El 3D en relación con el argumento de la simulación

Por: Cesar Pachón

Disponible en: http://www.cesarpachon.com

cesarpachon@gmail.com

"este televisor tiene mejor resolución que el mundo real!"
(tomado de un episodio de "futurama")

Abstract

in 2003 Nick Bostrom presented a paper called “are you living in a computer simulation?” since then, lot of derivated work had been produced around this amazing idea, and had had a big influence in movies, books and popular culture. This paper presents a superficial review of our current technology in 3d graphics computation and tries to highlight common points and arguments in favor of a eventual possibility of taking our real technology as a base for the construction of such simulation, with the idea of support Bostrom arguments.

Resumen

En el año 2003 Nick Bostrom presentó un artículo titulado “estas viviendo en una simulación de computador?” desde entonces, gran cantidad de trabajos han derivado de esta asombrosa idea, y han sido una gran influencia en películas, libros y la cultura popular. Este artículo presenta una revisión superficial de nuestra tecnología de 3d actual y trata de resaltar puntos comunes y argumentos en favor de una eventual posibilidad de tomar nuestra tecnología real como base para la construcción de un simulador, con la idea de soportar los argumentos de Bostrom.

Acerca del autor

Estudiante de maestría en software libre, mg. En educación virtual (e-learning), ing sistemas UNAL, miembro del grupo de investigación PRISMA, investigador asociado del grupo CCPP de filosofía de la UNAL, consultor independiente en computación gráfica, realidad virtual y mundos virtuales. http://www.cesarpachon.com

INTRODUCCION

La tecnología influye o predetermina la forma en la que pensamos el mundo a través de las metáforas y analogías que construimos para explicarlo. Esto implica que cambios en la tecnología producirían cambios en dichas metáforas, en algunos casos sólo de forma, pero en otros, quizá un poco más profundos. Con la masificación de los computadores por ejemplo, la división de hardware y software se convirtió en una metáfora inevitable de la dualidad cuerpo y mente, metáfora de uso tan común que es corriente escuchar expresiones como "se me borró del disco duro" para hacer referencia a algo que no logramos recordar.

Podría considerarse que el argumento de la simulación es una aplicación un poco más profunda de esta metáfora o que por lo menos forma parte de manera implícita de su constitución. Este argumento, planteado por el profesor Nick Bostrom[1] de la facultad de filosofía de Oxford en el año 2003, bajo el título de "are you living in a computer simulation?", ha sido objeto de atención del mundo académico y también del imaginario popular, y podemos encontrar evidencias de esta idea en el cine y la literatura contemporáneos.

Este artículo no pretende explorar a fondo los razonamientos filosóficos que sustentan el argumento de la simulación, sino más bien especular un poco con las posibilidades del mismo desde el punto de vista de la computación gráfica y en particular, la generación de gráficas tridimensionales en tiempo real, a partir de la siguiente pregunta: si nosotros fueramos los encargados de crear una simulación como la propuesta por el profesor Bostrom, que tipo de algoritmos y técnicas conocidas actualmente en el área de la computación gráfica tendrían algún potencial para ser aplicadas en el desarrollo de dicho simulador?

LA TUBERIA DE VISTA y el mito de la caverna

En "La república" Platón pone en boca de Sócrates la conocida explicación del mito de la caverna como una analogía que pretende describir la diferencia entre lo perceptual o sensible, y lo real o inmutable; mejor conocido como "el mundo de las ideas". En esta analogía, los hombres estamos condenados a concentrar nuestra atención en proyecciones incompletas, parcializadas e imperfectas de las cosas, que son las que impactan nuestros sentidos, y a darle la espalda a los objetos reales que producen dichas proyecciones ó sombras[2].

La idea de que todo lo que consideramos real es sólo una proyección parcial y por tanto imperfecta de algo que existe más allá es sospechosamente similar a la idea de la realidad dentro del simulador. Y la similitud no es casual: desde el "velo de Maya" indú hasta el argumento de la simulación existe una gran cantidad de trabajos y reflexiones de filósofos de todas las épocas, escuelas y opiniones con respecto a la naturaleza misma de la realidad y su relación con nuestras estructuras cognitivas.

