`__NSFireTimer + 83
   frame #13: 0x00007fff3fa8a064
```

Figura 5. Reporte de la pila de ejecución mostrado por lldb

Haciendo esto en todas las funciones que no llamaban otras, se puede generar un árbol, siendo estas funciones, las hojas de este. En la siguiente figura (figura 6), se puede ver el orden de ejecución de funciones hasta un nivel más abajo que el llamado de las funciones condetinas en *stepSimulation* y se trató de presentar en forma de árbol.

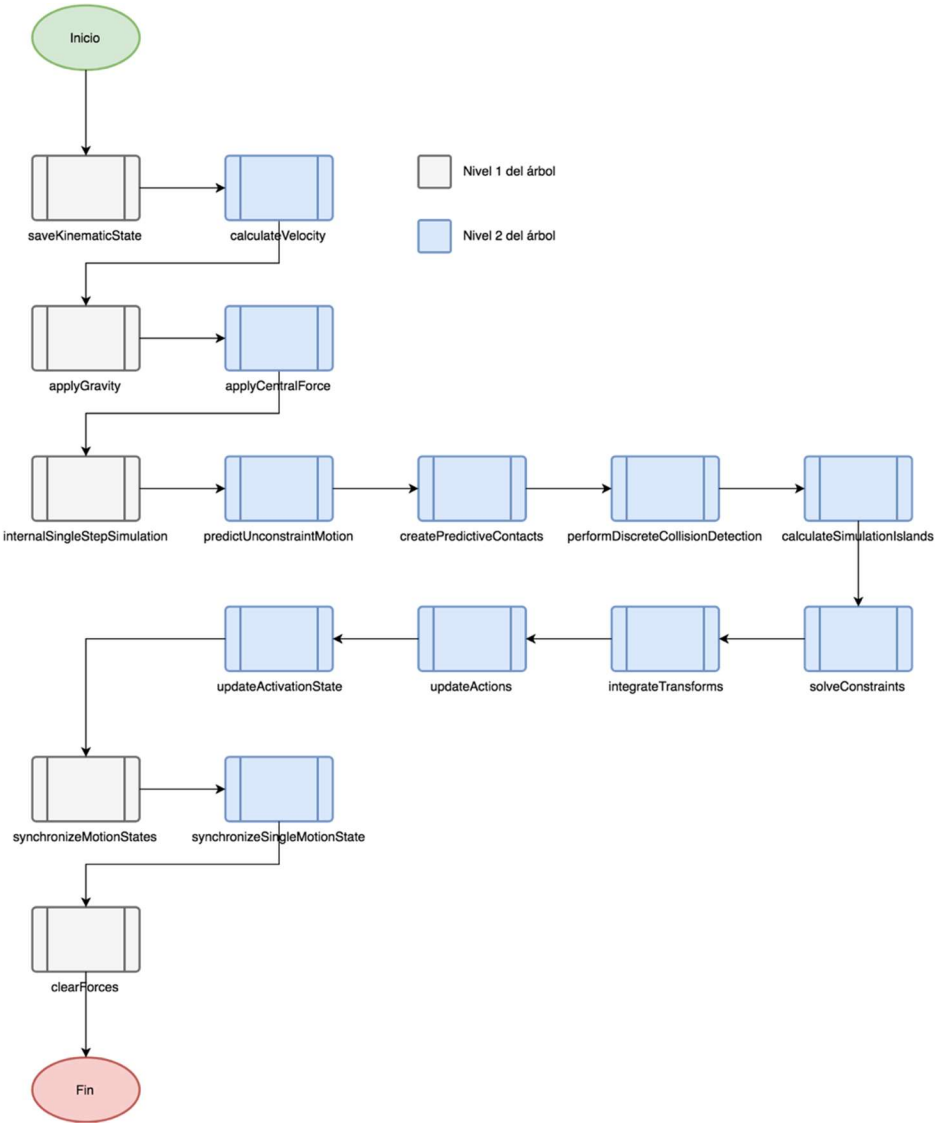


Figura 6. Diagrama de flujo de llamados de funciones del método step simulation

En el diagrama se pueden apreciar llamados de funciones. Los de color gris son los del primer nivel de árbol, lo que quiere decir que son funciones llamadas en la función *stepSimulation* de estas funciones, salen otros llamados, pero de color azul, lo cual significa que son el segundo nivel del árbol, es decir, estos métodos son llamados dentro del llamado que se ve de color gris.

4. Variación sistemática de los coeficientes de materiales y configuración

La simulación de cuerpos suaves de *Bullet Physics* tiene varios parámetros configurables por objeto, entre estos se encuentran:

4.1. kLST: Coeficiente de Rigidez Lineal

En la figura 7, se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1 en el extremo derecho. Los coeficientes kAST y kVST tienen valor 1. Su dominio de posibles valores es de [0,1].

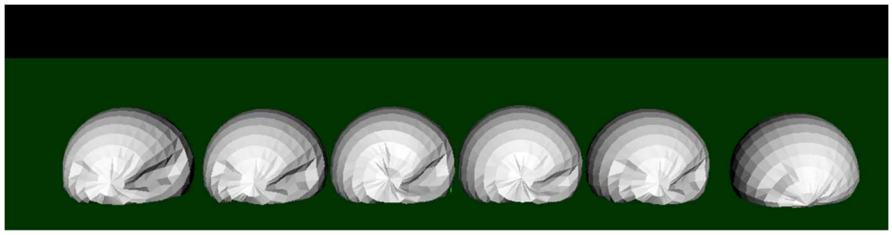


Figura 7. Cuerpos suaves con diferente coeficiente kLST.

4.2. kAST: Coeficiente de Rigidez Angular o de Área

En la figura 8 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1 en el extremo derecho. Los coeficientes kLST y kVST tienen valor 1. Su dominio de posibles valores es de [0,1].

Figura 8. Cuerpos suaves con diferente coeficiente kAST.

4.3. kVST: Coeficiente de Rigidez Volumétrica

En la figura 9 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1 en el extremo derecho. Los coeficientes kLST y kAST tienen valor 1. Su dominio de posibles valores es de [0,1].

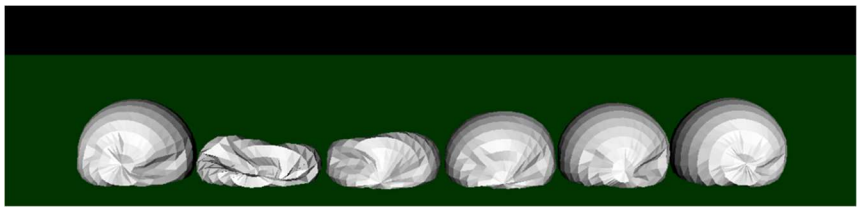


Figura 9. Cuerpos suaves con diferente coeficiente kVST.

4.4. Masa

Masa del cuerpo. Si es cero es un cuerpo estático. Su dominio de valores es de [0, ∞).

4.5. kVCF: Factor de corrección de velocidades (Baumgarte)

En la figura 10 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 10.000 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de (-∞, ∞).

