UNIVERSIDAD DE MONTEMORELOS

FACULTAD DE INGENIERÍA Y TECNOLOGÍA

Desarrollo de un entorno de simulación 3D como herramienta de apoyo para la enseñanza de la física

Ingeniero en Sistemas Computacionales

PRESENTA:

Azareel Beristain Cortes

ASESOR:

Ing. Ignacio Cruz Domínguez

MONTEMORELOS, NUEVO LEÓN. 15 DE ABRIL DEL 2016

I. Tabla de contenido

Desarrollo de un entorno de simulación 3D como herramienta de apoyo para la enseñanza de la física

Azareel Beristain Cortes, Asesor: Ignacio Cruz Domínguez

Resumen— ***El uso de nuevas tecnologías para la enseñanza de física es de gran ayuda para el aprendizaje, ya que la enseñanza de esta área es más comprensible por medio de gráficos en movimiento simulando la realidad. Por lo cual este proyecto esta basado en el desarrollo de una herramienta para la simulación del comportamiento de objetos en 3D de la física básica utilizando el motor gráfico Unity como herramienta de desarrollo. Esta herramienta muestra la información realizando los cálculos con las formulas para cada caso presentado. Esto ayudará a estudiantes de del área de física básica a comprender con mayor facilidad, de la misma forma los docentes podrán usar esta herramienta de simulación para la enseñanza a sus alumnos. En este trabajo se realiza la simulación de caída libre, plano inclinado y vectores.***

Palabras claves: Unity, simulación, física, aprendizaje, objetos 3D, gráficos, motor de física, script.

II. Introducción

LOS primeros sistemas de simulación de física que existieron fueron los videojuegos. La física en los videojuegos ha sido fundamental desde los años 70s. Sin embargo, la aplicación de física en los videojuegos estaba limitada por la falta de hardware capaz de realizar extensos cálculos matemáticos. La simulación educativa puede ser definida como el aprendizaje, a través de la interacción de algún modelo basado en algún fenómeno. Una simulación educativa tiene como objetivo favorecer el aprendizaje y desarrollar las habilidades implicadas en la investigación de un fenómeno de naturaleza física o social. Proporcionan un entorno de aprendizaje abierto y altamente interactivo basado en modelos reales [1].

A. Antecedentes

La enseñanza de la física se ha basado tradicionalmente en la visión del profesor sobre el contenido y la percepción del estudiante. La enseñanza tradicional de la Física tiene como principales características que su enseñanza y aprendizaje están orientados hacia el conocimiento y no hacia el proceso de aprendizaje [2].

Los alumnos tienen la posibilidad de complementar otras formas de aprendizaje utilizadas en el aula gracias a l uso de las TIC en clase, que pueden mejorar la comprensión de conceptos difíciles o imposibles de observar a simple vista en los laboratorios escolares [3].

B. Definición del problema

Según la autora Elizondo Treviño [2], en los últimos años se ha detectado que los alumnos de los colegios presentan dificultades en la comprensión de conceptos implícitos en los enunciados de problemas de física, por lo tanto existen dificultades en el proceso de enseñanza-aprendizaje de física. Las dificultades se presenta por su formalismo, en la mayoría de los alumnos constituye un obstáculo la dificultad para interpretar graficas y poder relacionar los datos con hechos reales [17].

En este trabajo nos enfocaremos en el desarrollo de simulación del comportamiento de la física de los siguientes temas: caída libre, plano inclinado (máquina simple) y vectores.

C. Justificación

El uso de simuladores predispone al estudiante a la experimentación. Esto fundamenta la idea de aprender explorando, en el proceso de aprendizaje [4]. Uno de los principales desafíos de los simuladores gráficos es lograr realismo e inmersión. Además de la calidad de la visualización de la escena; la mayoría de las aplicaciones virtuales actuales incluyen simulación de distintos fenómenos físicos. Estas simulaciones han evolucionado al punto donde la mayoría de los objetos en el mundo virtual pueden ser manipulados, reaccionando de forma realista de acuerdo con fuerza aplicada, fricción y elasticidad. El agregado de comportamiento físico en

aplicaciones de escenarios virtuales permite al usuario desempeñarse en un ambiente similar al mundo real. Un ejemplo claro de esto son los videojuegos [5].

Con el desarrollo de este proyecto de simulación es posible beneficiar a los alumnos que tienen dificultades de comprensión, haciendo uso de este simulador pueden tener mayores ventajas al poder interactuar y observar el comportamiento de los cuerpos a través de un mundo virtual.

D. Objetivos

1) Objetivos generales:

a) Crear un software simulador de física en 3D para el apoyo en la enseñanza.

