PROYECTO MINECRAFT

William Castro 6000180, Juliana Ordoñez 6000458

Universidad Militar Nueva Granada

Cajicá

Resumen-En este proyecto vamos a realizar una investigación para hacer un video juego en three.js con la geometría cubo, haciendo uso de la implementación de colisiones entre cubos y entre otras para así poder llegar a imitar una parte del video juego Minecraft.

I. INTRODUCCIÓN

En este documento presentamos la propuesta al proyecto de computación gráfica a realizar, entonces iniciamos con un mundo de cubos donde tenemos un personaje, a la audiencia se le muestra la visión principal del personaje en modo primera persona conocido normalmente en el mundo puedes colocar diversos materiales predeterminados, y construir con estos estructuras cúbicas.

II. OBJETIVO

- Obtener información de cómo implementar las colisiones en cubos.
- Realizar el análisis , diseño y desarrollo de el juego minecraft utilizando la metodología de colisiones en three.js

III. MARCO TEÓRICO

A.VIDEO JUEGO

Es una aplicación interactiva creada para el entretenimiento, en donde se plantean un conjunto de retos u objetivos a cumplir por el usuario en un mundo acotado. Dicho mundo, suele estar conformado por un conjunto de escenarios virtuales creados en 2D o 3D donde conviven diversas entidades. Comúnmente, se le delega el control de una o varias entidades al usuario garantizando la interacción y la retroalimentación con éste. Formalmente, la mayoría de videojuegos suponen un ejemplo representativo de aplicaciones gráficas de despliegue en tiempo real, donde los escenarios virtuales que componen su mundo son descritos matemáticamente para que puedan ser manipulados por un computador. Representando así, una simplificación de la realidad, incluso para los casos en que estos se basan en una ficticia, ya que resulta impráctico incluir todos los detalles que posee. En la informática gráfica, el despliegue o rendering se define como un proceso de generación de una imagen que representa a una escena, bien sea 2D o 3D, en un dispositivo de salida. Generalmente, las escenas están compuestas por un conjunto de objetos, que se definen mediante recursos o por algoritmos que los generan. Según [1], desde un punto de vista abstracto, una aplicación gráfica de despliegue en tiempo real se basa en un ciclo donde en cada iteración se realizan los siguientes pasos:

- El usuario visualiza una imagen resultante del proceso de despliegue.
- El usuario actúa en función de lo visualizado, interactuando directamente con la aplicación, por ejemplo mediante un teclado.
- En función de la acción realizada por el usuario, la aplicación gráfica genera una salida u otra, es decir, existe una retroalimentación que afecta a la propia aplicación.

En el caso de los videojuegos, este ciclo de visualización, actuación y despliegue ha de ejecutarse con una frecuencia lo suficientemente elevada como para que el usuario se sienta inmerso en el videojuego, y no lo perciba como una sucesión de imágenes estáticas. De esta forma, se debe desplegar al menos quince cuadros por segundo para que el ojo humano no note discontinuidades en la animación.

B.MINECRAFT

Es un videojuego de mundo abierto donde la exploración y las construcciones son fundamentales. Creado por *Markus <<Notch>> Persson*, nos permite desarrollar nuestros propios universos fantásticos y artísticos, mediante la colocación y destrucción de bloques. Al ser un videojuego de mundo abierto, no tiene una misión concreta (salvo alguno de sus modos de juego) y consiste en la construcción libre mediante el uso de cubos con texturas

tridimensionales. Los bloques representan distintos elementos de la naturaleza y el jugador puede desplazarse por su entorno y modificarlo mediante la creación, recolección y transporte de esos bloques. Nunca se generan dos mundos iguales, pues se crean mediante el uso de algoritmos.
El juego de los cabezas cuadrados se divide en cuatro **modos de juego**:

- **Supervivencia**: Es el modo principal. Tenemos que conseguir recursos y sobrevivir al ataque de múltiples criaturas que surgen en la oscuridad, hasta que logremos vencer al dragón. Nuestro personaje cuenta con diez corazones de vida para conseguir su objetivo.
- **Creativo**: Las reglas de juego las ponemos nosotros y podemos construir libremente. Tenemos recursos ilimitados de todos los bloques y objetos del juego. No somos atacados por ningún monstruo, ni recibimos ningún tipo de daño.
- **Aventura:** Destinado para los jugadores que se dedican a crear mapas para otros usuarios. No podemos romper bloques para hacer construcciones, sólo podemos romperlos si contamos con la herramienta adecuada. En algunas versiones, no se puede modificar la dificultad.
- **Modo Espectador:** si nos matan, podemos seguir jugando, pero sí ver la partida.