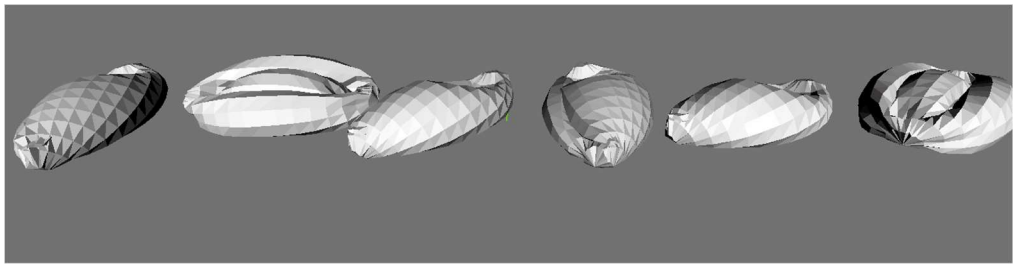


Figura 10. Cuerpos suaves con diferente coeficiente kVCF

4.6. kDP: Coeficiente de amortiguamiento

En la figura 11 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0,1].

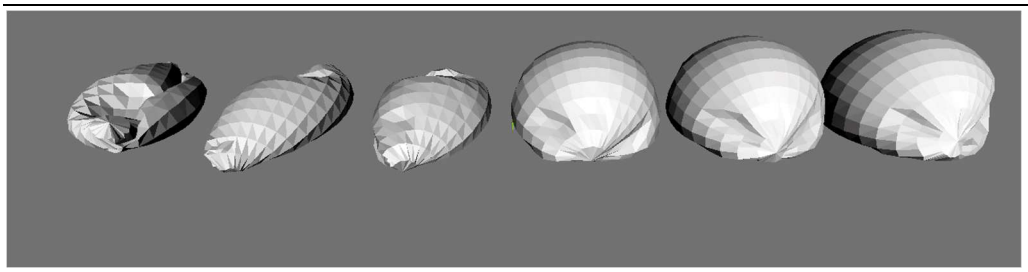


Figura 11. Cuerpos suaves con diferente coeficiente kDP

4.7. kDG: Coeficiente de resistencia al aire

En la figura 12 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1.000.000 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, ∞).

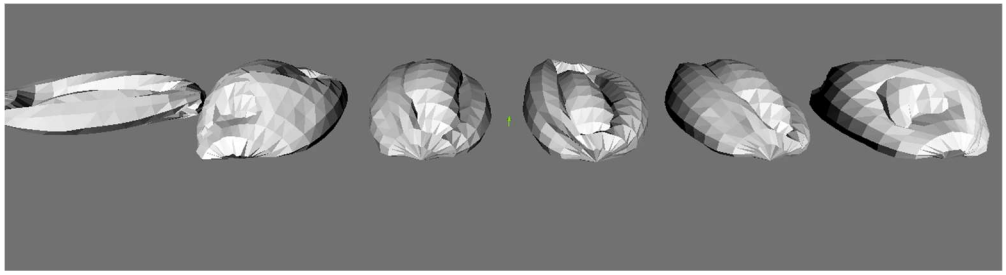


Figura 12. Cuerpos suaves con diferente coeficiente kDG

4.8. kLF: Coeficiente de sustentación

En la figura 13 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1.000.000 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, ∞).

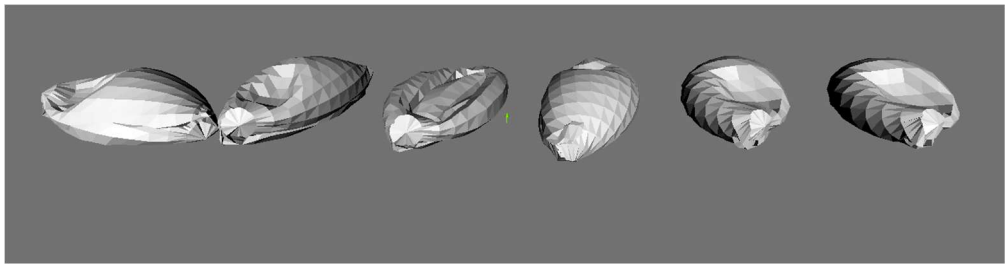


Figura 13. Cuerpos suaves con diferente coeficiente kLF

4.9. kPR: Coeficiente de presión

En la figura 14 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde -1.000.000 en el extremo izquierdo, hasta 1.000.000 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de (-∞, ∞).

Figura 14. Cuerpos suaves con diferente coeficiente kPR

4.10. kVC: Coeficiente de conversión de volumen

En la figura 15 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1.000.000 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, ∞).

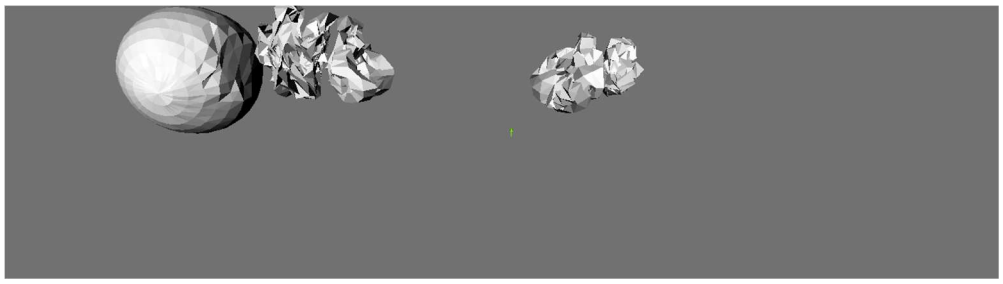


Figura 15. Cuerpos suaves con diferente coeficiente kVC

4.11. kDF: Coeficiente de fricción dinámica

En la figura 16 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1.000.000 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, 1].

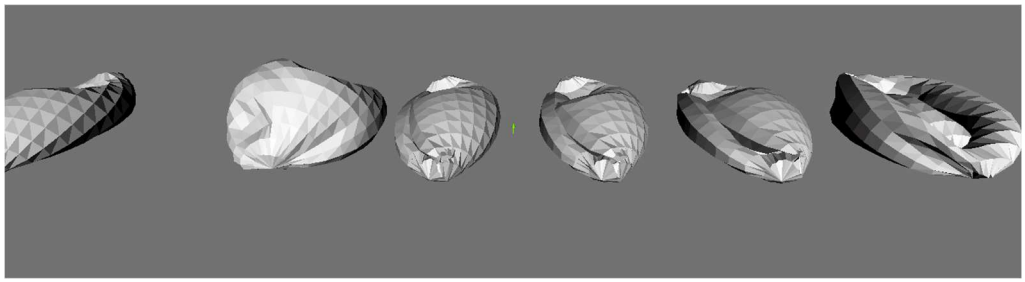


Figura 16. Cuerpos suaves con diferente coeficiente kDF

4.12. kMT: Coeficiente de coincidencia de postura

En la figura 17 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1.000.000 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, 1].

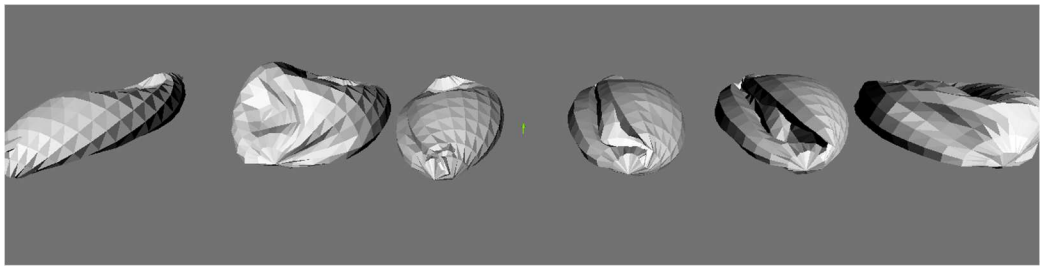


Figura 17. Cuerpos suaves con diferente coeficiente kMT

4.13. kCHR: Rigidez de contactos rígidos

En la figura 18 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, 1].