2) Objetivos específicos:

a) Implementar las simulaciones de caída libre, plano inclinado y vectores.
b) Experimentación con el motor de física de Unity.
c) Implementar interactividad de la herramienta de simulación con los usuarios.

E. Hipótesis

¿Será posible que los resultados obtenidos por el simulador en comparación con los resultados teóricos son aproximados?

El uso de simuladores puede ser usados con independencia del estilo metodológico del profesor, esto permitiría que ayuden ser empleados a modo de proactiva virtual, se cree que pueden ser usados como parte de estrategias docentes más elaboradas y eficaces [6]. Lo cual impulsaría a un aprendizaje más rápido y con una mejor comprensión.

III. FUNDAMENTOS TEÓRICOS

A. Marco teórico.

El uso de tecnología en la enseñanza y aprendizaje han sido muy importantes en los últimos años, ya que permite comprender con mayor facilidad por medios de los gráficos en movimiento que simulan la realidad. Los simuladores en realidad se utilizan en varios ámbitos para el aprendizaje por lo que en este proyecto se pretende crear un simulador de física.

Unity es un entorno de gráficos 2D y 3D para sistemas operativos de Windows y Mac que viene empaquetado como una herramienta para la creación de videojuegos, aplicaciones interactivas, visualización y animaciones en 3D en tiempo real. Unity utiliza Physx como motor de física avanzado que se muestra como una interfaz simple y fácil de usar en el desarrollo de simuladores sin tener que trabajar en exceso con programación [7].

La animación basada en la física es utilizada para crear animaciones con objetos que parecen realistas y muy dinámicos. Unity cuenta con un sistema de física que incluyen componentes y otros datos que se pueden conectar con los objetos para hacer que se comporten de manera realista. Entre los más importantes podemos encontrar: gravedad, torsión y flexión [8].

Generalmente las animaciones creadas a través de código son la mejor forma de hacerlo. Los scripts se adjuntan al objeto que se va animar, de esa forma los cuerpos obtienen la interactividad que se le quiere dar. El comportamiento de un objeto es controlado por los componentes que se les unen a ellos, Unity permite crear sus propios componentes utilizando scripts. El uso de lenguaje de programación C# es muy conocido y utilizado para desarrollo en Windows y en desarrollo web [9].

Blender es un software libre para la creación de modelos en 3D, es compatible con la totalidad de modelos 3D, animación, simulación, renderizado, composición y seguimiento de movimiento, incluso con la edición de videos y la creación de videojuegos. Blender es multiplataforma y funciona en los sistemas operativos Linux, Windows y Macintosh. Su interfaz utiliza OpenGL para proporcionar una experiencia coherente [10].

Simulación es una técnica numérica que mediante la modelación de sistemas que permite imitar el comportamiento de las variables y su comporta. Un simulador reproduce el comportamiento de un sistema en ciertas condiciones, suelen combinar partes mecánicas o electrónicas y partes virtuales que le ayudan a simular la realidad, por lo tanto estos pueden utilizarse para el ámbito profesional o como un instrumento de entretenimiento. Gracias a los simuladores los alumnos pueden entrenarse hasta adquirir experiencia y el conocimiento necesario para desempeñarse profesionalmente [11].

Realidad Virtual se define como un sistema informático que genera en tiempo real representaciones de la realidad que de hecho no son más que ilusiones ya que se trata de una realidad perceptiva sin ningún soporte

físico y que únicamente se da en el interior de los ordenadores.
La simulación que hace la realidad virtual se puede referir a escenas virtuales, creando un mundo virtual que sólo existe en el ordenador de lugares u objetos que existen en la realidad. También permite capturar la voluntad implícita del usuario en sus movimientos naturales proyectándolos en el mundo virtual que se está generando, proyectando en el mundo virtual movimientos reales. Además, también nos permite hundirnos completamente en un mundo virtual, desconectando los sentidos completamente de la realidad teniendo la sensación la persona que está dentro de que la realidad corresponde en el mundo virtual [12].

La realidad virtual comprende la interface de hombre-máquina que permite al usuario sumergirse en una simulación gráfica 3D generada por ordenadores que permiten interactuar en tiempo real, desde una perspectiva centrada en el usuario [13].

Caída libre es el movimiento de los cuerpos por la acción de su propio peso ver figura 1, es una forma de rectilíneo uniformemente acelerado. La distancia recorrida (d) se mide sobre la vertical y corresponde, por lo tanto, a una altura que se representa (h). La aceleración en los movimientos de caída libre, conocida como aceleración de la gravedad se representa con la letra g y toma un valor aproximado de 9.81 ms2 [14]. La simulación se implementa en el comportamiento de la velocidad respecto al tiempo, ya que la velocidad de un cuerpo en caída libre no es constante.