Es un juego muy útil para iniciar a los chic@s en el mundo de la programación ya que se ve algo de código Java, especialmente a la hora de modificar las clases de los personajes, crear variables y condicionantes. Los mods, son el camino para enseñarles código de forma sencilla. Además, para jugar en modo multijugador, podemos aprender a crear nuestros propios servidores.

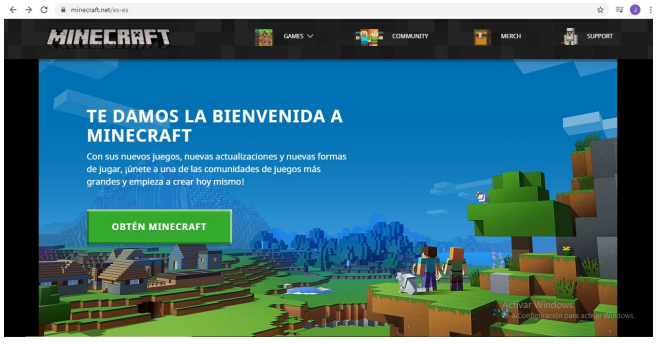

Fig.1 imagen tomada de la pagina de Minecraft

C.Polygons, edges, vertices and meshes
Las formas 3D están compuestas de pequeñas formas 2D:

- Polygons: Formas 2D (por ejemplo en Unity son triángulos)
- Edges: Los triángulos están formados por edges (bordes) conectados entre sí.
- Vértices: Punto en el que convergen los edges.
- Meshes: son formas complejas creadas a partir de la interconexión de muchos polygons compuesta de tres partes caras,aristas y vértices.

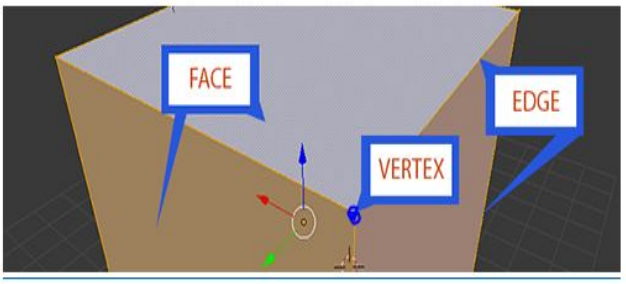

Fig.2 ejemplo de malla en un cuadrado.

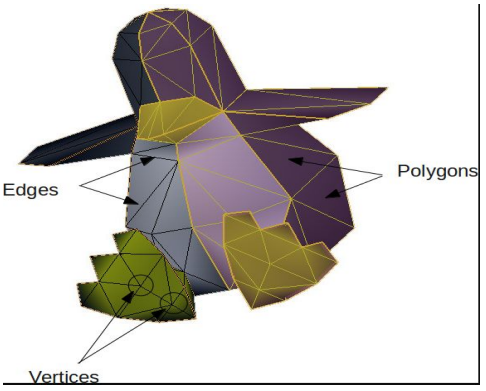

Fig.3 ejemplo de polygons,edges and vertices

Al conocer estos puntos y polygons, los motores de juego son capaces de hacer cálculos sobre los puntos de impacto, conocidos como colisiones. Se utiliza la detección de colisiones complejas con Mesh Colliders (colisionadores de malla), por ejemplo en juegos de shooters para detectar la ubicación exacta en la que una bala impacta sobre un objeto. Los meshes pueden tener otros usos, por ejemplo se pueden utilizar para especificar una forma del objeto menos detallado.Esto puede ayudar a mejorar el rendimiento del motor de física evitando que este compruebe un mesh muy detallado.
D.TECNOLOGÍAS A USAR
THREE.JS
Es una biblioteca escrita en JavaScript para crear y mostrar gráficos animados en 3D en un navegador Web. Puedes ver ejemplos y descargarla en http://threejs.org. Esta biblioteca fue creada y liberada en GitHub por el español Ricardo Cabello en abril de 2010, conocido por su seudónimo de Mr. Doob. Con el tiempo se ha popularizado como una de las más importantes para la creación de las animaciones en WebGL. A día de hoy hay más de 390 colaboradores que se encargan de mejorarla.