Podemos organizar a estos pensadores ubicándolos sobre un eje que en un extremo llevaría la etiqueta de "empiristas", y en el otro la de "racionalistas".

Los filósofos empiristas como Hume y Locke sostienen que la única fuente posible de conocimiento es la percepción, es decir, los estímulos que recibimos a través de los sentidos. De ser así, no tiene mucho sentido hablar de algo más real que aquello que puede impactar nuestros órganos sensoriales, y limitar la definición de "real" a lo perceptible o derivado de ello es la posición más razonable.

En el otro extremo los racionalistas rechazaron a la percepción como fuente de conocimiento y se concentraron en la reflexión, en nuestra propia estructura interna para a partir de allí ir dando validez y carácter de "real" a aquello que podía ser deducido ó sustentado por experiencias subjetivas. Es el caso de Descartes y su conocido "Discurso del método"

Figura 1: El mito de la caverna

Ahora bien, para explicar el mito de la caverna en términos de computación gráfica, basta con comprender el concepto de "tubería de vista" para ver como ambos conceptos encajan, al menos en sus extremos.

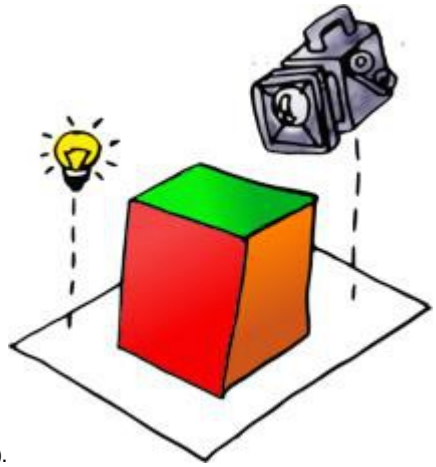
QUE ES UNA TUBERIA DE VISTA

Uno de los extremos que nos describe Platón es el de los hombres esclavos que viven al fondo de la caverna, atados de manera que toda su vida sólo han podido ver las sombras y ecos de los objetos que pasan por detrás.

Para Platón estas sombras y proyecciones son los equivalentes a las imágenes que llegan a nuestra conciencia gracias al sentido de la vista. Así que es completamente directo incluir dentro del conjunto de proyecciones a aquellas generadas por algoritmos de computación gráfica dentro de un computador, y desplegadas a través de cualquier dispositivo de visualización, como un monitor. Continuando con nuestro paralelo, el esclavo vendría a ser el usuario sentado al frente de la pantalla, tratando de escudriñar a través de una matriz de pixeles iluminados las verdaderas estructuras (objetos) responsables de los valores y cambios de dichos pixeles.

Pero Platón incluye otros elementos en su descripción: están los objetos reales, que desfilan al frente de la puerta de la cueva, y la cueva en sí misma, importantísima por que no es simplemente algo que limita la percepción, sino que en realidad la posibilita, eso sí, añadiendo deformaciones y pérdidas de información a lo percibido, por ejemplo, las

distorsiones acústicas (eco, reberveración) y ópticas (la pérdida de color, de profundidad, los cambios en la geometría



de los objetos causada por las diferencias en las proyecciones de la luz).

Figura 2: Escena ideal

Los objetos "reales" en nuestra analogía vienen a ser el modelo que tiene en memoria el computador, el cual contiene toda la información necesaria para que a partir de la aplicación de determinados algoritmos se pueda producir imágenes (proyecciones) del objeto en cuestión.