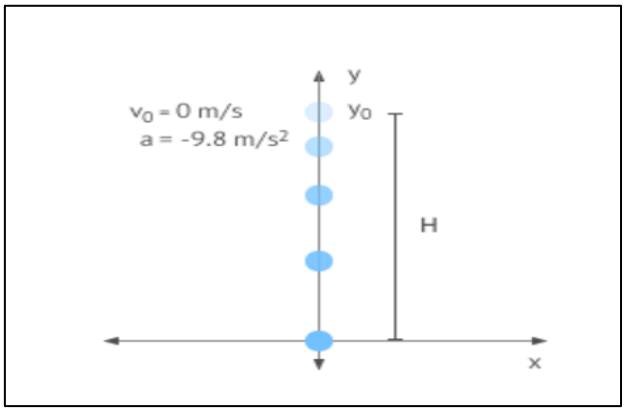

Figura1. Caída libre

Cuando los cuerpos sólidos están en contacto, aparecen fuerzas entre los átomos de los cuerpos que no le permiten a uno a travesar el otro. A estas fuerzas se le llama fuerzas de contacto, unos casos particulares de estas fuerzas aparecen al apoyar un cuerpo sobre un plano inclinado ver figura 2, nombrando esta fuerza se vincula como la fuerza normal [14].

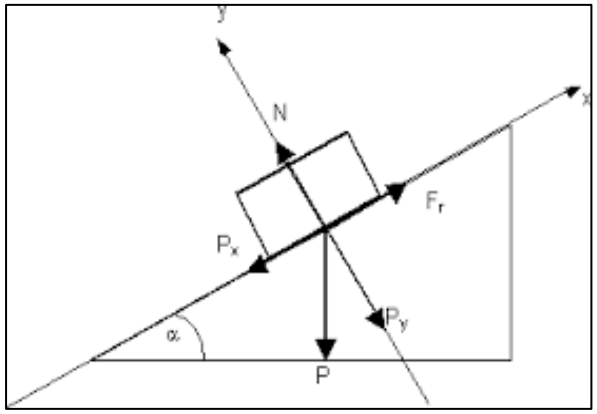

Figura 2. Plano inclinado

Vectores. A partir de la representación de R, como una recta numérica, los elementos (a,b) se asocian con puntos de un plano definido por dos rectas perpendiculares que al mismo tiempo definen un sistema de coordenadas rectangulares donde la intersección representa al origen de coordenadas (0,0) y cada par ordenado (a,b) se asocia con un punto de coordenada a en la recta horizontal (eje X) y la coordenada b en la recta vertical (eje Y) como se aprecia en la figura 3. Análogamente, los puntos (a,b,c) se asocian con puntos en el espacio tridimensional definido con tres rectas mutuamente perpendiculares. Estas rectas forman los ejes del sistema de coordenadas rectangulares (ejes X, Y y Z) [15].

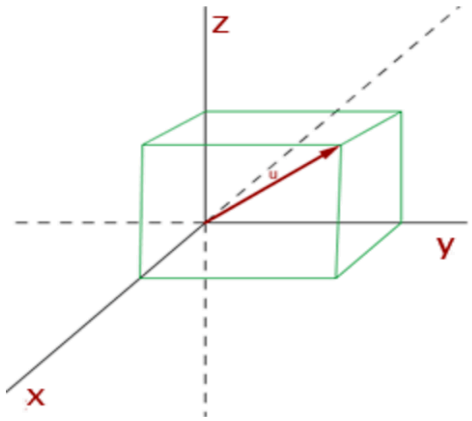

Figura 3. Vector

B. Estado del arte

1) Simulación por ordenador y enseñanza de la Física:

La simulación por ordenador ha sido una de las ideas con más potencial en los medios educativos. La apertura es verdaderamente importante cuando los

contenidos y la interacción presente en el programa son de interés para fines educativos, por lo que se trata de un criterio estrechamente unido a las funciones que el software realiza. Todo programa de simulación es mínimamente abierto por cuanto permite al usuario variar algunos datos y parámetros de control de simulación. Tales programas se denominan Interactive Physics y consiste principalmente en un entorno de simulación que permite realizar diferentes pruebas de simulación dentro de la enseñanza de la física.
Las simulaciones que pueden organizarse por parte del profesor y alumnos son similares, cada simulación consiste en el diseño de uno o varios modelos (construido mediante formas poligonales) de una simulación entre ellos, y la aplicación de unas fuerzas que determinan su movimiento.
Las simulaciones son siempre visuales y el movimiento se ve en la pantalla del computador es una representación de los movimientos reales de los cuerpos. La representación es realista en el sentido de describir la trayectoria de las leyes de la Física. Los factores generales como: fuerza, gravedad, fricción o elasticidad pueden ser variados globalmente y afectar a la trayectoria de los objetos implicados. Mucho del software educativo siguen siendo de calidad baja, y son pocos o mal provistos por los programadores o diseñadores [16].

