Visualización de Cuencas Hidrocarburíferas con Realidad Virtual

Federico Marino[1], Horacio Abbate[1], Ricardo A. Veiga[1], Ema Aveleyra[1], Silvia Barredo[1], Patricia Larocca[1], María Alejandra Arecco[1] y Gabriela Savioli[1]

[1] Universidad de Buenos Aires, Facultad de Ingeniería, Buenos Aires, Argentina
{fmarino, habbate, rveiga, eaveley, sbarrero, plarocc, marecco, gsavioli}@fi.uba.ar

Abstract. Se presenta una aplicación web de realidad virtual, orientada a la industria de Gas y Petróleo, capaz de construir una representación interactiva y tridimensional de una cuenca sedimentaria con potencial hidrocarburífero. Esta herramienta didáctica permite navegar un modelo 3D utilizando un casco de realidad virtual. Las vistas 3D se generan mediante la combinación de las técnicas de rendering tradicional con la técnica de Raymarching. Las superficies se describen a partir de funciones SDF (Signed Distance Functions) y son computadas mediante un píxel shader en la GPU. Esto permite alterar en tiempo real, la forma de las superficies sin utilizar triángulos. El usuario puede controlar con sus manos, un conjunto de operadores que permiten sustraer partes del volumen. La visualización del terreno permite analizar sus propiedades en distintas direcciones, facilitando la comprensión de los componentes de la cuenca y la identificación de reservorios de hidrocarburos.

Keywords: Raymarching, Realidad Virtual, Rendering volumétrico, Industria de Gas y Petróleo, Visualización científica.

1 Introducción

En la industria de gas y petróleo existen numerosos usos de sistemas de realidad virtual para aumentar la eficiencia y reducir los riesgos de la exploración y de las operaciones (Sony, Samsung, Qualcomm, entre otros). Si bien estas técnicas se aplicaron extensivamente en las distintas empresas para entrenamiento y capacitación, hoy se ha extendido a otras áreas como operación y mantenimiento de instalaciones de superficie, y también para exploración y producción.

Como menciona Jampeisov [1], una persona recibe el 80% de la información sobre el mundo a través de la visión, y el uso de visualizaciones 3D ayuda a mejorar la eficiencia en el análisis de vastos volúmenes de datos

En la Facultad de Ingeniería de la Universidad de Buenos Aires (FIUBA) hay un interés creciente en la utilización de diferentes tecnologías informatizadas para la formación de ingenieros [2]. Se considera que la incorporación de TIC en las metodologías de enseñanza ofrece una alternativa para mejorar la comprensión y aplicación de modelos científico-tecnológicos [3], [4], [5]. En particular, con el uso de la Realidad

Virtual (RV) se busca: a) fomentar el aprendizaje interactivo, con el cual los estudiantes puedan involucrarse de manera más activa en la construcción de los conceptos que están analizando mediante la exploración y el análisis de los datos; b) facilitar la contextualización de las experiencias de aprendizaje; c) enriquecer el conocimiento científico a través de la visualización de diversos escenarios; d) aumentar la motivación, interacción y colaboración entre pares. Experiencias desarrolladas en ciencias básicas muestran que la utilización tanto de la Realidad Virtual como Aumentada permiten una nueva manera de interactuar con los modelos físicos (ya sea en su visualización o control) de forma más simple y en condiciones de accesibilidad más flexibles en términos de tiempo y espacio [6], [7], [8].

1.1 El problema a resolver

El proyecto aquí descrito surge de la necesidad de construir una representación integral de una cuenca hidrocarburífera y sus sistemas petroleros, que sea didáctica, interactiva y tridimensional. Este proyecto fue financiado por el proyecto PIDAE 2018 (1-166) denominado “Realidad Virtual y Aumentada en Cuencas Hidrocarburíferas Digitales”.

La figura 1 representa esquemáticamente una imagen sísmica de una cuenca petrolera con sus estructuras interpretadas. Éste es el tipo de representación que se utiliza habitualmente y consiste en una secuencia de cortes bidimensionales con etiquetas que indican cuáles son los diferentes tipos de componentes geológicos; entendiéndose por tales la cuenca y su geometría, el relleno representado a través de estratos afectados por fallas, entre otros.

