Descripción de diseño del software

Para el trabajo de grado Ane-Stent

Stephanie Domínguez Andrade
s.dominguez@javeriana.edu.co
Juan Sebastián Espinosa Torres
espinosa_j@javeriana.edu.co
Jose Antonio Quintero Gómez
j.quinterog@javeriana.edu.co
David Alonso Villamizar Lizcano
villamizar.david@javeriana.edu.co

1. Historial de cambios

Versión	Descripción	Fecha	Responsable
1.0	Versión inicial del documento	03/03/2018	Jose Antonio Quintero Juan Sebastián Espinosa
1.1	Agregadas secciones de introducción, comportamiento del sistema e interfaz de usuario, se expandieron secciones referentes al diseño detallado y la arquitectura del sistema.	24/03/2018	Juan Sebastián Espinosa
1.2	Se completaron secciones de interfaz de usuario y comportamiento del sistema	07/04/2018	Juan Sebastián Espinosa
1.3	Versión final del documento	20/05/2018	David Alonso Villamizar

2. Prefacio

Este documento presenta la descripción de diseño de software de la prueba de concepto del proyecto de investigación del equipo Ane-stent. Aquí se muestra el planteamiento de la arquitectura de software, así como el diseño detallado en el que se explican los componentes del sistema y de igual manera su comportamiento.

3. Tabla de contenidos

4. Lista de figuras

5. Lista de tablas

6. Arquitectura

6.1. Recursos y versiones

Dado que el objetivo general del proyecto consiste en evaluar la factibilidad técnica de una simulación desplegado en una pequeña aplicación *standalone* que renderice en VTK la información calculada en Bullet Physics, la arquitectura de software que se planteó sólo requería de una unidad de despliegue en un solo componente como se ve representado en la figura 1 diagrama de despliegue.

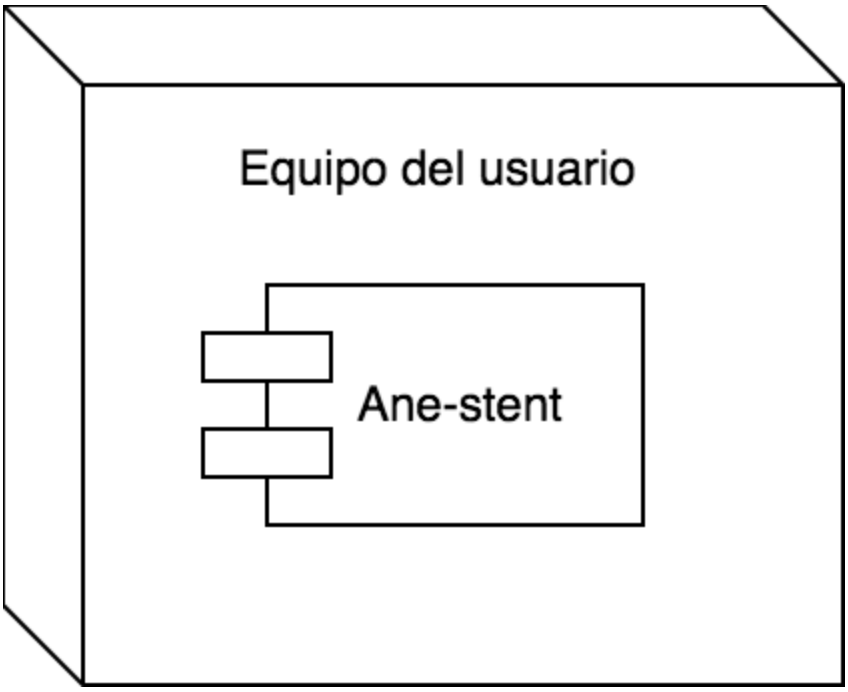

Figura 1. Diagrama de despliegue

7. Diseño detallado

En esta implementación del uso de las librerías *Bullet Physics* y VTK en conjunto, es muy importante recalcar que el diseño de bajo nivel representa a los componentes de mayor enfoque del proyecto, ya que esto es lo que permite el desarrollo de un ejecutable que permitirá evaluar los tiempos, el consumo de recursos y la calidad de los resultados de la simulación.