2) La enseñanza y el aprendizaje de la física y el trabajo colaborativo con el uso de las TIC:

El segundo proyecto se creó para promover en profesores de física, técnicas que promuevan el aprendizaje activo de la física en sus alumnos, se ha desarrollado un seminario utilizando vídeos, animaciones y experimentos. Esta experiencia se enmarca en el proyecto Europeo MOSEM. Participaron veintiún profesores de física de la Región de Murcia. Se analizó el desarrollo del seminario para determinar el efecto que produjo en la formación de los docentes respecto a dos grandes bloques: competencias y uso de tecnologías de la información y comunicación (TIC), expectativas y grado de satisfacción respecto al seminario. Los resultados indican que, con una formación adecuada, el docente haciendo uso de TIC (específicamente, simulaciones de física), podría mejorar el proceso de enseñanza y aprendizaje de la física en secundaria [3].

3) *Phun 2D*:

Simulador de física elaborado por Emil Ernerfeldt, recurso de apoyo para aprender las leyes de la física que permite conocer y manipular las propiedades como; la gravedad, la fricción, los fluidos y los fenómenos de la luz. Fue realizado en la universidad de Umea de apoyo para que los alumnos observen y comprendan la interacción del los cuerpos mediante la aplicación de la física [17].

4) Newton 3D Multimedia Lab for Exploring Physics:

Un mundo virtual 3D que proporciona una forma nueva de aprendizaje de la exploración de la física de la cinemática, y dinámica. El mundo virtual de Newton es la física simulada, permite construir, manipular y analizar los experimentos de forma interactivo, además de poder seleccionar objetos abstractos de los cuerpos geométricos más simples. Se puede ajustar los parámetros físicos como; masa, elasticidad, fricción entre otras más [18].

Pueden configurarse más de una cámara para la captura de puntos de experimentación. Los resultados pueden medirse y ser evaluados. La versión mono usuario tiene un precio de 150 Euros, cuenta con una versión monousuario para estudiantes con un precio de 49 Euros.

5) Algodoo:

Es un software único de simulación 2D de Algoryx Simulación AB . Está diseñado de una manera lúdica, dibujos animados, por lo que es una herramienta para la creación de escenas interactivas. Explorar la física, construir inventos sorprendentes, diseñar juegos frescos o experimentar con Algodoo en sus clases de ciencias. Algodoo anima a los estudiantes y de los niños a usar su propia creatividad, la capacidad y la motivación para construir el conocimiento mientras se divierten. Por lo que es tan entretenido como es educativo. Este simulador es una ayuda para que los niños aprendan y la física de la práctica en el hogar [19].

IV. METODOLOGÍA

A. Herramientas de apoyo para el desarrollo del simulador

Como herramientas de apoyo fundamentales para la elaboración de este proyecto contamos con Unity por las facilidades que nos proporciona con su motor de física, para la elaboración de modelos de objetos usaremos Blender ya que esta herramienta permite la edición y creación de modelos en 3D que pueden ser exportados a Unity, También contamos con Mono Develop para la creación del código de este proyecto, se utilizó C# como

lenguaje de programación por la documentación extensa que existe.

B. Etapas del proyecto

Este proyecto se dividió en las siguientes cuatro etapas: escena de menú, escena de caída libre, escena de plano inclinado, y escena de vectores. Estos se describirá con más detalle en los siguientes párrafos en el cual se mostrarán los pasos que se siguieron en el desarrollo.

C. Pasos del desarrollo

Inicialmente se creó un proyecto en Unity 3D llamado FitFisic ya que Unity tiene la opción 2D y 3D, se crearon cuatro escenas que se requieren para la elaboración de esta herramienta de simulación. Siempre que se crea una nueva escena, ésta contiene una cámara y una luz los cuales son los componentes principales que permiten visualizar todos los componentes que tiene una escena.

1) Escena Menú:

Inicialmente se crearon objetos 3D, en primer lugar se creó un terreno que representa el suelo de la escena, a este terreno se le asignó una textura llamada pasto. También se creó un plano y se le asignó una textura llamada cielo, este plano se escaló y se rotó para colocarlo como fondo de la escena. Para asignar las texturas a los objetos se hizo simplemente arrastrando la textura hacia el objeto, las texturas están en formato PNG y fueron descargadas de internet. estos objetos se crearon en todas las escenas ya que son los objetos necesarios para iniciar con la simulación de un espacio 3D.