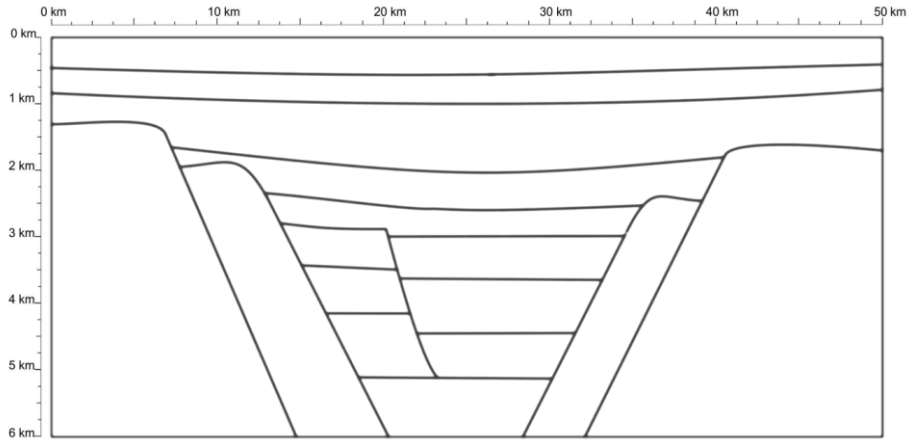

Fig. 1. Esquema inicial de un corte geológico simulado para el modelo de RV.

Los modelos de cuencas sedimentarias se elaboran a partir de datos representados en diferentes formatos. Los archivos LAS (Log ASCII Standard) permiten tener un continuo de la información en la vertical de un pozo, fácilmente correlacionable con la información directa de afloramiento.

Hasta ahora el estudiante debía analizar un esquema bidimensional como el de la figura 1, para hacerse una imagen mental de la forma tridimensional de cada uno de los estratos como medios porosos y posiblemente permeables. El modelo de RV le permitirá al estudiante no solamente observar imágenes bidimensionales, sino tener una percepción tridimensional de la cuenca y explorar su interior en una forma dinámica e intuitiva. De esa manera, podrá analizar la litología, las texturas y las estructuras sedimentarias, como así también las probables propiedades petrofísicas asociadas a ese contexto.

1.2 Dispositivos de realidad virtual

Los dispositivos de realidad virtual resultan cada vez más accesibles al público en general. Los mismos consisten en una pantalla de alta resolución más un sistema de lentes que proporcionan una imagen independiente para cada ojo (estereoscopía). Además, cuentan con un conjunto de sensores para medir la actitud y posición de la cabeza del usuario en el espacio tridimensional. Actualmente existe una variedad de soluciones tecnológicas para medir dichos parámetros; algunos sistemas utilizan cámaras externas con iluminación infrarroja para rastrear patrones específicos en la superficie del casco, combinados con giróscopos, acelerómetros y magnetómetros que permiten determinar la orientación y posición del usuario. Además, muchos de ellos, poseen controles de mano inalámbricos (*joysticks*) cuya posición y actitud también son sensados.

Existen dos grandes grupos de dispositivos: a) aquellos que requieren de una computadora (conectada vía USB, HDMI o WIFI) para la generación de la imagen y el procesamiento de los datos de sus sensores; y b) aquellos que funcionan en forma autónoma porque cuentan con su propia CPU y GPU. Estos últimos, entre los que se encuentra el Oculus Quest, aún poseen una capacidad de procesamiento limitada, equivalente a la de un dispositivo móvil.

En este proyecto se decidió utilizar un dispositivo Oculus Quest conectado a una PC ya que los requerimientos de procesamiento gráfico son muy elevados.

1.3 Estándares WebGL, WebVR y WebXR

El estándar WebGL (Web Graphics Library) es una especificación estándar de una API implementada en Javascript para la renderización de gráficos 3D dentro de cualquier navegador web. No precisa la instalación de complementos y funciona en cualquier plataforma que soporte OpenGL 2.0 y OpenGL 2.0 ES. WebGL permite la utilización de la aceleración por hardware de la GPU para la renderización de imágenes en un elemento *canvas* del documento HTML.