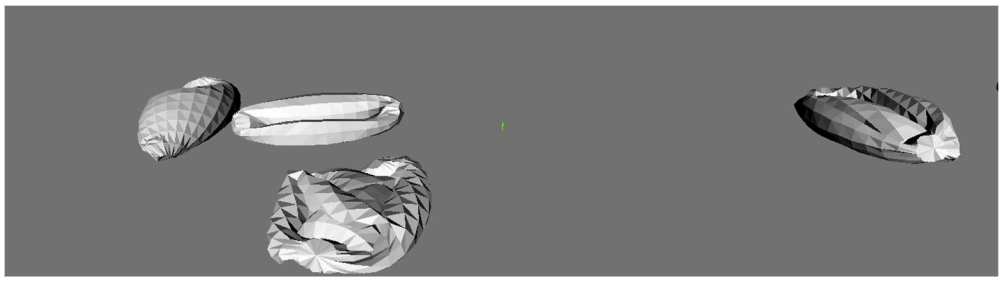


Figura 18. Cuerpos suaves con diferente coeficiente kCHR

4.14. kKHR: Rigidez de contactos cinemáticos

En la figura 19 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, 1].

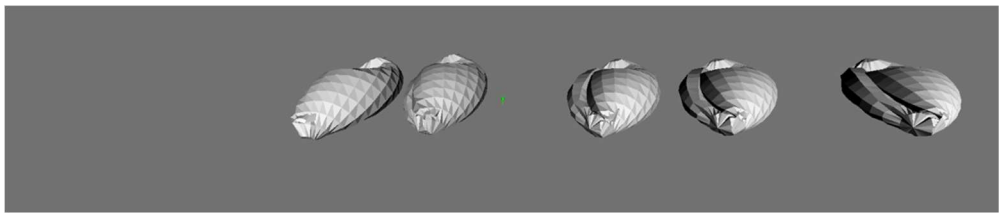


Figura 19. Cuerpos suaves con diferente coeficiente kKHR

4.15. kSHR: Rigidez de contactos suaves

En la figura 20 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, 1].

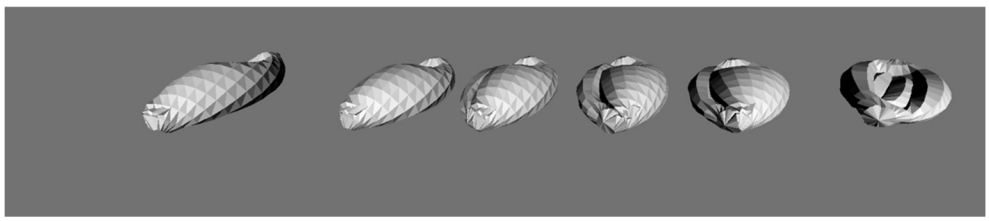


Figura 20. Cuerpos suaves con diferente coeficiente kSHR

4.16. kAHR: Rigidez de anclado

En la figura 21 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 1 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, 1].

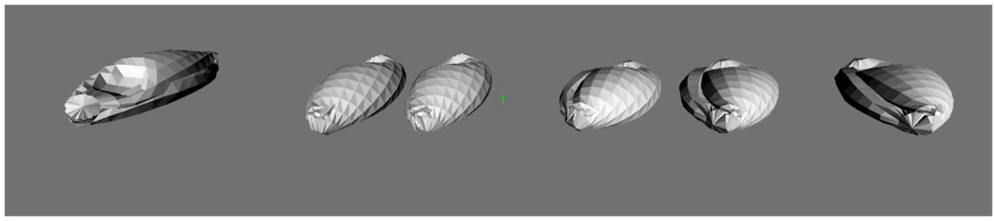


Figura 21. Cuerpos suaves con diferente coeficiente kSHR

4.17. maxvolume: Proporción máxima de volumen para la postura

En la figura 22 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde -10 en el extremo izquierdo, hasta 10 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [-10, ∞).

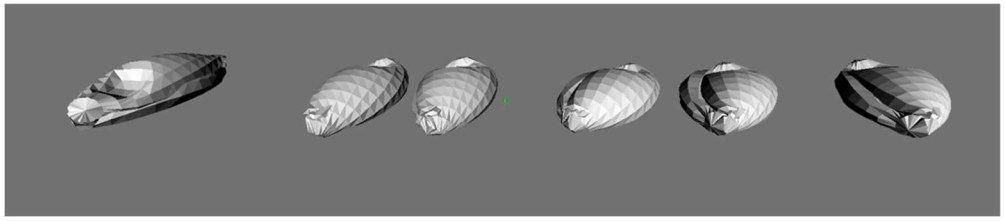


Figura 22. Cuerpos suaves con diferente coeficiente maxvolume

4.18. timescale: Escala del tiempo de simulación

En la figura 23 se muestran los efectos de variar este coeficiente. En este escenario, los valores son (de izquierda a derecha): 1, 2, 4, 8, 16, 32. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de (-∞ ∞).

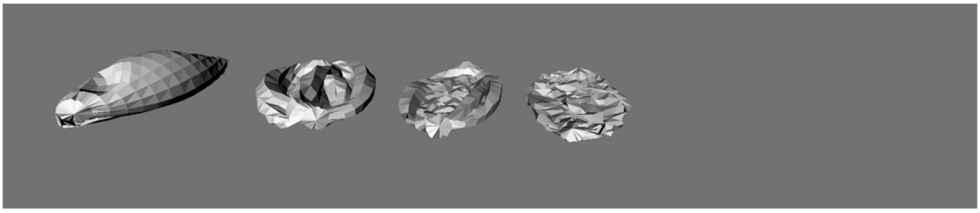


Figura 23. Cuerpos suaves con diferente timescale

4.19. viterations: Iteraciones del solucionador de velocidades

En la figura 24 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 5 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, + ∞).

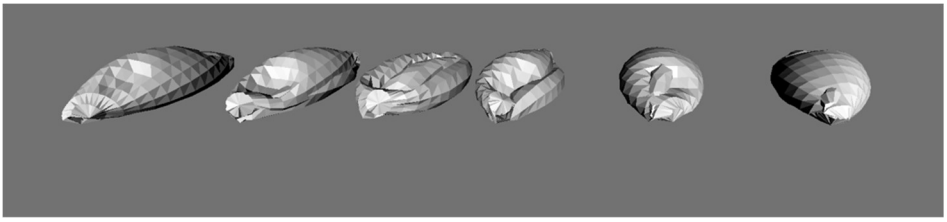


Figura 24. Cuerpos suaves con diferente viterations

4.20. piterations: Iteraciones del solucionador de posiciones

En la figura 25 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 5 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, ∞).