WebGL

Tradicionalmente, para utilizar gráficos 3D era necesario crear una aplicación autónoma 1 usando un lenguaje de programación como C o C ++ y una API gráfica especializada, restringiendo su uso a sistemas muy específicos como computadoras con una gran capacidad de cómputo o a consolas de videojuegos. Con el gran avance que han experimentado los navegadores, junto con su amplia gama de tecnologías de desarrollo se volvió una necesidad el uso de gráficos 3D en la web. En un principio, se tomó un subconjunto de instrucciones de OpenGL y surgió una versión de dicha API para sistemas embebidos o embedded systems llamada OpenGL ES, donde el sufijo "ES" es un acrónimo que se usa para hacer referencia a dichos sistemas. OpenGL ES estaba diseñado para ser utilizado exclusivamente en dispositivos móviles. Posteriormente, éste se adaptó para que pudiera funcionar en los navegadores, dando como resultado la creación de WebGL. Por lo tanto, se puede definir a WebGL como una API que permite el despliegue e interacción de gráficos 3D dentro de los navegadores. WebGL es capaz de combinarse con tecnologías como HTML5 y JavaScript dándole así a los desarrolladores web un fácil acceso a los gráficos 3D. Adicionalmente, juega un rol importante en el desarrollo de interfaces de usuario minimalistas e intuitivas así como contenido web interactivo.

DETECCIÓN DE COLISIONES Y LEYES DE LA FÍSICA

Al desarrollar juegos o en ciertas escenas es necesario simular cómo las leyes de la física afectan a los objetos presentes. Para ello hay que tener en cuenta parámetros como la gravedad, la aceleración, el volumen, la velocidad, el peso o si las superficies son resbaladizas o rugosas. También hay que determinar qué pasa cuando dos objetos chocan entre sí, por ejemplo, las bolas de fuego que lanzan los magos al impactar contra los enemigos. Por suerte disponemos de varias librerías JavaScript que realizan buena parte de los cálculos para nosotros.

COLISIONADORES

Son áreas que envuelven al GameObject, utilizadas para detectar las colisiones entre diferentes objetos del juego. Un collider está formado, por un lado, de una determinada forma, que se recomienda sea lo más ajustada a la forma del objeto, y por otro lado, de un determinado material físico, que proporcionen características de rebote o fricción.

DETECCIÓN DE COLISIONES

La detección de colisiones ha sido un tópico de extenso estudio en computación gráfica y geometría computacional. Sus campos de aplicación incluyen la robótica, biología computacional, juegos, y simulaciones en cirugía, física y biología.

BOX3

Los cuadros delimitadores son de tipo Box3 y tienen métodos útiles como intersection Box, contains Box o contains Point que satisfará todas sus necesidades cuando desee detectar colisiones, la propiedad bounding Box contiene un objeto THREE.Box3. Este objeto Box3 consta de dos objetos THREE.Vector3, min y max.

```
var geometry = new THREE.CylinderGeometry (...);
var material = new THREE.LineBasicMaterial (...);
var mesh = new THREE.Mesh (geometría, material);

var boundingBox = mesh.geometry.boundingBox.clone ();
alert ('coordenadas del cuadro delimitador:' +
      '(' + boundingBox.min.x + ',' + boundingBox.min.y + ','
+ boundingBox.min.z + '),' +
     '(' + boundingBox.max.x + ',' + boundingBox.max.y + ','
+ boundingBox.max.z + ')');
```

Para formas más complejas, como las que se cargan desde archivos de objeto JSON, la propiedad del cuadro delimitador no está definida de forma predeterminada. Debe calcularse explícitamente.

```
var loader = new THREE.ObjectLoader ();
loader.load (imagePath, function (object) {

      geometría = object.children [0] .children [0] .geometry;
// sustituye la ruta por tu geometría

       geometry.computer BoundingBox (); // de lo contrario,
geometry.boundingBox no estará definido

     var boundingBox = geometry.boundingBox.clone ();
     alert ('coordenadas del cuadro delimitador:' +
        '(' + boundingBox.min.x + ',' + boundingBox.min.y +
',' + boundingBox.min.z + '),' +
         '(' + boundingBox.max.x + ',' + boundingBox.max.y
+ ',' + boundingBox.max.z + ')');
}
```