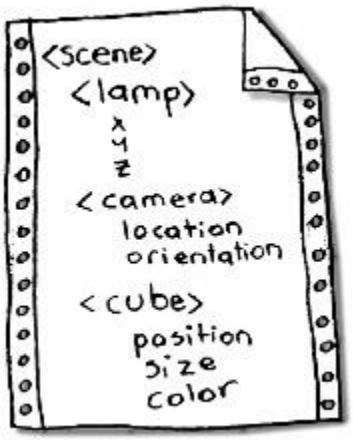


Figura 3: Escena en archivo

Para entender exactamente la naturaleza de este objeto, pensemos en un ejemplo más concreto: nuestro objeto podría estar representado por un modelo descargado de internet, quizá desde el catálogo de Google Sketch Up, ó un modelo gratuito para el programa de modelado 3D libre "Blender 3D". en cualquiera de los casos, el modelo estará almacenado en un archivo en alguna parte del disco duro del ordenador. Para comprender hasta qué punto puede ser válida la analogía consideremos el siguiente hecho: si el computador es apagado, el modelo ("objeto real") seguirá existiendo, independiente del hecho de que no se estén ejecutando los algoritmos que lo proyectan, y que no exista un observador atento a recibir sus proyecciones. Este punto no sólo es consistente con el principio de inmutabilidad de las ideas en

Platón, sino que además parece inevitable la evocación de "cosa en sí" de shopenhauer [3], al menos en una primera aproximación.

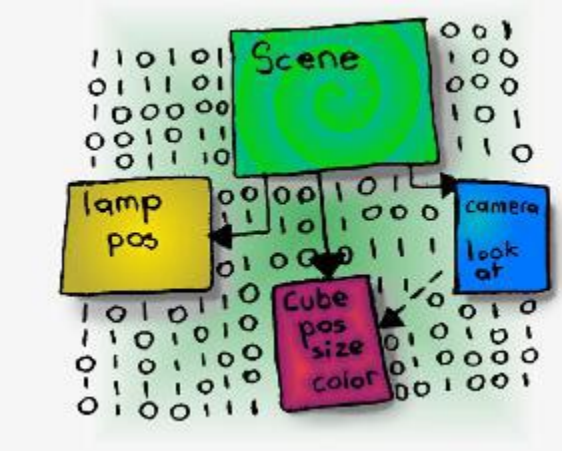


Figura 4: Representación en memoria

Y la cueva? Pues bien, entre el sujeto que percibe (usuario sentado al frente del monitor) y el objeto real (archivo con la información del modelo a visualizar) existe toda una compleja colección de capas de software e incluso de hardware que reciben el nombre de "tubería de vista".

Antes de entrar a revisar en detalle la estructura de una tubería de vista, hay que aclarar que en términos estrictos estamos saltando o mejor aún, dando por supuesta la existencia de un componente adicional de software, una pieza que se ubicaría entre la tubería de vista propiamente dicha y el modelo, encargado de cargar, leer e interpretar el modelo, y mantener en la memoria de acceso rápido del computador (RAM) una estructura equivalente del modelo original que pueda ser enviada de manera eficiente por la tubería de vista. Llamaremos a este último componente una "librería de modo retenido", y a la tubería de vista en sí, "librería de modo inmediato". Continuaremos hablando de la tubería de vista entendida esta como modo inmediato, y regresaremos al concepto de modo retenido hacia el final del artículo.

El "modo inmediato" y el "modo retenido" son conceptos bastante clásicos en computación gráfica. Por ejemplo, OpenGL, la librería gráfica 3D más usada (y que en realidad es una especificación libre, con diferentes implementaciones de muchos fabricantes y para muchos tipos de hardware) es estrictamente hablando una tubería de vista de modo inmediato, ya que no ofrece ningún tipo de mecanismo para interactuar con los archivos de modelos que un usuario podría tener en su disco duro. Esto no es algo para nada malo: de hecho, su competencia directa, el Direct X de Microsoft, en sus orígenes (versión 6, allá por el año 2003) ofrecía de manera separada el Direct3D Modo Retenido y el Direct 3D Modo inmediato. El modo retenido estaba pensado para programadores con bajo nivel, incapaces de programar sus propias representaciones de los modelos a visualizar. Este modo desapareció del mercado en la siguiente versión, y nunca más volvió a regresar.