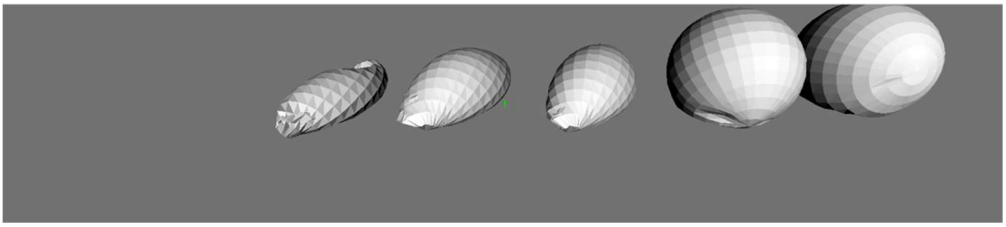


Figura 25. Cuerpos suaves con diferente piterations

4.21. diterations: Iteraciones del solucionador de deriva

En la figura 26 se muestran los efectos de variar este coeficiente. En este escenario, los valores van desde 0 en el extremo izquierdo, hasta 5 en el extremo derecho. Los demás coeficientes conservan un valor constante que es el valor por defecto. Su dominio de posibles valores es de [0, ∞).

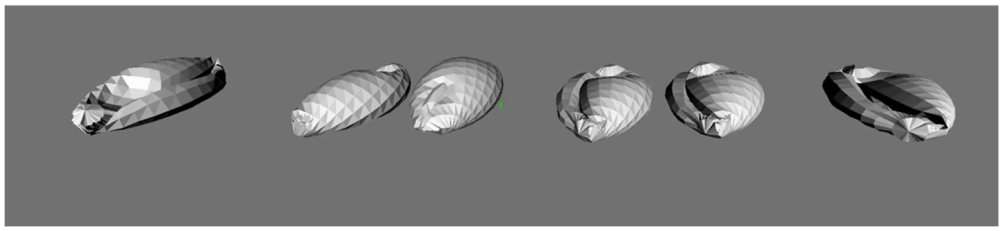


Figura 26. Cuerpos suaves con diferente diterations

5. El método de los contornos activos

Teniendo en cuenta los resultados del proceso de ingeniería inversa, se determinó que el modelo de cuerpos suaves que usa *Bullet Physics* es una versión modificada del método de contornos activos.

El objetivo principal de este método es encontrar un contorno que mejor se aproxime al perímetro de un objeto (Kass, Witkin, & Terzopoulos, 2010).Dicho contorno también es conocido como *snakes* o contornos activos, son curvas definidas dentro del dominio de una imagen que pueden moverse dentro de la influencia de fuerzas internas dentro de la curva y fuerzas externas derivadas de los datos de la imagen. Las fuerzas internas y externas son definidas para que el *snake* pueda conformarse dentro del borde de un objeto. Los *snakes* son usados comúnmente para detección de bordes, segmentación, modelado de formas y rastreo de movimiento. (Pichumani, 1997)

Para realizar este algoritmo se debe tener en cuenta los siguientes pasos:

1. Primero, el *snake* debe ser colocado cerca del objeto para hacer el perímetro.
2. Durante el proceso iterativo, el *snake* está siendo atraído hacia el contorno, por diferentes fuerzas que controlan la forma y la locación de la *snake* dentro de la imagen.

5.1. Formulación matemática:

Un *snake* puede ser definida como $x(s) = [x(s), y(s)], s \in [0,1]$, se mueve por el dominio espacial de la imagen para minimizar la energía funcional (suma de varias energías actuando sobre el contorno).

$$E = \int \Big((\alpha(s)E_{cont} + \beta(s)E_{curv}) + \gamma(s)E_{imagen} \Big) ds$$

Siendo los parámetros α, β y γ constantes de elasticidad que controlan la influencia sobre los términos de energía que puede variar con C. El término E_{cont} representa la continuidad del contorno y E_{curv} es la suavidad del contorno. (Kass, Witkin, & Terzopoulos, 2010)

La energía de la función sobre los *snakes* es la suma de las energías externas e internas, con la siguiente ecuación

$$E^*_{snake} = \int_0^1 E_{snake}\big(x(s)\big) ds$$

$$= \int_0^1 (E_{interno}\big(x(s)\big) + E_{imagen}\big(x(s)\big) + E_{usuario}(x(s))) ds$$

Los términos que definen la ecuación son los siguientes:

La energía interna ($E_{interno}$) tiene como propósito controlar las deformaciones creadas por el *snake*.

La energía de la imagen (E_{imagen}) es la variable que controla el contorno dentro de la imagen. En el caso dentro del proyecto Ane-Stent, se interpreta como el mundo visco elástico de *Bullet Physics*.

La energía del usuario ($E_{usuario}$) son las restricciones que introduce el usuario. (Xu & Prince, 2011)

5.2. Energía interna

La energía interna de las *snakes* está compuesta por la continuidad del contorno (E_{cont}) y la suavidad del contorno (E_{curv}) .

$$E_{interno} = E_{cont} + E_{curv}$$

Y se puede expandir

$$E_{interno} = \frac{1}{2}(\alpha(s)|x_s(s)|^2) + \frac{1}{2}(\beta(s)|x_{ss}(s)|^2)$$
$$= \frac{1}{2}\left(\alpha(s)\left\|\frac{d\bar{x}}{ds}(s)\right\|^2 + \beta(s)\left\|\frac{d^2\bar{x}}{ds^2}(s)\right\|^2\right)$$

Donde α y β son los términos que respectivamente controlan la tensión y la rigidez de la *snake*. $x'(s)$ y $x''(s)$ la primera y la segunda derivada de $x(s)$ con respecto a s. (Xu & Prince, 2011)

5.3. Energía de la imagen

Energía de la imagen, también conocido como la energía externa, se calcula a partir de la sumatoria de las fuerzas de la imagen (E_{imagen}) y la combinación de las fuerzas introducidas por el usuario ($E_{usuario}$).

$$E_{imagen} = w_1 I(x,y) + w_2|\nabla \mathrm{I}(x,y)|^2 + \cdots$$

Donde w_1 es el coeficiente de línea y w_2 es el coeficiente del borde. Si w_1 tiende a valores positivos, la *snake* tendera a partes oscuras de la región dentro de la imagen. Por lo contrario, si tiende a valor negativo, tiende a regiones más claras. Por otro lado, Si w_2 tiende a valores positivos, la *snake* se alineará con los bordes más definidos y si tiende a valores negativos, los evitará. (Pichumani, 1997)

5.4. Complejidad

El algoritmo tiene una complejidad de $\mathrm{O}(n)$, ya que está usando una técnica iterativa usando el método de matriz disperso. Cada iteración utiliza los pasos de resolución de ecuaciones diferenciales definidas en el método implícito de Euler con respecto a la

energía interna, la imagen y las constantes de energías externas. Las consideraciones numéricas son importantes, ya que, sin estas, el método explícito de Euler tendría una complejidad de $O(n^2)$ y cada una con un $O(n)$ para recorrer la *snake*. (Kass, Witkin, & Terzopoulos, 2010)

6. Arquitectura

En esta sección, se encuentra descrita la arquitectura de software que se planteó para el desarrollo de la prueba de concepto, así como el diseño de bajo nivel de abstracción.

Dada la naturaleza de este proyecto, el diseño arquitectural corresponde al despliegue de una aplicación *stand-alone* lo que permite la ejecución de la prueba de concepto en un ambiente controlado en el que se puede observar y evaluar la ejecución de una simulación editando los diferentes parámetros que permiten representar vasos sanguíneos utilizando *Bullet Physics* y *VTK*.

La figura 27 muestra el diseño en un alto nivel de abstracción, como se puede observar, la prueba de concepto se despliega en un solo nodo.