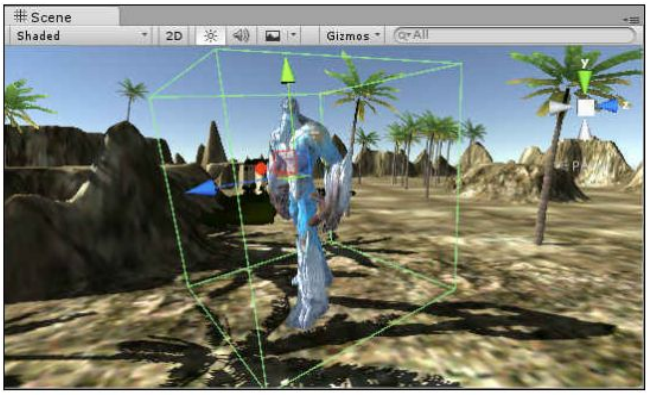

Fig.4 ejemplo de colisiones en forma de cubo del software unity

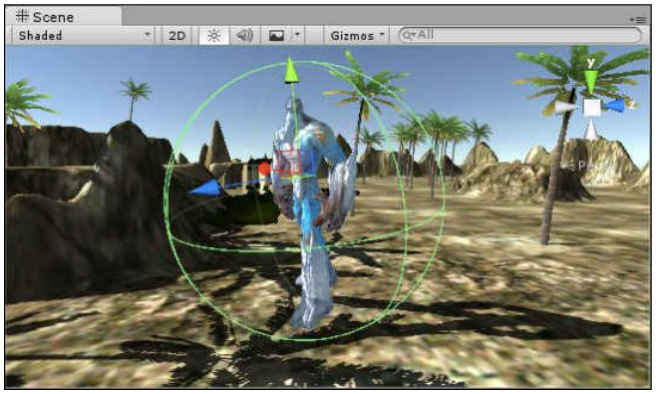

Fig.5 ejemplo de colisiones en forma de círculo del software unity

MESH COLLIDER (colisionar malla)

Colisionador de malla. El colisionador se acopla a la malla del objeto al que se le asigna. Es una forma de obtener colisiones de manera muy precisa, pero consume muchos recursos.

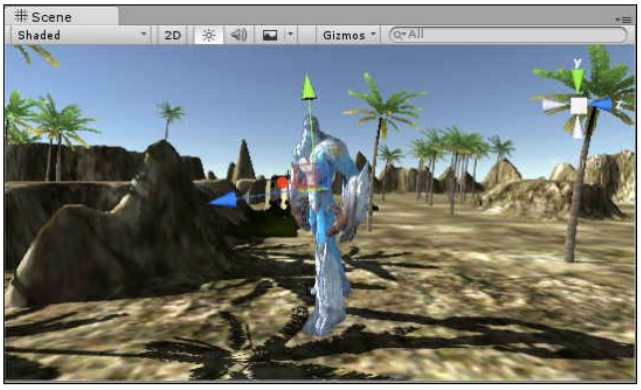

Fig.6 ejemplo de mesh collider en forma de cuadrado del software unity

IV. CRONOGRAMA

CRONOGRAMA				
FASES	ACTIVIDADES	10-sep	17-sep	24-sep
Como?	Definir idea	Buscar id	Idea lista	
			Investigacion	
	Investigacion :		Resumen	
	1. Resumen, Introcuccion, Objetivos		Marco Teorico	
	2. Marco Teorico, Cronograma,		Alace y Disseño	
	3. Alcance, Diseño			Entrega Investigacion
Creacion	Colusiones De Cubos			
	Cracion de Camaras			
	Implementacion diseño Minecraft			
Codificacion	Pruebas			
	Cracion de			
	Presentacion final			
	Actividades Pendientes			
	Entregas			

Fig.7 cronograma de actividades excel

V. ALCANCE

El proyecto va a tener como finalidad comprender todos los conceptos de la implementación de colisión respecto a cubos, haciendo uso de todas las funcionalidades de la librería de Three.js, creando una emulación de Minecraft con respecto al despliegue gráfico, se proveerá iluminación local y sombreado, distintos tipos de fuentes de luz, efectos creados por la interacción con la luz y efectos de texturizado. Adicionalmente, se ofrecerán técnicas para optimizar el despliegue mediante técnicas de descarte y se agruparán los objetos de los escenarios virtuales usando un grafo de escena.

En la interacción con el usuario, crearemos un módulo que sea principalmente responsable de procesar los eventos de entrada del usuario, dichos eventos estarán asociados a la pulsación de una tecla, al movimiento del ratón, entre otros.