En esta escena se crearon tres cubos y una esfera, al primer cubo se le asignó una textura metal y se escaló para crear un plano inclinado o también llamado rampa, el segundo cubo se le asignó una textura madera, al tercer cubo se le agregó la textura madera además de una animación para que tuviera un movimiento en tiempo de ejecución, a la esfera se le asignó una textura balón y se le agregó una animación llamada balon1. Para crear una animación primero se seleccionó el objeto que se quiere animar y después en la parte superior en Windows se seleccionó Animation [20], esto funciona asignando cierto tiempo para cada movimiento del objeto como se puede observar en la figura 4.

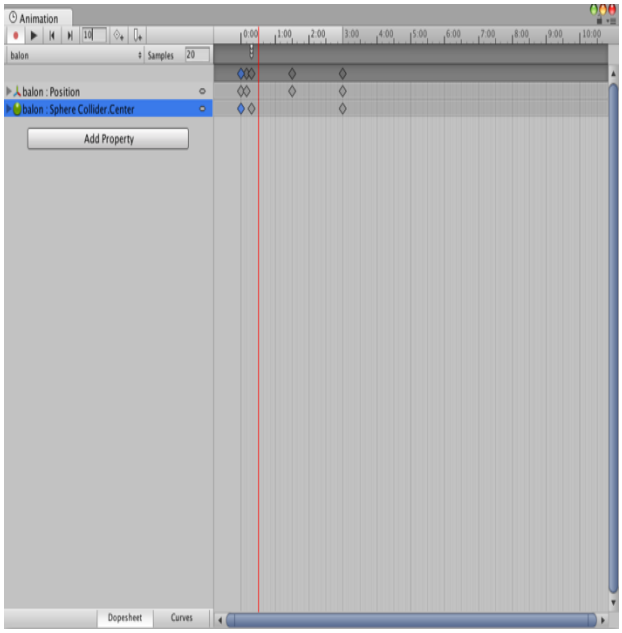

Figura 4. Ventana de animación

Para esta escena se creó un script en C# llamado botonesMenu el cual se arrastro a la cámara principal, este script crea los botones que cargan las demás escenas que se han creado en el proyecto, cada botón tiene un texto que indica a la escena que cargará al dar clic sobre el botón.

```
Botón Caída libre;
Botón Plano inclinado;
Botón Vectores;

 interfaz grafica de usuario{
si (Caída libre){
   carga la escena de caída libre
  }

si (Plano inclinado){
   carga la escena de plano inclinado
  }

si (Vectores){
   carga la escena de vectores
  }
}
```

Para que las escenas se carguen al momento de dar clic en los botones del menú deben configurarse desde Buil Setting agregando las escenas existentes desde Add Current como se muestra en la figura 5.

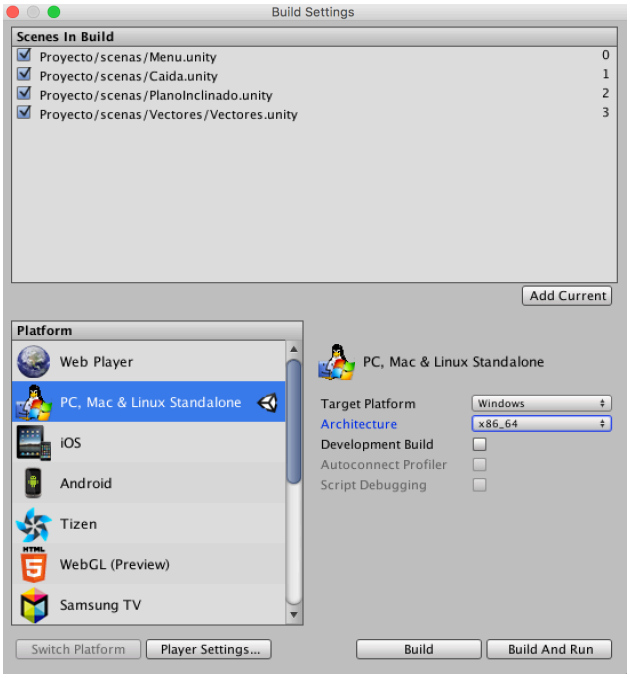

Figura 5. Configuración de las escenas que se cargan

Finalmente, la escena menú se visualiza como se muestra en la figura 6, el cual contiene tres botones y los objetos que tienen movimiento durante la ejecución.