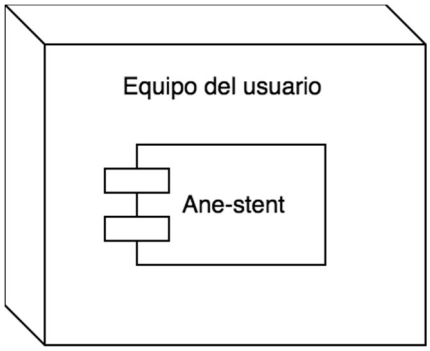


Figura 27. Diagrama de despliegue

7. Diseño detallado

A partir del planteamiento realizado en este proyecto de investigación en el que se integran las tecnologías de *Bullet Physics* y *VTK*, se tiene una prioridad alta en el planteamiento del diseño de bajo nivel, el cual se construyó para utilizar los componentes que permiten simular física y reproducirlos de forma gráfica gracias a estas librerías.

7.1. Estructura del sistema

Dado que la implementación de la prueba de concepto se planeó hacer en el lenguaje de programación C++ utilizando el paradigma orientado a objetos (según se encuentra establecido en los documentos de plan de proyecto y SRS), en la figura 28, se puede observar el diagrama de clases de la implementación de la prueba de concepto para el modelo de investigación.

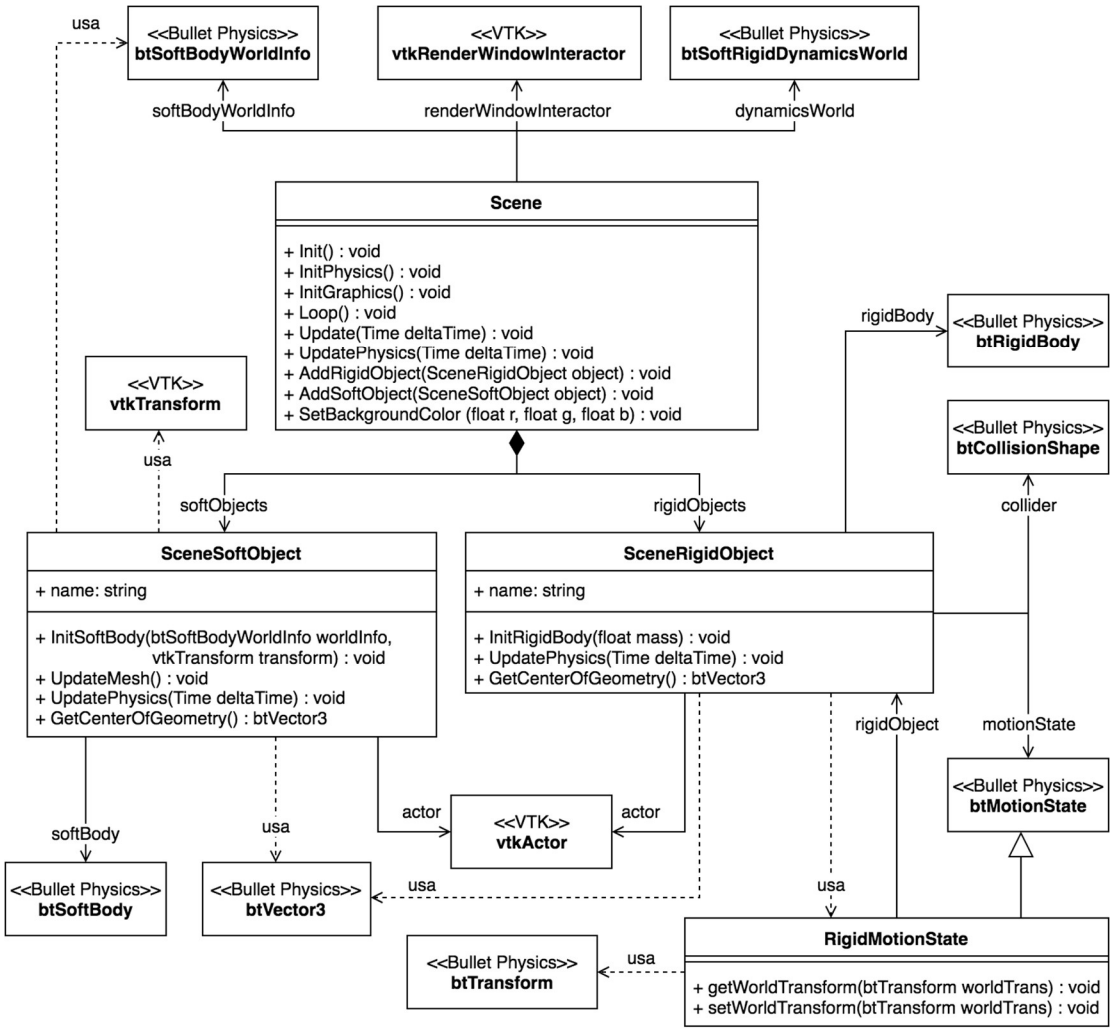


Figura 28. Diagrama de clases

Clases

A continuación, en la tabla 4, se presentan las clases que componen el código con el que se desarrolló la prueba de concepto.

Clase	Descripción

Scene	Primera clase instanciada en la ejecución de código contiene todos los objetos de la escena y el mundo. Provee la inicialización física y visual y asegura las actualizaciones de la escena.
SceneRigidObject	Representación de un objeto rígido en la escena, cubriendo las propiedades tanto gráficas como físicas.
SceneSoftObject	Representación de un objeto suave en la escena, cubriendo las propiedades tanto gráficas como físicas.
RigidMotionState	Extensión de la clase btMotionState de *Bullet Physics* que actualiza el objeto renderizado en *VTK* al recibir actualizaciones de física de *Bullet Physics*.

Tabla 4. Clases del sistema

Relaciones

A continuación, en la tabla 5, se describe la relación entre las clases con el fin de para explicar cómo estas se instancian y solicitan servicios.

Clase origen	Clase destino	Descripción
Scene	SceneRigidObject	La escena puede tener cero o muchos objetos rígidos.
	SceneSoftObject	La escena puede tener cero o muchos objetos suaves.

RigidMotionState	SceneRigidObject	RigidMotionState tiene una referencia a su SceneRigidObject para actualizar el actor de *VTK* al recibir cambios de *Bullet Physics*.

Tabla 5. Relaciones de clases del sistema

7.2. Comportamiento del sistema

En esta sección, se explica el comportamiento de cada uno de los métodos de cada una de las clases del sistema.

Métodos

A continuación, en la tabla 6, se describen los métodos que se utilizan en cada una de las clases del modelo que representa la simulación que integra *Bullet Physics* con *VTK*.

Clase	Método	Descripción
Scene	Init (): void	Invoca las funciones InitPhysics() e InitGraphics().
	InitPhysics (): void	Configura la simulación de *Bullet Physics* con ajustes como la gravedad, los algoritmos de detección de colisiones, si el mundo es viscoelástico, etc.
	InitGraphics (): void	Inicializa la ventana y la cámara, se agregan ejes en el centro del mundo para obtener un punto de referencia.

	Loop (): void	Inicializa un callback que cada cierto tiempo invoca Update().
	Update (Time deltaTime): void	Invoca el método UpdateP-hysics y renderiza los cam-bios.
	UpdatePhysics (Time deltaTime): void	Crea un paso en la simula-ción con el tiempo transcu-rrido desde la anterior e in-voca el método UpdateP-hysics de cada objeto.
	AddRigidObject (SceneObject ob-ject): void	Agrega un objeto rígido a la lista de objetos rígidos, al mundo visco-elástico y a la ventana de renderizado.
	AddSoftObject(SceneSoftObject object) : void	Agrega un objeto suave a la lista de objetos rígidos, al mundo visco-elástico y a la ventana de renderizado.