Esta sección evidencia el diseño que se construyó y se tomó en cuenta para la implementación del prototipo, mostrando las clases que se plantearon para representar objetos que representan a los diferentes cuerpos que intervienen en la simulación y los elementos necesarios para integrar las herramientas gráficas de VTK con el modelo físico de *Bullet Physics*.

7.1. Estructura del sistema

Dado que la implementación del prototipo se planeó hacer en el lenguaje de programación C++ utilizando el paradigma orientado a objetos (Ver plan de proyecto) (Ver documento SRS), a continuación, en la figura 2, se puede observar el diagrama de clases de la implementación de la prueba de concepto para el modelo de investigación.

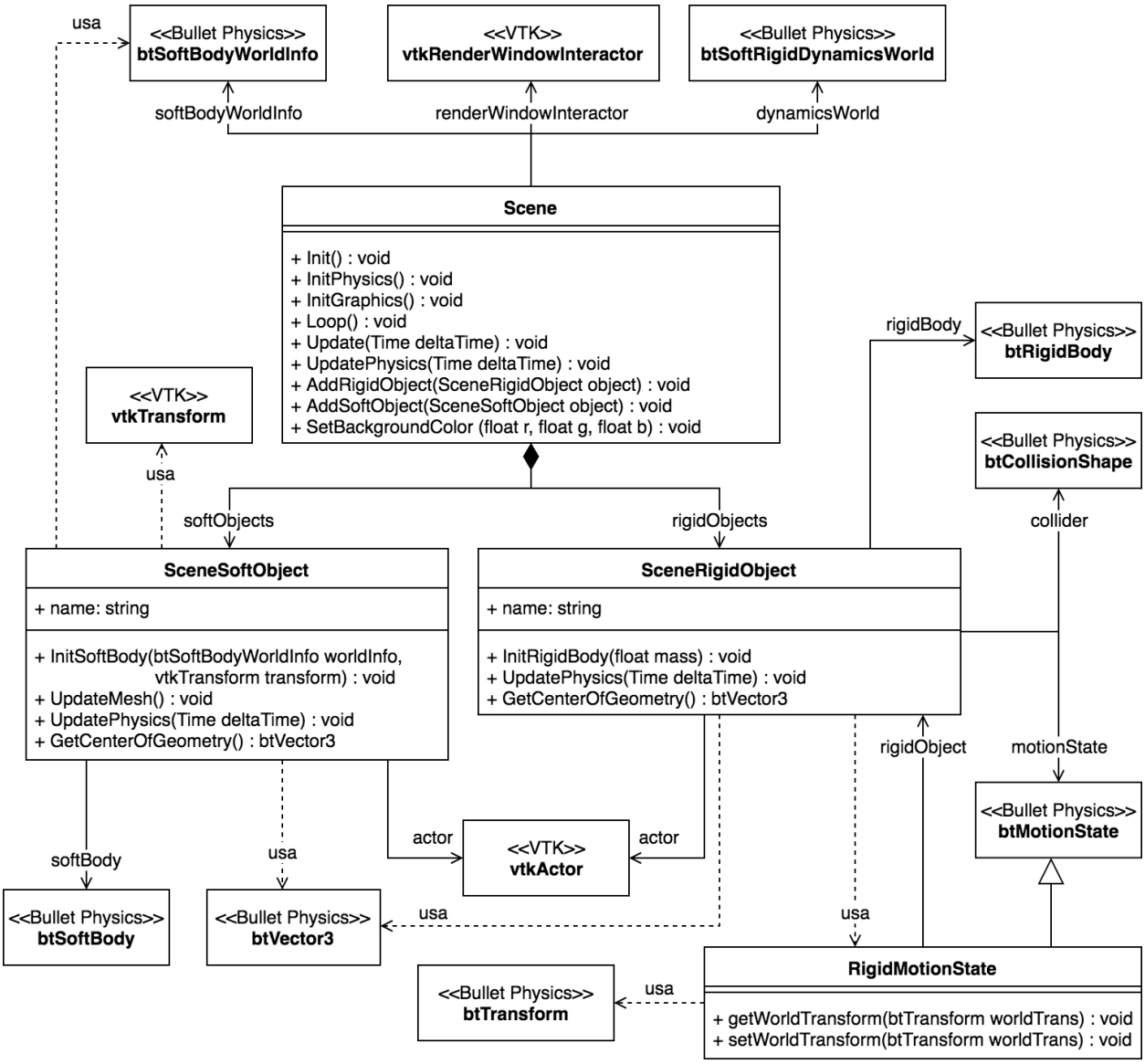

Figura 2. Diagrama de clases

7.1.1. Clases

A continuación, en la tabla 1, 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 1. Clases del sistema

7.1.2. Relaciones

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 2. 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.