Para entender mejor cómo funciona a grandes rasgos una tubería de vista imaginemos el siguiente ejemplo:

Un pintor sale de viaje, dejando en su casa todos sus materiales de trabajo. Durante el viaje, se le ocurre un cuadro el cual quisiera realizar en aquel mismo instante. Afortunadamente, nuestro pintor ha tenido la precaución de adquirir un novedoso sistema de tele-pintura, el cual consiste en un brazo robot conectado a un sistema de reconocimiento de voz a través del teléfono. Entusiasmado por probar el nuevo sistema, el pintor toma su teléfono y se comunica a su estudio. El robot se activa, y espera las instrucciones de pintado. Desafortunadamente no es un sistema muy inteligente: el pintor no puede decir "Pinta un cielo azul", ya que el robot no entiende que es "cielo". De hecho, tampoco entiende qué es

azul**.** El pintor dará comandos del tipo: "trazar zona desde punto 0,0 hasta punto 20, 30 con color de posición de paleta número 1". Ya se podrá imaginar el lector la enorme cantidad de pequeños comandos que tendrá que realizar el pintor, sin tener la menor idea de cómo va quedando su obra! El robot sigue las instrucciones simples, sin tener la menor conciencia del tipo de cuadro que está pintando. De hecho, si el Pintor olvidó llenar de pintura la paleta antes de salir de viaje, es perfectamente posible que el robot esté pintando el cuadro ... sin pintura! (la versión 2 de nuestro sistema traerá un sistema de alarma para prevenir este incidente).

Terminado el ejemplo, se puede ver cuál es la forma de trabajo de una tubería de vista: aceptar montones de pequeños comandos, uno a la vez, relacionados con objetos sencillos. Las tuberías de vista trabajan preferiblemente con triángulos, y por eso es que las tarjetas de video se evalúan según su capacidad de pintar cierta cantidad de millones de triángulos por segundo. Así pues, es responsabilidad del componente de modo retenido transformar una representación compleja de un modelo en un conjunto de triángulos para ser enviados por la tubería de vista.

Los triángulos enviados de esta manera son transformados, escalados, rotados, deformados, recortados, comparados unos contra otros, en una rápida sucesión de operaciones matemáticas que los preparan para la etapa final. Este primer conjunto de procesos ocurre en un espacio tridimensional pero en la etapa final conocida como "rasterizado" se opera a nivel 2D, convirtiendo las proyecciones de los triángulos sobrevivientes en conjuntos de pixeles que se mezclan para producir la imagen final.

Diferentes usuarios poseerán máquinas con características distintas. En una máquina poderosa, algoritmos más complejos se pueden ejecutar sin sacrificar la tasa de refresco (velocidad de pintado) del sistema. En una máquina más modesta, se requiere usar versiones ligeras de los mismos algoritmos, sacrificando calidad e incluso introduciendo "artefactos" o efectos extraños en la visualización. Compárese esto con la afirmación de que las percepciones son subjetivas al observador, y pueden variar entre diferentes observadores, mientras que "la cosa en sí" es inmutable y perfecta.

Figura 5: escena como triángulos

Hasta aquí, hemos establecido un paralelo entre el mito de la caverna y una tubería de vista. Se comparó al esclavo con el usuario final, a las ideas inmutables con los modelos a visualizar, y a la caverna en sí con el software y hardware que componen el modo inmediato de una tubería de vista. Pero podemos continuar ampliando la analogía y acercándonos más al concepto de la teoría del simulador de Bostrom si nos concentramos en el componente que intencionalmente

dejamos de lado: el software de modo retenido que intermedia entre el modelo a visualizar y la tubería de vista, y que alimenta a esta última con el río de triángulos a procesar. Es en este componente donde aparecen los algoritmos más interesantes para el tema que nos ocupa, ya que tienen mucha relación con los argumentos a favor y en contra de la teoría del simulador.