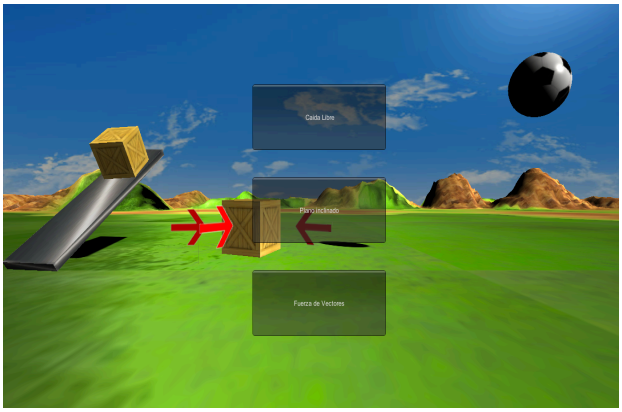

Figura 6. Escena de menú

2) Escena de Caída Libre:

Para esta escena además de tener una cámara principal, una luz, un terreno y un plano nombrado fondo, se creó un cubo que es el objeto que se utiliza para representar la caída libre de un cuerpo a este objeto se le agregó una textura y el componente cuerpo rígido (Rigidbody) el cual nos permite hacer uso de la función de gravedad que se aplica para la simulación de caída libre. Se creó una segunda cámara como hijo del objeto cubo con la finalidad de que esta cámara siga al cubo y muestre donde se encuentra cuando se le asigna una altura, esta cámara se activa y se desactiva dependiendo la posición del objeto, también se creó un Text Mesh como hijo del objeto cubo el cual muestra la altura actual del objeto.

Se creó un objeto llamado plano cubo como padre del objeto cubo, el cual su función principal es llevar el objeto cubo a la posición de altura que el usuario introduce, a este objeto se le desactivo el Mesh Collaider con la finalidad de que el objeto cubo pueda bajar desde la altura que se encuentre y no exista colisión entre ellos, también se desactivo el Mesh Render para que el objeto no sea visible en la escena. Estos componentes se desactivaron directamente desde el Inspector.

Se creó un Script llamado panelText el cual se encarga de mostrar la información acerca de la posición en (y) del objeto que se deja caer, así como el tiempo que transcurre para poder hacer los cálculos en tiempo definidos en el Script.

```
PanelText

altura es igual a string vacío;

objeto cubo;
objeto planoCubo;
objeto textMesh

tiempo = t;
altura = h;
velocidad = v;
gravedad = g;

posy = y;
g = 9.8086;

input textFiel;
h =  Entrada texto;
convierte altura a tipo float;

posiciona en (y) el objeto planoCubo en la altura actual
agrega la altura a textMesh para mostrarlo;

tiempo;
posición y;
velocidad;
se crea caja de texto Y teórico;
si la posición cubo == posición planoCubo{
    t = 0;
} De lo contrario {
    t ++;
}
si (h != “”){

// se repite esta condición para tiempo 1,2,3,4,5
```

```
    si (t =< 1{
        texto t = 1;
        texto y = posy
        tiempo1;
        texto v =(t*g);
        texto hF = (0,5*g.t^2);
    }
}
```

Como se puede observar en la figura 7, esta herramienta muestra la posición del objeto que se deja caer, así como la velocidad respecto al tiempo.

La fórmula que se utilizó para calcular la altura del objeto respecto al tiempo es la siguiente $h = \frac{1}{2}\, g.t^2$. Esta fórmula se aplico dentro de una caja de texto para mostrar los valores con los tiempos en segundos, y para mostrar la posición que se encuentra el objeto. El resultado se obtuvo restando la distancia recorrida a la altura inicial.

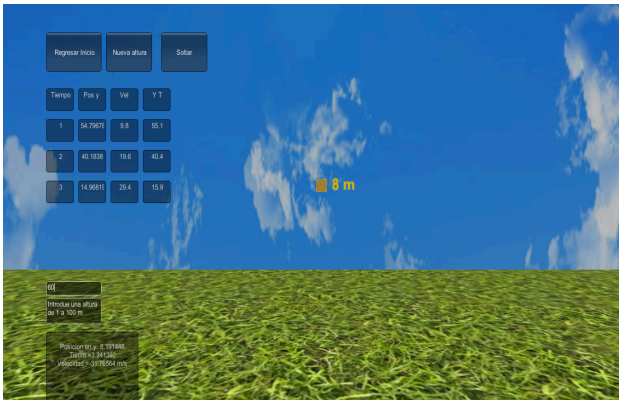

Figura 7. Escena caída libre.

Finalmente, se creó un script llamado “trigger”, la finalidad principal del script fue para detener la ejecución de la escena cuando el terreno y el cubo se colisionan, esto permite al usuario observar el tiempo que tardo en caer el objeto desde su posición inicial a su posición final, así mismo para observar la velocidad final del cubo.

3) Escena Plano inclinado:

Inicialmente se crearon tres cubos, el primer cubo se le nombró ¨pivote¨, el segundo cubo ¨plano inclinado¨ y el tercer cubo se le nombró ¨caja¨. Para esta escena se importaron dos objetos que fueron descargados de la página oficial Blender el cual cuanta con una gran variedad de objetos en 3D gratuitos, el primer objeto que se importó fue un modelo llamado contenedor, y el segundo objeto fue un modelo humanoide que se muestra en la figura 8. Cabe mencionar que se seleccionó este modelo porque contiene una animación llamada “andar”, el cual permite la simulación del humanoide caminando.