7.2.1. Métodos

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 visco-elástico, 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 UpdatePhysics y renderiza los cambios.
	UpdatePhysics(Time deltaTime) : void	Crea un paso en la simulación con el tiempo transcurrido desde la anterior e invoca el método UpdatePhysics de cada objeto.
	AddRigidObject(SceneObject object) : 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.
SceneRigidObject	+ 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 3. Métodos de las clases y objetos del sistema

7.2.2. Interfaz de usuario

Teniendo en cuenta que el principal entregable de este proyecto consistirá en el código fuente y los resultados de la evaluación de la prueba de concepto, y que, los usuarios finales son estudiantes e investigadores (Ver sección Características de usuario de SRS), se optó por manejar la interfaz de usuario en la forma del visualizador de VTK e interfaz línea de comandos.

- Visualizador de VTK: Con el fin de mostrar los datos con los que se opera, VTK provee una clase (vtkRendererWindow) cuyas subclases proporcionan una interfaz unificada que se adaptan a diferentes sistemas operativos y kits de herramientas de interfaz gráfica. (Kitware, 2015)

En la figura 3, se puede observar un modelo cargado y representado en VTK:

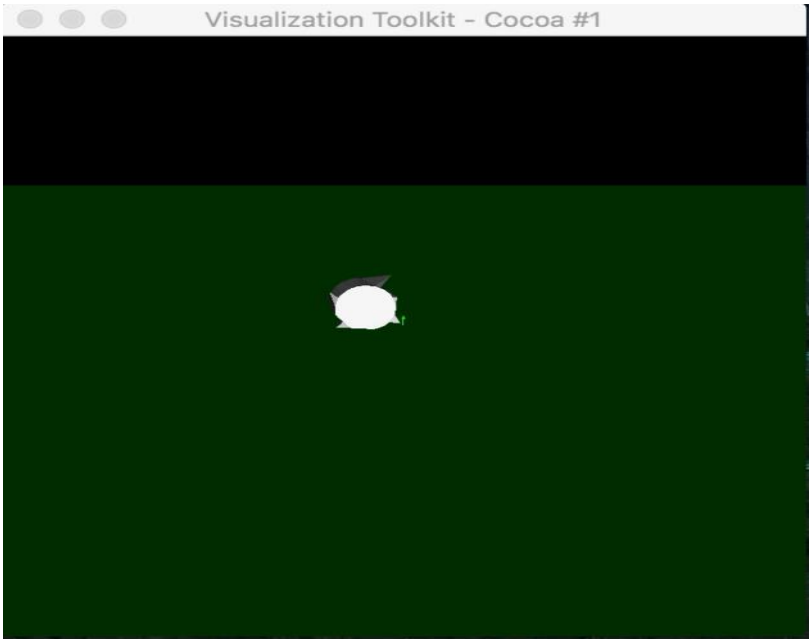

Figura 3. Modelo cargado y representado en VTK

- Línea de comandos: Medio de comunicación directa entre usuario y software, no posee interfaz gráfica por lo que esta comunicación se debe dar por medio de texto, es decir, por el uso de comandos para el usuario y por medio de retroalimentación textual desde el software. (Microsoft Corporation, 2009)

Para más información acerca de los comandos y posibles formas de interacción, consultar manual de usuario.

8. Referencias

Kitware. (2015, 03 25). *VTK Interaction and GUI Support*. Retrieved from VTK - The Visualization Toolkit: https://www.vtk.org/features-interaction-and-gui-support/

Microsoft Corporation. (2009, 11 09). *Command shell overview*. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb490954(v=technet.10)