	SetBackgroundColor (float r, float g, float b) : void	Edita el color de fondo de la ventana de *VTK*.
SceneRig-idObject	+ InitRigidBody(float mass) : void	Inicializa el cuerpo rígido de *Bullet Physics* con la masa

		dada. Una vez ejecutado este método, el cuerpo rígido queda listo para ser agregado a la simulación.
	+ UpdatePhysics(Time deltaTime) : void	Virtual. Es un método que siempre se llama antes de calcular el paso de simulación de *Bullet Physics*. Por defecto no hace nada. Está hecho para que clases que extiendan SceneRigidObject puedan agregar comportamiento, como restricciones de presión, sobreescribiendo este método.
	+ GetCenterOfGeometry() : btVector3	Obtiene el centro del modelo mediante el promedio de todos los vértices del modelo.
SceneSoftObject	+ InitSoftBody(btSoftBodyWorldInfo worldInfo, vtkTransform transform) : void	Inicializa el cuerpo suave de *Bullet Physics* con la información del mundo y transformación dadas. Una vez ejecutado este método, el

		cuerpo suave está casi listo para ser agregado a la simulación.
	+ UpdateMesh() : void	Actualiza los vértices del actor de *VTK* con los vértices recibidos por *Bullet Physics*.
	+ UpdatePhysics(Time deltaTime) : void	Virtual. Es un método que siempre se llama antes de calcular el paso de simulación de *Bullet Physics*. Por defecto no hace nada. Está hecho para que clases que extiendan SceneSoftObject puedan agregar comportamiento, como restricciones de presión, sobreescribiendo este método.
	+ GetCenterOfGeometry() : btVector3	Obtiene el centro del modelo mediante el promedio de todos los vértices del modelo.
RigidMotionState	+ getWorldTransform(btTransform worldTrans) : void	Sobreescrito. Es llamado por *Bullet Physics* al inicio de la simulación. En este se

		modificia el objeto worldTrans para que quede con la transformación inicial del objeto en la simulación. Esta transformación debe incluir solo translación y rotación.
	+ setWorldTransform(btTransform worldTrans) : void	Sobreescrito. Obtiene la matriz de transformación para ese objeto y lo asigna al actor en *VTK*.

Tabla 6. Métodos de las clases y objetos del sistema

VI. DESARROLLO DE LA SOLUCIÓN

Para dar solución al problema se definieron varios hitos para cumplir con los objetivos planteados. El desarrollo de cada hito definido ayudó en el cumplimiento de los cuatro objetivos de este trabajo de grado. (Ver sección Objetivos)

1. Proceso de desarrollo

Para el desarrollo del proyecto, el grupo trabajó principalmente de modo presencial en la casa de cada uno de los integrantes. Semanalmente cambiaba la casa a la que se iba a trabajar. El anfitrión tenía como responsabilidad dar todo lo necesario para el desarrollo del proyecto, energía, internet y comida.

Adicionalmente, el anfitrión era el responsable de asignar tareas tanto para la reunión como para hacer hasta el próximo encuentro y representar el equipo.

Las reuniones se hacían todos los sábados desde las 9 de la mañana, la hora de salida variaba, pero se estipulaba hasta la 5 de la tarde. Esto era sujeto a cambios por eventos fortuitos, de modo que se buscaba la forma de reponer (por ejemplo, trabajar el domingo).

De igual forma, se tenía programado un control semanal con el director para ver avances, asignar tareas y orientar algunos temas. Después de dicha reunión, se hacía la retrospectiva del sprint.

Durante el transcurso de todo el semestre, se mantuvo un reporte de problemas en el que se relata comportamientos no deseados que retrasaban el desarrollo del proyecto, así como su solución y justificación (Para más detalles, referirse a reporte de problemas). También en este documento, se ve el reporte de los tiempos en el que se puede evidenciar lo que se hizo, quien lo hizo y cuánto tiempo tomó.

2. Metodología

Así mismo, para el proceso de desarrollo, se hizo uso de algunos factores de algunas metodologías de desarrollo, las cuales son:

2.1. AUP

De esta metodología, se usó el desarrollo por fases, las cuales son 4: comienzo, elaboración, construcción y transición. Las fases de comienzo y elaboración también siguen los lineamentos de AUP, sin embargo, estos se completaron en la etapa de planeación el semestre pasado, este semestre se trabajó en las fases construcción y transición.

Esta es la última etapa según las fases AUP que se definieron y en la que se está trabajando para la realización de este documento.

2.2. Scrum

Este semestre se hizo la fase de construcción en la cual se hizo uso principalmente scrum, de este se usó los *sprints*, los cuales eran semanales; y las tareas, las cuales se asignaban en el sprint backlog al principio de las reuniones semanales.

Al final de la reunión con el director, como se mencionó anteriormente, se hacía el sprint *retrospective*.

2.3. Extreme Programming y Mob Programming

Dentro de los *sprints,* se usaban aspectos de *Extreme Programming* y Mob *Programming* como la programación en pares y la programación en grupo respectivamente.

3. Hitos

3.1. Capacitación en *Bullet Physics*

En primera instancia, se estudió *Bullet Physics*, ya que no se tenía conocimiento del uso de la librería. Al ser subjetivo el objetivo de "aprender" se definió el hito como definir un ambiente de desarrollo y un diseño aptos para compilar *Bullet Physics* junto con *VTK*. Para verificar esto, se utilizó un ejemplo de tipo "Hello World" de *Bullet Physics* y poniendo los resultados reflejados en un objeto cargado a *VTK* en formato *VTK* y VTP.

Adicionalmente, se cambió la estructura del código para que quedara con el paradigma orientado a objetos y se detalló cada línea de *Bullet Physics* para entenderla y documentarla.

Con esto también se creó la primera versión del manual de instalación de *Bullet Physics* y *VTK*.

3.2. Cuerpos rígidos

Con una base en código y conocimiento, se cambia el objetivo para que un objeto cargado en *VTK* sea cargado también en *Bullet Physics* y se visualice cayendo y colisionando contra el suelo con las propiedades físicas de dicho objeto, ya que antes se tomaban las propiedades de una esfera.

Cargar el objeto a *Bullet Physics* también fue cuestión de especificar la forma en que va a colisionar, la cual fue un casco convexo con los puntos que se necesitaban para formar el objeto en *VTK*.

Como antes era una esfera, sólo se tenía en cuenta el eje Y para el movimiento, por lo que se agregaron el resto de los ejes. Para la rotación, hubo dificultades, ya que nadie tenía conocimientos en el manejo de cuaterniones.

Con un poco de investigación, se obtuvo la rotación para cada eje, sin embargo, al no saber el origen de la rotación, se veía raro y atravesaba paredes. Finalmente, se optó por tomar la matriz de transformación de *Bullet Physics* y asignarla a *VTK*.

3.3. Cuerpos suaves

Una vez se pudo simular correctamente un cuerpo rígido cayendo, se dedicó a hacer lo mismo, pero con cuerpos suaves.

En un principio, se ralentizó el proceso, ya que la falta de experiencia en CMake hizo que dos computadores quedaron inutilizados para la compilación del proyecto. Por este motivo, se avanzó en esta fase en los documentos necesarios, creando así una gran parte del SDD (*Software Design Description*).

Para adaptar la simulación con el fin de aceptar cuerpos suaves, fue necesario cambiar la configuración del mundo de *Bullet Physics* y darle más parámetros (como la densidad del aire).