Figura 8. Modelo humanoide.

El objeto padre (pivote) permite la inclinación del objeto hijo (plano inclinado), esto permite que la rotación que ocurra en el objeto padre será afectado en el objeto hijo. La inclinación del plano es manipulado desde un deslizador que permite una inclinación de 0º a 90º, este deslizador se creó desde un script llamado “rotar”, se utilizó la función de rotación. Para el objeto contenedor fue necesario escalar la altura (y) con respecto a los grados de inclinación del plano, la finalidad de esta escala fue para que el contenedor estuviera a la altura de la parte final del plano inclinado y que el objeto que sea empujado suba hasta el contenedor.

```
Clase rotar;
objeto pivote;
objeto contenedor;

grados = g;

función OnGUI{
g = deslizador inclinación (0 a 90);
}

función Update{
    pivote.rotar(grados);
    //escala la altura del contenedor en el eje y
    contenedor.escalaLocal (x, y*(grados*0.1), z);
}
```

Como se muestras en la figura 9, también cuenta con un segundo deslizador llamado “Fuerza”, éste permite al usuario interactuar con la fuerza que será aplicada en el

humanoide el cual empujara al objeto “Caja”. Para esta parte se creó un script llamado “Empujar”.

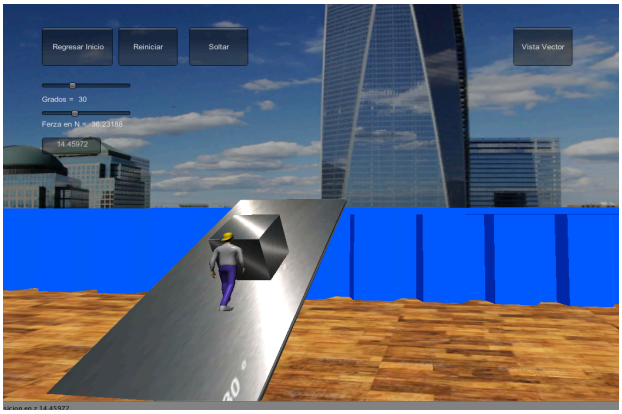

Figura 9. Cámara principal de plano inclinado

Se creó un objeto capsula como padre del objeto humanoide, el cual se le agregó un script llamado “empujar” , este script aplica una fuerza a través del eje “z” que es la profundidad de la escena, también se le agregó el componente de cuerpo rígido para hacer uso de la gravedad, para este componente se activaron las casillas de congelar rotación para (x,y,z), esto se realizó directamente en el inspector. La finalidad de esto fue para que el humanoide no sea afectado al momento de la inclinación, por lo contrario este caerá hacia atrás ni hacia los lados. El objeto humanoide se le agregó un una caja de colisión para que no atraviese el plano inclinado y tampoco para que el objeto cubo pueda ser empujado al momento de aplicar la fuerza.

```
Clase empujar;
Objeto humanoide;

Fuerza = 0;

Cuerpo rígido rb;

función OnGUI{
    Fuerza = deslizador inclinación (0 a 100);
}
[14] [6]
Función fixedUpdate {
    Humanoide. añadir fuerza (hacia delante * Fuerza);
}
```

Esta escena cuenta con tres botones en la parte superior los cuales se crearon desde código, el primer botón regresa al menú principal del proyecto, el segundo botón vuelve a cargar la escena después de alguna ejecución realizada, el tercer botón pausa la ejecución del simulador, con la finalidad de que el usuario pueda ver detalladamente los datos cuando sea necesario.

En la figura 10, se muestra la vista desde la cámara secundaria que se creó. Esto se realizó con la finalidad de que el usuario pueda ver el movimiento de los objetos desde otra perspectiva se pueda observar el movimiento de los objetos sobre el plano inclinado, además se puede visualizar los vectores de fuerzas.

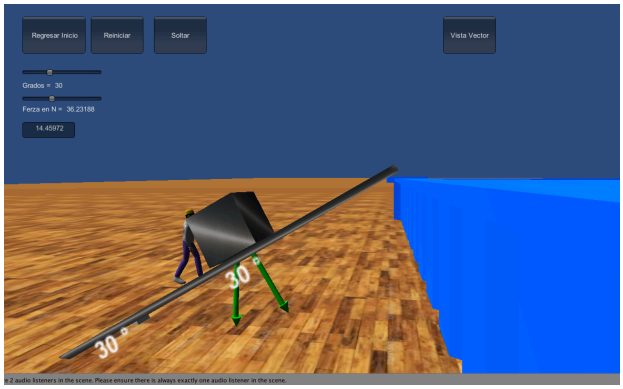

Figura 10. Cámara secundaria del plano inclinado

4) Escena de Vectores:

En Blender se creó un objeto llamado vector, este objeto está compuesto por dos objetos, el primer objeto es un cilindro y el segundo es un cono que representa la punta del vector, el objeto vector se exporto a Unity con formato fbx, esto se realizó con la finalidad de que el objeto vector tuviera el pivote en el lugar que se quería obtener para realizar las rotaciones necesarias en esta etapa.