El estándar WebVR, creado en 2014 por Vladimir Vukićević de Mozilla, es una API experimental en Javascript que proporciona acceso a los dispositivos de realidad virtual como HTC Vive, Oculus Rift, Oculus Quest o Google Cardboard desde un navegador. Este estándar tiene como objetivo detectar los dispositivos VR disponibles, obtener sus características, obtener la posición y orientación del dispositivo en el espacio 3D y mostrar imágenes en el dispositivo con la frecuencia de refresco adecuada. La última versión 1.1, lanzada en 2017 depende de versiones especiales de los navegadores como

Firefox Nightly o una versión personalizada de Google Chrome. El estándar nunca llegó a implementarse por completo en todos los navegadores hasta ser sustituido por WebXR.

El estándar WebXR [9] (que reemplaza a WebVR), reúne en una misma API la capacidad de implementar aplicaciones de Realidad Virtual y Realidad Aumentada (XR = AR + VR) ya que los dispositivos para ambas aplicaciones tienen necesidades en común como ser medir la posición y orientación del dispositivo en el espacio y generar una imagen acorde al punto de vista actual.

En el caso de aplicaciones de realidad aumentada, el estándar contempla la utilización tanto en dispositivos móviles como celulares o tablets basadas en Android o iOS como en dispositivos específicos como el Microsoft Hololens.

Aún en mayo de 2021, WebXR continúa siendo un "working draft" y no está disponible en todos los navegadores.

1.4 La plataforma seleccionada

Para este trabajo se propuso desarrollar una aplicación web basada en Javascript, HTML, CSS, WebGL y WebXR junto con las bibliotecas de código abierto, Three.js, React.js y Bootstrap.

Three.js es una biblioteca Javascript, construida sobre WebGL que facilita el desarrollo de aplicaciones 3D, proveyendo componentes básicos para la gestión de mallas 3D, jerarquías de objetos, cámaras, fuentes de iluminación, materiales, shaders y renderizado.

React.js es una biblioteca Javascript para construir interfaces de usuario o componentes UI.

Bootstrap es un framework CSS para desarrollar sitios web responsivos.

El uso de la plataforma web permite independizarse del sistema operativo y de características específicas del hardware sobre el cual puede ejecutarse. Además, no requiere instalación y puede ser utilizada eventualmente en dispositivos móviles.

Más allá de que el objetivo central sea el uso de la aplicación en realidad virtual, es factible que en futuras versiones la aplicación pueda ser utilizada por un docente directamente en clases por medio de un teclado, un mouse y un monitor o proyector o publicada en la web para ser accedida por el público en general.

1.5 La solución propuesta

Se diseñó una aplicación donde el usuario puede navegar libremente alrededor del volumen 3D que representa la cuenca. Mediante el uso de controles de mano es posible posicionar ciertos objetos 3D denominados "operadores booleanos" (esfera, plano o cilindro) que permiten recortar el volumen de la cuenca en forma interactiva, sustrayendo así porciones de la misma y permitiendo al usuario visualizar su interior, como se observa en la figura 2.