Específicamente para los objetos, un casco convexo no era correcto, ya que para el cuerpo suave era necesario tener en cuenta las concavidades. La forma de colisión correcta era una malla de triángulos, lo que implicó una pequeña capacitación para

cargarlo así obteniendo los datos de *VTK*. Adicionalmente, estos objetos requerían más variables para ser cargado (coeficientes de elasticidad) y para ver reflejada la simulación en *VTK*, era necesario tomar cada punto y moverlo (en lugar de trasformar la figura como en los cuerpos rígidos).

3.4. Múltiples cuerpos suaves interactuando

Con el modelo y el código como está en este punto, ya puede soportar varios cuerpos, tanto rígidos como suaves, sin embargo, hay ciertas restricciones que hay que cumplir y tener en cuenta.

En primer lugar, los objetos no deben estar sobrelapados, de ser así, la simulación empezaría en un estado degenerado y la interpretación que le da *Bullet Physics* a esto es que son 2 objetos que deben rebotar con mucha fuerza, por lo que explotarían y dejarían de ser visibles por la cámara.

Por otro lado, los objetos pueden interactuar si no empiezan en un estado degenerado, sin embargo, cuando estos están cerca y colisionan, el sistema sufre una gran caída de fluidez de la simulación.

4. Solución

Se desarrolló una aplicación con el fin de probar el alcance mediante experiencia e investigación. Dicha aplicación está escrita en C++ en el estándar del 2011 y CMake como el manejador de proyectos.

La aplicación ejecuta un escenario con un piso, el cual es estático (dejando la masa del colisionador de *Bullet Physics* en 0), como se puede ver en la figura 29, y se cargan los objetos por medio de línea de comandos como se explica en el manual de uso.

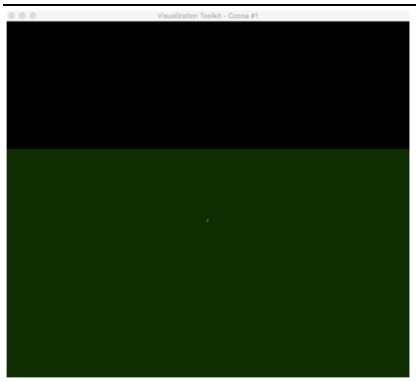


Figura 29. Estado inicial de la aplicación

Por entrada por consola, se pueden agregar objetos como se muestra en la figura 30, el cual se ve afectado por la física y caería hasta chocar con el piso como se ve en la figura 31. Después de un tiempo tras el choque con el piso, el objeto, dependiendo de los coeficientes de rigidez, tomará una forma que puede o no ser la misma con la que se inició como se ve en la figura 32.

Figura 30. Estado inicial de la simulación de un cilindro de dos radios.

Figura 31. Choque del cilindro contra el plano deformándose.

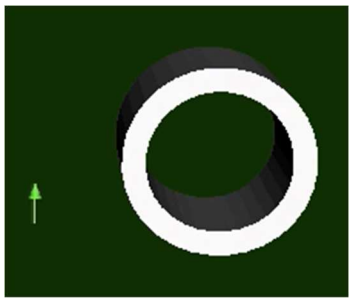


Figura 32. Recuperación de forma tras el choque.

VII. RESULTADOS

A partir del proceso que se llevó durante el semestre, se logró el cumplimiento del objetivo general definido en la propuesta de trabajo de grado. A continuación, se describen los hitos completados que favorecieron a esta propuesta.

Durante el proyecto, se definió una arquitectura basada en *Bullet Physics* a partir de la construcción de una prueba de concepto, la cual se planteó utilizando la metodología de desarrollo descrita en el plan. (Ver sección diseño de la solución)

Esta prueba de concepto puede cargar modelos con una forma que se aproxima a la de un segmento de vaso sanguíneo y a la de un *stent*. Este cargador de modelos ya ha sido probado satisfactoriamente y se puede evidenciar sus resultados en las plantillas y reporte de pruebas, en estos, se encuentran de tipo unitarias, integración, funcionales, subjetivas y de rendimiento.

Con el fin de cumplir uno de los objetivos específicos, se definió como criterio de factibilidad que la prueba de concepto tenía que pasar dos pruebas, una de rendimiento y otra subjetiva que definiera la similitud con la realidad.

Para las pruebas de rendimiento se definió que sería aprobado si la prueba de concepto en su última versión disponible el 20 de mayo de 2018 tiene tiempo promedio de duración por cuadro en segundos inferior a 0,033, para modelos de hasta 100 vértices, con 5 iteraciones. El resultado de estas a estas pruebas se evidencia en la figura 33. La figura 35 muestra cómo el número de iteraciones puede afectar el rendimiento

de la simulación, aunque la hace más precisa en términos de realismo, tiene un costo alto en tiempo de ejecución, como se muestra en la figura 35.

Por otra parte, también se definió un criterio de realismo en cual se espera que mediante una prueba subjetiva, un grupo de personas calificara la simulación opinando sobre la similitud con la realidad y como se ve en la figura 34, se observó una favorabilidad general del realismo de la simulación de cuerpos suaves. (Ver anexos plantilla y reporte de pruebas)

Como los videos proporcionados para la prueba subjetiva tenían 5 iteraciones (los cuales se puede ver que tienen un promedio menor a 0.033 de tiempo por cuadro) y con esto, se obtuvo que era similar a la realidad, se considera que el proyecto es factible.

Figura 33. Gráfico del tiempo promedio en cuerpos suaves

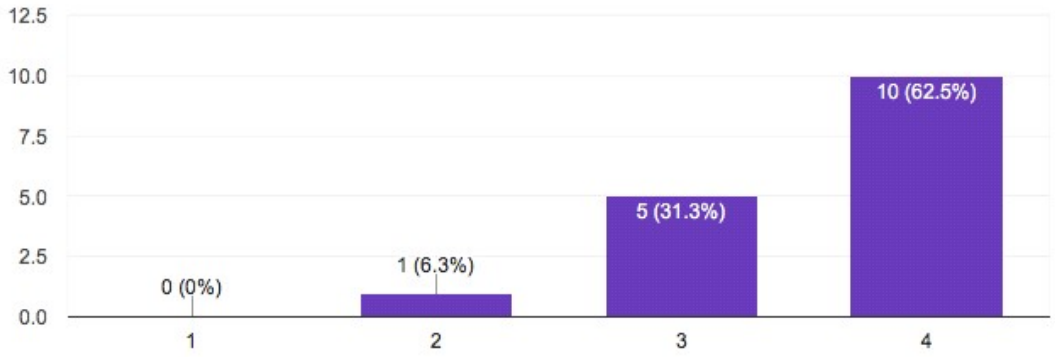


Figura 34. Percepción de realismo para usuarios en la simulación de cuerpos suaves

Figura 35. Tiempo por cuadro promedio en cuerpos suaves según número de iteraciones

Por otra parte, la plantilla de reporte de problemas fue un punto clave en el desarrollo del proyecto como se detalla en la sección desarrollo de la solución. Este fue un proceso que se realizó durante todos los *sprints* y se puede encontrar en el anexo. (Ver anexo reporte de problemas y tiempo)

A partir del hecho de que se llevaba un manejo de múltiples documentos se crearon procesos de calidad, en los cuales, se utilizaron plantillas de aseguramiento de esta, cuyos resultados se pueden ver en la plantilla de calidad (en las demás hojas de cálculo

del mismo libro), donde se muestra el calificador, la sección o documento que se calificó y su valoración del trabajo. En general, la calidad favorable, con algunos errores menores que se arreglaban en el momento de ejecutar la revisión, por ejemplo, errores de tipografía. Por otro lado, si algunos criterios de calidad no aplicaban, como algunas secciones que no requerían referencias, no se tenían en cuenta.