OPTIMIZACIONES Y ALGORITMOS DE MODO RETENIDO: APROXIMACION A LA TEORIA DEL SIMULADOR

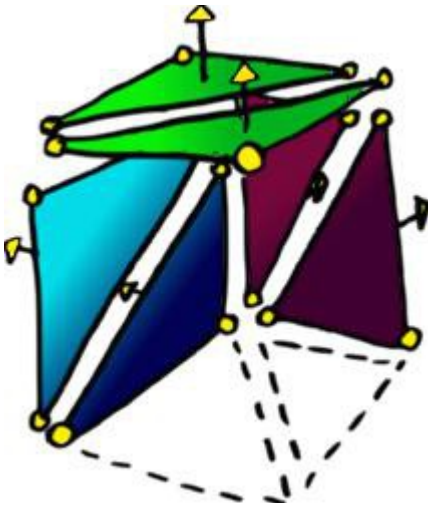


Figura 6: Eliminación de superficies ocultas

En el ejemplo del desafortunado pintor usando el sistema de tele-pintura quedó claro que una de las limitaciones del mismo era la "poca inteligencia" del sistema sobre su dominio de operación. También resultaba evidente que una mejora en el nivel de conocimiento que el sistema tuviese radicaría significativamente en mejoras para el usuario final. En otras palabras, si el sistema de alguna manera poseyera conceptos similares a los que usa un pintor, la comunicación con el usuario sería mucho más eficaz.

El modo retenido puede pensarse como algo así: en vez de limitarse a conjuntos masivos de triángulos, este componente puede poseer diferentes estructuras y grafos de estructuras que le permite tener más "inteligencia" sobre la escena, extrayendo de esta triángulos de una manera más eficiente.

Qué tiene esto que ver con el argumento del simulador de Bostrom?

Bueno, algunas de las objeciones típicas al argumento del simulador tienen que ver con temas como la cantidad de recursos (energía, recursos de procesamiento) necesarios para ejecutar una simulación que pudiera pasar por ser "el mundo real", la complejidad de modelar los diferentes niveles de detalle que encontramos en el mundo real, y el manejo de inconsistencias en el proceso de simulación. Veremos que estos problemas a pequeña escala han sido una constante en la industria de los videojuegos y cómo han lidiado con ellos desde el desde el modo retenido.

OBJECION DE CAPACIDAD DE PROCESAMIENTO

Una de las objeciones a la teoría del simulador es la de la viabilidad técnica para construir un computador capaz de ejecutar todos los procesos asociados a la simulación. Usando leyes de la termodinámica, se estimaron las necesidades

de tamaño, masa y energia que se requerirían para crear un computador capaz de correr una simulación lo suficientemente compleja [4].

En los videojuegos ocurre algo similar. Desde el punto de vista de las gráficas 3D, el número de triángulos a procesar por la tubería de vista supera en varios órdenes de magnitud la capacidad física de la máquina. El truco para que la necesidad de procesamiento se adapte a la disponible consiste en reducir la cantidad de polígonos que se envía a la tubería a sólo los estrictamente necesarios según el punto de vista de la cámara (no procesar lo que no se ve) y esto se logra gracias a la combinación de diferentes técnicas, que van desde las sencillas hasta algoritmos verdaderamente sorprendentes.

Entre las técnicas sencillas podemos citar:

- backface culling: normalmente los triángulos sólo se pintan por un lado. Basados en la dirección en la que está la cara visible del triángulo y comparándola con el punto de vista de la cámara, se pueden descartar los triángulos que no miran hacia ella.
- far clipping: se define una distancia máxima desde la cámara, a partir del cual los objetos se descartan por completo.
- Fog: esta técnica permite agregar un efecto de niebla que diluye los objetos lejanos, facilitando aplicar la técnica de far clipping.
- frustum culling: el volumen de vista de la cámara tiene forma de pirámide truncada. La idea es comparar los objetos con los planos que componen esa pirámide para descartar aquellos que están fuera de ella.
- Bounding Boxes: para simplificar los cálculos en vez de evaluar contra cada triángulo de un objeto, se envuelve este en un sólido sencillo, como una caja. Existen diferentes tipos, como el AABBOX (caja alineada a los ejes) y el OOBBOX (caja alineada al objeto).

y a nivel de técnicas más elaboradas, están las diferentes estructuras y algoritmos de partición de espacio, como por ejemplo:

- árboles BSP: árboles de partición binaria. Toman el plano de un triángulo del modelo y empiezan recursivamente a clasificar los demás triángulos en "atrás" y "adelante" del triángulo de referencia. Tiene el inconveniente de que los triángulos que chocan con el plano de referencia deben ser cortados y esta operación tiende a producir polígonos complejos, malo para la tubería de vista.
- Quadtrees y Octrees: estructura de partición del espacio basada en la división en estructuras del mismo tamaño que se aplica recursivamente. Los quadtrees se usan en 2D y son cuadrados divididos en cuadrantes. Los octrees usan cubos. Grafos de escena: los motores de visualización 3D modernos usan el concepto de grafo de escena, que es un grafo de objetos emparentados según la lógica de la escena y generalmente basados en los bounding boxes de los objetos del grafo para realizar las operaciones de descarte de polígonos.

OBJECION DE NIVELES DE DETALLE

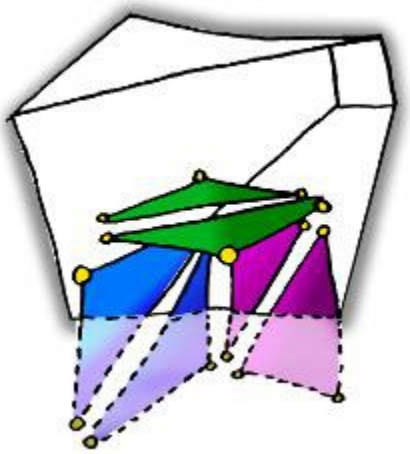


Figura 7: recorte contra volumen de vista

Esta objeción hace referencia a que sería demasiado complejo simular todos los niveles de detalle de un objeto. Por ejemplo, si en la simulación se requiere una silla, habría que modelar la silla a nivel de estructura, luego a nivel de átomos, y luego a nivel subatómico.

La respuesta a este argumento es que sólo se necesita simular el nivel de detalle con el que está interactuando en ese momento de manera consciente el observador. Por ejemplo, si el observador simplemente interactúa con la silla para sentarse en ella, no es necesario procesar un modelo de la misma que represente toda su estructura atómica.

Esta respuesta es bastante similar a la manera como se soluciona el problema en el mundo de las gráficas 3D: es muy usual que se requiera representar ciertos objetos con modelos muy detallados (de muy alta poligonalización, en la jerga de la industria) por ejemplo los personajes principales de un videojuego o en general cualquier objeto que tiene la posibilidad de ser enfocado de cerca por la cámara. Sin embargo, si la cámara se aleja del objeto, llegará un punto en el que este se verá tan pequeño que podría ser reemplazado por un modelo con menos triángulos, sin perder calidad gracias a la distancia. Se puede crear un conjunto de modelos, cada uno más simple que el anterior, e irlos reemplazando según cambia la distancia entre el objeto y la cámara. A esta familia de técnicas se le conoce como LOD (nivel de detalle) y se puede llevar a casos tan extremos como representar el objeto simplemente como un plano sobre el que se sobrepone una fotografía del objeto (Impostores), hasta algoritmos capaces de generar automáticamente los modelos intermedios y algoritmos capaces de reducir el "salto visual" causado por el cambio brusco de nivel de detalle en un modelo y que puede resultar perturbador para el observador (popping).

Las técnicas de LOD no se aplican sólo a los objetos del escenario, también al terreno en sí mismo, existiendo muchos algoritmos para simplificar dinámicamente la geometría de un terreno, como ROAM, el cual, combinado con técnicas de suavizado de popping como el geomorphing, producen resultados impecables.