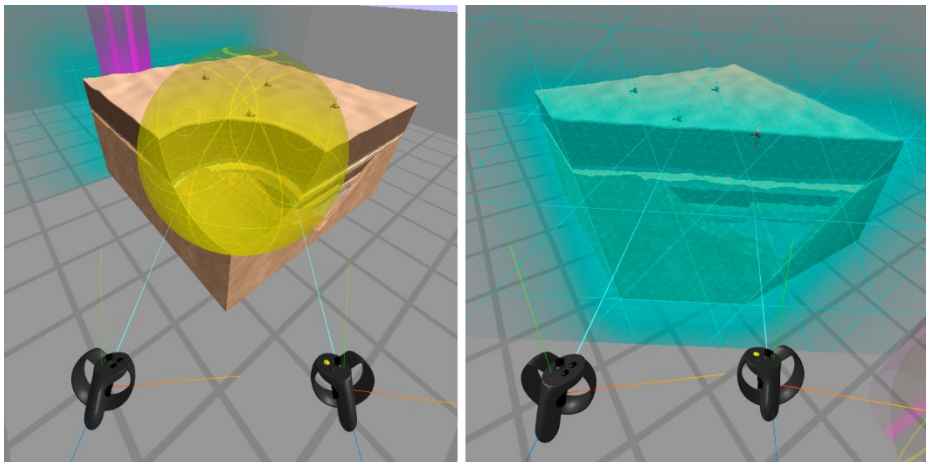

Fig. 2. Operadores booleanos siendo manipulados: izq. operador esfera y der. operador plano.

2 El motor de visualización

2.1 Representación de superficies

El problema de construir una representación gráfica de la cuenca se puede dividir en dos partes, como se observa en la figura 3: a) computar las coordenadas de todos los puntos que conforman las superficies resultantes de sustraer los operadores booleanos del cubo que representa la porción del terreno, b) computar el color de cada pixel de las superficies proyectadas en la pantalla. Para resolver el punto a) se debe considerar alguna forma de modelar dichas superficies dinámicamente, ya que el usuario manipula la ubicación y las escalas de los operadores booleanos en tiempo real.

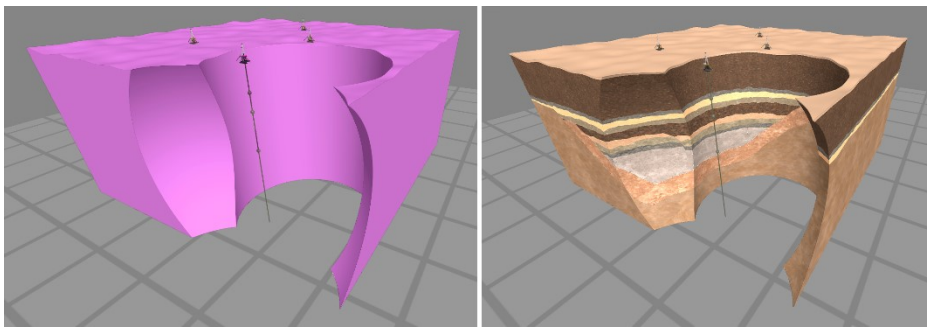

Fig. 3. Superficie resultante de sustraer el operador booleano esférico y cilíndrico (izq.) Superficie coloreada según información de la estructura interna de la cuenca y sus estratos (der.).

En computación gráfica existen dos formas de representar superficies, que son la forma paramétrica y la implícita, como se indica en [10]. La API WebGL permite renderizar escenas 3D a través de una serie de mecanismos de software y hardware que

se pueden resumir en lo que se conoce como un pipeline gráfico, donde una secuencia de etapas recibe información geométrica como entrada (vértices, triángulos, normales, etc.) y genera como resultado una imagen (mapa de bits) en el dispositivo de salida. Para poder visualizar las superficies usando el pipeline gráfico tradicional es necesario obtener una malla de triángulos a partir del modelo de las superficies.

La técnica de Marching Cubes [11], es un algoritmo cuyo objetivo es obtener una malla poligonal de una isosuperficie en un campo escalar discreto tridimensional (voxels). Esta técnica habitualmente es utilizada para la visualización de imágenes médicas tomadas mediante escáneres de resonancia magnética en donde se representa una función de densidad en escala de valores entre 0 y 1. En este caso la función de densidad podría ser simplemente computada a partir de los operadores booleanos y su posición y orientación relativos al volumen del terreno, asignando una densidad 0 para un voxel vacío y 1 para un voxel ocupado. Luego mediante el algoritmo de Marching Cubes se podría obtener la malla de triángulos que representa la superficie de la figura 3. Una de las desventajas de este enfoque es el tiempo de procesamiento requerido para obtener los triángulos, ya que este proceso debería ejecutarse cada vez que se modifique alguno de los operadores booleanos. Aunque existen maneras de implementarlo directamente en la GPU como se describe en GPU Gems 3 [12], esta implementación requiere el uso de *geometry shaders* y esta funcionalidad no está disponible en la plataforma WebGL 1.0.