A partir de estos resultados, se puede afirmar que, en términos generales, hubo un satisfactorio cumplimiento de los objetivos específicos planteados en el documento de propuesta, así como su contribución al objetivo general de este trabajo de grado.

VIII. CONCLUSIONES

1. Análisis de Impacto del Proyecto

En el corto plazo, el proyecto ofrece un aporte al estado del arte, brindando una evaluación en la factibilidad del uso de *Bullet Physics* como tecnología para construir simulaciones de vasos sanguíneos y postura de *stents* en estos. A partir de lo planteado e implementado en este proceso investigativo, en el mediano plazo, próximos proyectos investigando el área de computación gráfica apoyando a la bioingeniería, tendrán una base para expandir esta investigación. Finalmente, en el largo plazo, se espera que se pueda utilizar el conjunto de herramientas construidas a partir de los diferentes proyectos para que se implemente una aplicación la cual, pueda emplearse en el entrenamiento de médicos para el tratamiento de aneurismas mediante la postura de *stent* logrando mejorar su rendimiento en este tipo de cirugías brindando el beneficio de que estos puedan corregir cualquier tipo de problema en su técnica y así obtener resultados que se reflejan en la calidad de vida de sus pacientes.

2. Conclusiones y Trabajo Futuro

2.1. Conclusiones

- La lección principal que queda de este proyecto es que, a la hora de hacer una investigación, la estimación y los planes tienen que ser hechos con mucho cuidado o con mucha flexibilidad. Esto es debido a que, por la naturaleza de este, los integrantes están adquiriendo conocimientos a medida que el proyecto avanza y muchas veces el desconocimiento hace que muchas cosas no sean tomadas en cuenta.
- El modelo de cuerpos suaves de *Bullet Physics* está bien para usar como caja negra, sin embargo, falta documentación que informe sobre su funcionamiento interno al usuario.
- Se determinó que el modelo subyacente de *Bullet Physics* probablemente es el método de contornos activos.
- Las pruebas que terminaron en excepción de *Bullet Physics*, se encontró que la librería acepta máximo 58.990 polígonos, a partir de ahí, siempre termina en una excepción.
- Por medio de las pruebas de rendimiento, se obtuvo un caso especial con los archivos de entrada, con el cual, se descubrió que la figura tiene que ser una superficie cerrada ya que, de lo contrario, *Bullet Physics* identifica esta figura como una tela y no un cuerpo con volumen.
- El proceso de ingeniería inversa realizado produjo conocimiento previamente no disponible sobre el modelo de cuerpos suaves.
- Se logró hacer una aplicación en tiempo real. La funcionalidad de esta se encuentra sustentada con pruebas de rendimiento y subjetivas. La información de estas pruebas se puede ver más detalladamente en el reporte de pruebas.

2.2. Trabajo futuro

Gracias a que este proyecto se pondrá en un repositorio público y bajo la licencia zLib, es posible el trabajo de otras personas con el mismo. Los siguientes proyectos podrían incluir la generación de *stents* con diferentes configuraciones de porosidad u otras propiedades, modelos de dinámica de fluidos, carga de objetos más grandes, mejoras en rendimiento, investigación de parámetros de la solución en pro de mejorar el realismo que se pueda presenciar en la simulación o la demostración de que, efectivamente, el modelo subyacente detrás de *Bullet Physics* para representar los cuerpos suaves, es a través del método de contornos activos lo cual también contribuirá a la documentación de esta librería.

IX. REFERENCIAS

Courbebaisse, G. (25 de Junio de 2014). *A new model for the construction of virtual fully resolved flow-diverters | Thrombus-VPH*. Obtenido de Thrombus-VPH: http://thrombus-vph.eu/a-new-model-for-the-construction-of-virtual-fully-resolved-flow-diverters

Eckbert, M. (2006). *The Dawn of Fluid Dynamics: A Discipline Between Science and Technology.* Wiley.

Fenz, W., & Dirnberger, J. (18 de Marzo de 2015). *Real-time surgery simulation of intracranial aneurysm clipping with patient-specific geometries and haptic feedback.* Obtenido de EDGE: http://edge.rit.edu/edge/P15083/public/Build%20Test%20Document/haptic%20simulator%20paper.pdf

John Hopkins Medicine. (2 de Marzo de 2011). *Flow Diversion for Aneurysms with Stents*. Obtenido de John Hopkins Medicine: https://www.hopkinsmedicine.org/neurology_neurosurgery/centers_clinics/aneurysm/treatment/flow-diversion.html

Kass, M., Witkin, A., & Terzopoulos, D. (25 de Mayo de 2010). *Snakes: Active Contour Models.* Obtenido de Laboratorio de procesado de imagen | Image Processing Lab: https://www.lpi.tel.uva.es/muitic/pim/docus/Snakes.pdf

myPhysicsLab. (15 de Julio de 2009). *Physics Simulations*. Obtenido de myPhysicsLab: https://www.myphysicslab.com/

NHLBI. (29 de Marzo de 2018). *Aneurysm*. Obtenido de National Heart, Lung, and Blood Institute (NHLBI): https://www.nhlbi.nih.gov/health-topics/aneurysm

Perumal, D., & K. Dass, A. (Diciembre de 2015). *A Review on the development of lattice Boltzmann computation of macro fluid flows and heat transfer.* Obtenido de Science Direct: http://www.sciencedirect.com/science/article/pii/S1110016815001362

Pichumani, R. (7 de Julio de 1997). *Snakes: an active model*. Obtenido de Snakes: an active model: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/RAMANI1/node31.html

Rolleri, J. L. (20 de Mayo de 2013). *¿Qué son los modelos físicos?* Obtenido de Scielo: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2007-25382013000100007

Science Daily. (9 de Octubre de 2006). *Blood Vessel*. Obtenido de Science Daily: https://www.sciencedaily.com/terms/blood_vessel.htm

Thrombus-VPH. (20 de Diciembre de 2013). *Thrombus-VPH*. Obtenido de Blood flow simulation within stented aneurysms: http://thrombus-vph.eu/blood-flow-simulation-within-stented-aneurysms

Wegman, E., & Wright, I. (Junio de 1983). *Splines in Statistics on JSTOR.* Obtenido de JSTOR: http://www.jstor.org/stable/2288640

Xu, C., & Prince, J. (5 de Marzo de 2011). *Gradient Vector Flow: A New External Force For Snakes - Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference.* Obtenido de IACL: http://iacl.ece.jhu.edu/pubs/p087c-ieee.pdf

zLib. (15 de Enero de 2017). *zLib License*. Obtenido de zLib: https://zlib.net/zlib_license.html

X. ANEXOS

1. Arquitectura
2. Descripción de pruebas del sistema
3. Diagrama de clases
4. Diagrama de flujo
5. Manual de configuración de ambiente de desarrollo
6. Manual de Usuario
7. Plan de pruebas
8. Plantilla Calidad Documentos
9. Plan de proyecto
10. Plantilla de pruebas del Sistema
11. PostMortem
12. Reporte de calidad
13. Reporte de problemas y tiempo
14. Reporte de pruebas
15. SDD
16. SRS