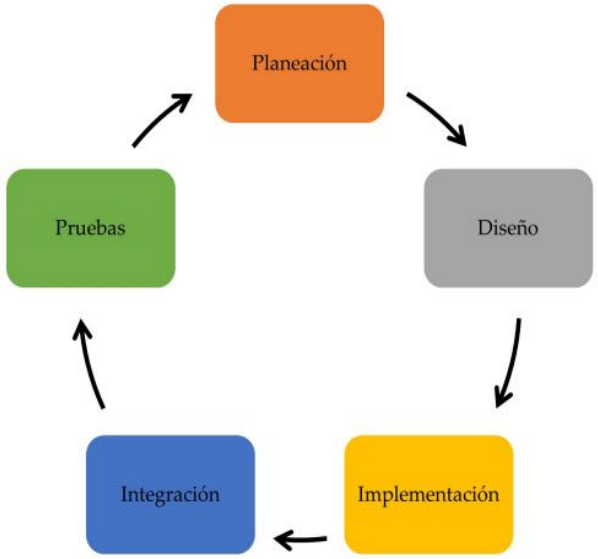

Fig.8 diagrama de alcance, creado en lucidchart

VI. DISEÑO

En esta sección vamos a tomar de ejemplo lo que llegaremos a diseñar del video juego Minecraft tomando imágenes de ejemplo del mismo.Haciendo uso de materiales,textura(pasto,tierra) geometría y efectos(niebla) en three.js.

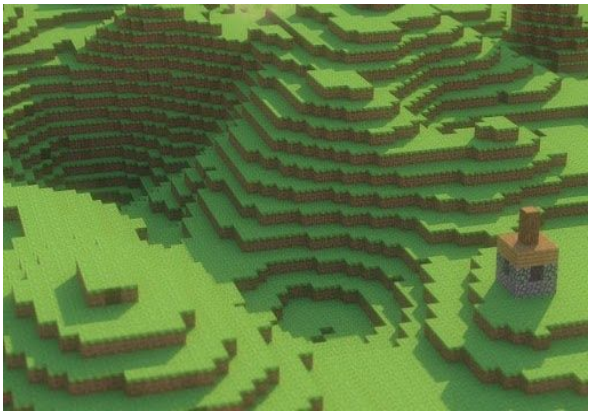

Fig.9 Imagen montañas del Minecraft

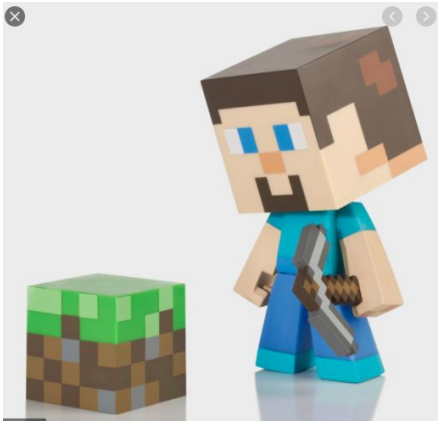

Fig.10 Imagen personaje y cubo de Minecraft

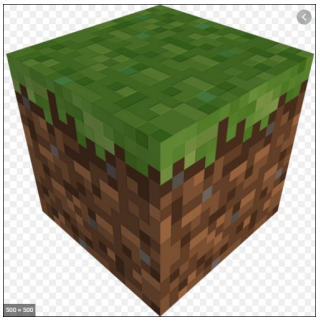

Fig.11 Imagen cubo con material pasto y tierra de Minecraft

Fig.12 Imagen menu de creacion de objetos nuevos Minecraft

Fig.13 Imagen ejemplo de implementación de Minecraft en three.js

VII. BIBLIOGRAFÍA

[1]Mundos tridimensionales,(2017.5 octubre) [en línea]adictosalainformatica.com, disponible :http://adictosalainformatica.com/category/3d/?print=pdf-search

[2]Any Object3D?,¿Alguna forma de obtener un cuadro delimitador desde un objeto Object3D de three.js?,(2013, 4 abril)[en linea],disponible:https://stackoverrun.com/es/q/4203998

[3]Colliders: detección de colisiones en videojuego Unity 3D – Academia Android,(2016,24 abril),[en linea]Academiaandroid.com,disponible:https://academiaandroid.com/colliders-deteccion-colisiones-juego-unity-3d/

[4]Colliders: detección de colisiones en videojuego Unity 3D – Academia Android,(2020, 10 enero),[en linea]Academiaandroid.com,disponible:http://saber.ucv.ve/bitstream/123456789/15018/1/Thesis.pdf

[5]Diseño de juegos 3d para web,(2005, 6 mayo),[en linea]Thefiveplanets.org,disponible: https://thefiveplanets.org/wp-content/uploads/2016/09/thefiveplanets-preview.pdf

[6]Descubre que es Minecraft y por que todo el mundo habla de él,(2016, 2 febrero),[en línea]Thefiveplanets.org,disponible : https://www.educaciontrespuntocero.com/noticias/que-es-minecraft-educacion/