2.2 Rendering de superficies implícitas con Raymarching

En esta aplicación se decidió utilizar la técnica de Raymarching para la representación de las superficies de la cuenca. Ésta no se basa en el uso de triángulos, sino en evaluar cada píxel de una imagen, a partir de representaciones implicadas [10] de superficies 3D descritas por funciones *Signed Distance Function* (SDF). Estas funciones son expresiones analíticas que representan campos de distancia, que miden cuán cerca se encuentra un punto $\boldsymbol{x}$ de un conjunto $\boldsymbol{S}$ y su signo cambia según se encuentre de un lado u otro del conjunto, como se indica en [13].

Las superficies 3D a ser representadas deben definirse usando múltiples funciones de distancia de formas geométricas primitivas [14] (esferas, planos, cubos, cilindros, etc.), combinadas mediante operaciones de unión, intersección, sustracción, etc. Esto plantea un desafío, si se desea representar alguna superficie arbitraria específica. En el caso particular de esta aplicación, las primitivas involucradas son pocas y simples (cubos, esferas, planos y cilindros).

Esta técnica ofrece grandes ventajas en términos de desempeño, ya que toda la lógica y los datos necesarios para resolver la vista de la escena, pueden incluirse completamente dentro un programa *pixel shader* escrito en lenguaje GLSL, sin necesidad de acceder a memoria para su evaluación. Este programa es suministrado al hardware de procesamiento gráfico (GPU) y es ejecutado muy eficientemente mediante múltiples núcleos de procesamiento que trabajan en paralelo, permitiendo resolver una vista completa en menos de 10 milisegundos, con el hardware adecuado.

2.3 El algoritmo de Raymarching

La esencia del algoritmo puede resumirse de la siguiente manera, para cada píxel de la pantalla, se evalúa un rayo definido por un vector origen (posición de la cámara u ojo del observador) y un vector dirección hacia el pixel en el plano cercano de la cámara. Luego mediante un proceso iterativo, se avanza sobre dicha dirección, evaluando en cada paso si el rayo impactó alguna superficie.

Existen diferentes variantes del algoritmo, algunas de ellas como la propuesta por [13] avanzan a paso constante y por lo tanto requieren un muestreo muy fino sobre el recorrido del rayo, lo cual impacta en el desempeño. Por otra parte, Hart presenta un método más robusto conocido como Sphere Tracing [15], donde la distancia avanzada en cada paso no es constante, sino que depende de la evaluación de una función de distancia SDF. Si el valor resultante es negativo significa que la posición evaluada se encuentra dentro de la superficie implícita definida por la SDF.

```
bool raymarch(vec3 rayOrigin, vec3 rayDirection) {
  float t=0.0f;
  for(int i=0; i<maxSteps; i++) {
    float d = sceneDistance(rayOrigin+rayDirection*t);
        if (d<epsilon)  return true;
        t+= d;
    }
  return false;
}
```

Cuando se cumple la condición $d < epsilon$, entonces se considera que el rayo impactó la superficie. A partir del punto de impacto se debe evaluar su color e iluminación.

El color de las superficies es la característica de interés central en la aplicación ya que permitirá al usuario reconocer el tipo de estrato o capa geológica. Se utilizó el modelo de Phong [16] para calcular la iluminación de las superficies.

2.4 Función de determinación del color

Dadas las coordenadas (*x, y, z*) de un punto de las superficies 3D obtenidas mediante Raymarching, es necesario determinar el color del pixel que dependerá del tipo de estrato predominante en ese punto de la superficie.