OBJECION DE INCONSISTENCIAS DE SIMULACION

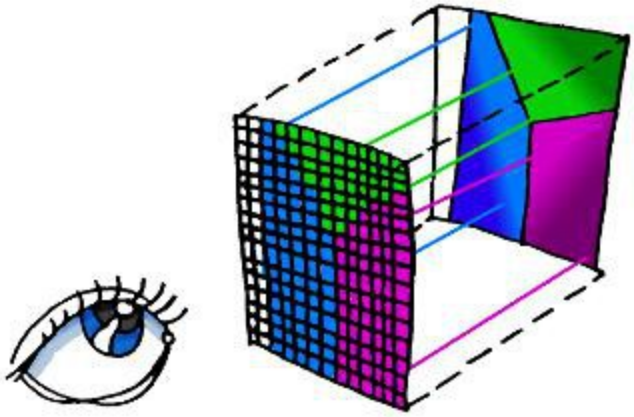


Figura 8: Proyección a pantalla

Al ejecutar una simulación compleja es inevitable que ocurran de vez en cuando errores que puedan ser percibidos por los agentes dentro del simulador. (recordemos los deja-vu en la película Matrix). La respuesta a esta objeción es que si bien es cierto que ocurrirán estos errores, también es cierto que el tiempo en la simulación es también algo simulado, por lo que puede volverse atrás arbitrariamente para reprocesar la simulación desde un punto anterior a la ocurrencia del error, incluyendo la conciencia de los agentes dentro del simulador.

Así funcionan muchos de los motores de simulación física usados en los videojuegos. El concepto aquí es el tamaño de paso adaptativo. Supóngamos que se tiene una esfera (un balón, por ejemplo) lanzado hacia una pared. Conocemos la fórmula para modelar su comportamiento, es decir, predecir su posición a través del tiempo (ecuaciones de newton). Asi que se calcula su posición en un instante t1, poco después de ser lanzada, y el siguiente paso del simulador ocurre un par de segundos después, en t4. El problema aquí es que al evaluar de manera discreta (primero t1, luego t4) en ninguna de ambas posiciones el balón choca con la pared. En t1 está a un lado de la pared, y en t4 al otro. En t3 sí que choca, pero con el tamaño de paso del simulador tan alto, nunca se evalua t3. El resultado: la pelota atraviesa mágicamente la pared.

Entonces, si se sabe que la pared existe, lo que se puede hacer es tratar de encontrar el tiempo exacto en el que al evaluar la simulación la pelota se detendrá contra la pared. Una forma de hacerlo consiste en empezar con un tamaño de paso grande e ir reduciendo a mitades hasta lograr el resultado buscado. En otras palabras, manipular el tiempo de la simulación.

CONCLUSION

A lo largo del artículo hemos desarrollado una presentación superficial del funcionamiento de una tubería de vista para la representación y visualización de escenas tridimensionales en el computador, presentándola como una analogía válida y contemporánea del mito de la caverna y el argumento de la simulación, y hemos aprovechado esta presentación para en paralelo realizar ciertas reflexiones relativas a las objeciones que desde el punto de vista técnico han sido esgrimidas frente al argumento de la simulación.

Sin pretender constituir una defensa sólida de dicho argumento, este artículo buscaba mostrar al lector cómo muchos de los interrogantes y objeciones que parecen surgir del sentido común en contra de la idea de una realidad simulada han sido (guardando las proporciones) problemas cotidianos de la industria de la computación gráfica.

Trabajo a futuro

La visualización de gráficas tridimensionales en tiempo real es sólo uno de los aspectos en los que el estado del arte de las ciencias de la computación actuales pueden arrojar luces sobre el argumento de la simulación. Otros campos igualmente fascinantes y fructíferos son los métodos numéricos, los mundos virtuales masivos persistentes (MMOG) con toda la tecnología de redes de telecomunicación y sincronización de simuladores, y por supuesto, la inteligencia artificial especialmente la aplicada a videojuegos.

Referencias

[1] Bostrom, Nick, Are You Living In a Computer Simulation? Philosophical Quarterly, 2003, Vol. 53, No. 211, pp. 243-255.

[2] Platón, La República, Libro IX

[3] Shopenhauer Arthur, El mundo como voluntad y representación

[4] Sandberg Anders, The Physhics of information processing superobjects: Daily Life Among the Jupiter Brains, 1999, disponible en linea en http://www.jetpress.org/volume5/Brains2.pdf