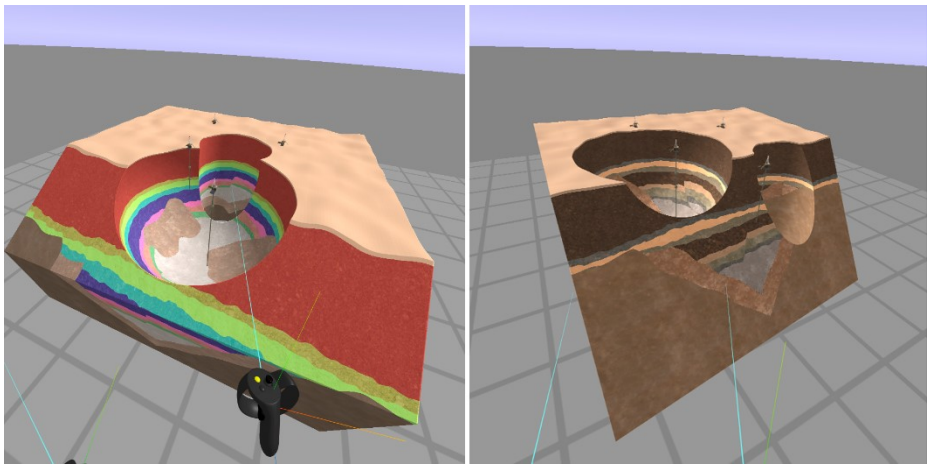

Fig. 4. Superficies coloreadas con escalas de colores (izq.) y con texturas fotográficas (der.)

Se evaluó la alternativa de codificar la composición de cada punto (*x, y, z*) de la cuenca utilizando múltiples texturas, donde para cada voxel se requerirían 8 bits para representar el estrato predominante. La ventaja de este esquema, es la facilidad para generar el conjunto de datos desde una aplicación externa y la simplicidad al momento de computar el color de las superficies. La desventaja es el alto consumo de memoria de GPU. Por ejemplo, si se divide el volumen en 1024 voxels por lado, el requerimiento de memoria RAM asciende a más de 1 gigabyte.

En el modelo que se observa en la figura 4, se definieron 11 tipos de estratos. Una función GLSL denominada *getTerrainColor* resuelve dicho color como una sumatoria de pesos que aportan el color de cada estrato a una variable que almacena el color RGB final.

Cada uno de estos pesos depende en algunos casos de la coordenada *z* y que valen 0, excepto en rangos específicos asociados a cada estrato. En otros casos los cálculos de los pesos tienen lógicas más complejas.

Además, se utilizan funciones de ruido pseudo-aleatorio para generar distorsiones a las coordenadas (*x, y, z*) que se usan como entrada de la función, para lograr un aspecto irregular en las distintas capas. El parámetro *colorMix* permite ajustar el esquema de colores entre 2 modos: a) un color constante para cada tipo de estrato, b) una textura predefinida para representar cada estrato (figura 4).

2.5 Mapeo de texturas

Para poder mapear una textura sobre una superficie, se requiere definir 2 coordenadas (*u, v*) que relacionen un punto (*x, y, z*) con un pixel de la textura (mapa de bits). Dado que las superficies en este caso se generan dinámicamente, se utilizó la técnica de mapeo *triplanar*. Esta técnica descripta por Geiss en [12], consiste en asignar (*u, v*) al plano coordenado XY, YZ o ZX que genere una menor distorsión en la proyección de

la textura; y esto se logra comparando el vector normal de la superficie N(*x, y, z*) con las normales de los 3 planos coordenados.

2.6 Interactividad y experiencia de usuario

Una vez iniciado el modo RV, el usuario está inmerso en un escenario virtual que consiste en un piso de extensión infinita y un cubo coloreado de 1 metro de lado que flota a la altura de la vista del operador. El usuario puede caminar alrededor o puede permanecer estático y modificar su ubicación en el espacio virtual utilizando la palanca del controlador derecho para controlar la velocidad e indicando la dirección de avance apuntando su mano en la dirección deseada.

El controlador de la mano izquierda permite mostrar y ocultar un panel de control virtual que consta de múltiples solapas. Mediante el controlador derecho, el usuario puede apuntar e interactuar con los diferentes controles (botones y cursores desplazables) confirmando la acción con el gatillo del controlador (figura 5).

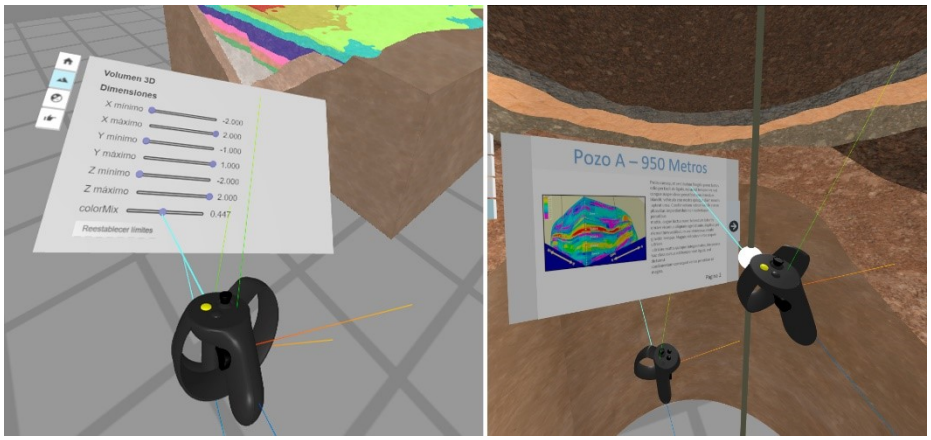

Fig. 5. Superficies coloreadas con escalas de colores (izq.) y con texturas fotográficas (der.)

Además, los controles de mano cuentan con botones laterales asociados al gesto de tomar y soltar, que permiten tomar, rotar, trasladar y soltar los 3 tipos de operadores booleanos, para recortar interactivamente el volumen 3D.

La escena 3D cuenta además con modelos 3D prediseñados que representan las torres de perforación y el recorrido del pozo. El usuario puede apuntar y seleccionar diferentes puntos a lo largo del pozo que funcionan a modo de "hipervínculos" desplegando información asociada a cierto punto de profundidad, en el panel de control virtual (figura 5 derecha). Por último, el usuario puede tomar una imagen instantánea desde su punto de vista, que automáticamente se descarga en formato JPG, permitiendo luego analizarla en detalle fuera de la realidad virtual.

2.7 Composición de la imagen final

Además de la representación visual del reservorio, otros elementos como el plano del suelo, el panel de control y los controles de mano deben ser renderizados utilizando triángulos. Por lo tanto, el proceso de renderizado se realiza en dos fases. En la primera, se renderiza la vista del reservorio, proyectando dos triángulos que cubren totalmente el *viewport*, cuyo shader de fragmentos computa el valor de cada píxel a partir del modelo implícitamente descrito en GLSL. Para cada fragmento se genera un vector RGB y un escalar que representa la profundidad (Z depth). En la segunda fase, se renderizan los modelos 3D restantes (controles de mano, pozos petroleros, panel de control, etc.) y se combinan con la fase 1 (Raymarching del reservorio) mediante el mecanismo de *depth testing*.

Cuando la aplicación funciona en modo "realidad virtual", el *framebuffer* se compone de dos mitades que representan la vista del ojo izquierdo y derecho.

3 Conclusiones

La aplicación desarrollada permite manipular y observar en una forma muy intuitiva el volumen 3D que representa la cuenca hidrocarburífera, ya que el usuario puede desplazarse directamente con su cuerpo en el espacio virtual y utilizar sus manos, eliminando la necesidad de aprender a utilizar complejas interfaces de usuario para mover y rotar objetos en 3D o cambiar el punto de vista, utilizando el teclado y el mouse.

Los operadores booleanos permiten estudiar fácilmente los límites entre los diferentes estratos y entender cómo se extienden dentro de la cuenca.

Además, la percepción de la profundidad aportada por la visualización estereoscópica ayuda a interpretar correctamente las formas y escala de las superficies generadas luego de aplicar recortes mediante los operadores booleanos.

La combinación de técnicas de Rendering permitió implementar una aplicación apta para ser utilizada en tiempo real con baja latencia, sin generar efectos incómodos habitualmente presentes en otras aplicaciones de realidad virtual.

En esta aplicación, la estructura interna de la cuenca se codificó en una función dentro del programa GLSL sobre la base de estudios geológicos previos integrales, de una cuenca productiva argentina. Los resultados obtenidos permitirán en un futuro implementar una representación basada en datos de múltiples trazas verticales (LAS) distribuidas en el volumen tridimensional.

La aplicación mejora la enseñanza de conceptos básicos sobre geología de las cuencas petrolíferas a partir de una interacción personal en 3D. Le permite al docente conducir al estudiante en el reconocimiento de los elementos de la cuenca y su contenido mineral a partir de la posibilidad de navegar en el modelo. Asimismo, y, con la incorporación de datos de pozos, el alumno podrá identificar las propiedades físicas de las formaciones, analizar el sistema petrolero y especular sobre su génesis y evolución en el tiempo.

Referencias

1. Jampeisov, Z.: Using Virtual Reality Technology in Oil and Gas Industry. 9, 124–127 (2019). https://doi.org/10.31033/ijemr.9.2.15.
2. Aveleyra, E.E.: Aportes para el debate: Las tecnologías en la enseñanza universitaria: nuevos escenarios, nuevos desafíos. En C. Nosiglia (comp.). La Universidad de Buenos Aires. Aportes para la CRES, pp.177-189. Editorial Universitaria de Buenos Aires, Buenos Aires (2018).
3. Cabero Almenara, J., Barroso Osuna, J.: Nuevos retos en tecnología educativa. Síntesis, Madrid (2015).
4. Coll, C., Monereo, C.: Psicología de la educación virtual. Aprender y enseñar con las tecnologías de la información y comunicación. Morata, Madrid (2008).
5. Martí, R., Gisbert, M., Larraz, V.: Ecosistemas tecnológicos de aprendizaje y gestión educativa. Características estratégicas para un diseño eficiente. Edutec. Revista Electrónica de Tecnología Educativa, (64), pp. 1-17 (2018).
6. Aveleyra, E.E., Racero, D., Gómez Toba, G. The didactic potential of AR in teaching physics. En: Memorias IEEE World Engineering Education Conference (EDUNINE), pp.1-3. Biblioteca digital IEEE Xplore® de IEEE, Buenos Aires, Argentina (2018).
7. Aveleyra, E.E., Proyetti Martino, M., Gómez Toba, G.: La Realidad Aumentada y la Enseñanza de la Física. En: Investigación, Innovación y Tecnología, la tríada para transformar los procesos formativos, pp.170-179. USACH, Santiago, Chile (2017).
8. García Marra, S., D. I. Maitia Petros, J. M. Lambre, R. A. Veiga y Ema Aveleyra, “Desarrollo de un Osciloscopio con Realidad Aumentada para un Curso Introductorio de Ingeniería Electrónica”, IEEE ARGENCON 2020, Resistencia, Chaco, Argentina (2020).
9. WebXR Standard, https://www.w3.org/TR/webxr, ultimo acceso 2021/6/5.
10. Menon, J.: An Introduction to Implicit Techniques. ACM SIGGRAPH 96 Course Notes: Implicit Surfaces for Geometric Modeling and Computer Graphics, pages A1–13 (1996).
11. Newman, T. S., Yi, H.: A survey of the marching cubes algorithm. 30, 854–879 (2006). https://doi.org/10.1016/j.cag.2006.07.021.
12. Nguyen, H.: GPU gems. Addison-Wesley, Upper Saddle River, NJ [u.a.] (2007).
13. Rendering Worlds with Two Triangles with raytracing on the GPU in 4096 bytes. https://www.iquilezles.org/www/material/nvscene2008/rwwtt.pdf, ultimo acceso 2021/6/5.
14. Modeling with distance functions, https://iquilezles.org/www/articles/distfunctions/distfunctions.htm, último acceso 2021/6/5.
15. Hart, J., C.: Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. The Visual Computer, 12(10):527–545, (1996).
16. Phong, B.: Illumination of Computer-Generated Images, Department of Computer Science, University of Utah, UTEC-CSs-73-129 (1973).