El objeto vector se agregó a esta escena y se duplico las veces necesarias que se ocuparon para la construcción de este simulador. Se crearon materiales con diferentes colores para poder identificar los vectores. Como se puede observar en la figura 11, el vector resultante se le asignó un material color rojo para identificarlo de los demás vectores.

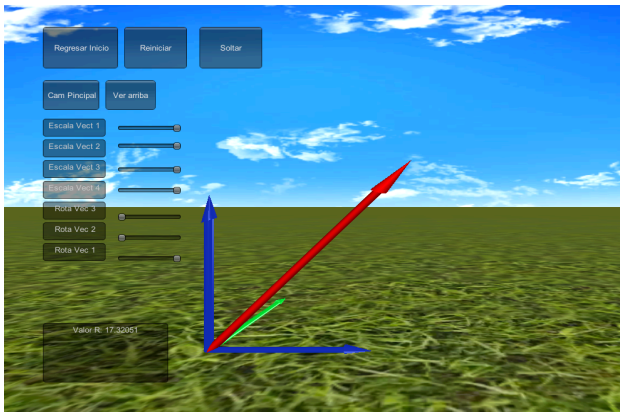

Figura 11. Vectores.

Para el vector resultante se utilizó la siguiente fórmula $|V| = \sqrt{a^2} + a^2 + a^2$, el cual se realizó el cálculo desde código. Como se muestra en la figura [11], la longitud de los vectores se manipulan desde los deslizadores que se les asignó un valor de 10 unidades.

```
Clase vector;

Objeto vector1;
Objeto vector2;
Objeto vector3;
Objeto vectorR;

Función Update{
Suma1 = vector1 al cuadrado;
Suma2 = vector2 al cuadrado;
Suma3 = vector3 al cuadrado;
SumaR = raíz cuadrada(suma1 + suma2 + suma3);

vectorR. escala(sumaR);

}
```

D. Resultados

Los resultados que se obtuvieron en la escena de caída libre, fueron muy similares, al momento de hacer pruebas con el simulador y comparar los resultados que se obtuvieron aplicando las ecuaciones físicas para el caso de la caída libre como se puede ver en la tabla 1.

Tabla 1. Resultados de caída libre.

Tiempo	Posición Y	Velocidad	Altura por Formula
1	94.9	9.8	95.1
2	80.1	19.6	80.4
3	54.9	29.4	55.9
4	21.1	39.2	21.6

En la simulación de plano inclinado se obtuvieron los resultados que se muestran en la tabla 2, como se puede observar las pruebas que se realizaron fueron en diferentes grados en el ángulo de inclinación del plano inclinado. Se realizaron los cálculos por medio de la fórmula $Fa = g.sen \propto$, el cual como resultado indica la fuerza que se requiere para mover el objeto con un peso de 80 unidades.

Tabla 2. Resultados de plano inclinado.

Ángulo	Py	Fa
15º	80	20.7
20º	80	27.3
25º	80	33.8
30º	80	40
35º	80	45.8

Para el caso de la simulación de vectores se muestran los datos en la tabla [3], se hicieron las siguientes pruebas para comparar los resultados que nos muestra a través de una caja de texto. Estos datos coinciden con los resultados que nos arrojan el simulador para cada presentado en la tabla.

Tabla 3. Resultados de vectores

	caso 1	caso 2	caso 3
Vector 1	5	4	10
vector 2	7	6	10
vector 3	9	8	10
vector R	12.44	10.77	17.32

E. Discusión

Para el caso de caída libre la simulación es muy similar a la realidad en caída libre, ya que haciendo los cálculos con las ecuaciones, se realizó una comparación con el tiempo que tarda en recorrer la distancia desde la posición inicial hasta llegar a la posición final, y el tiempo fue muy cercano al del resultado con los cálculos. Tomando en cuenta cada unidad en el eje (y) como una unidad de metro.

Nuestro software de simulación presenta algunas ventajas comparado con el simulador Phun 2D ya que éste cuenta con profundidad en la escena y los movimientos que se realizan son más realistas. En comparación con el simulador de física Newton 3D Multimedia Lab for Exploring Physics también pueden ajustarse algunos valores para que el usuario pueda interactuar con nuestra herramienta, aunque Lab for Exploring Physics es un laboratorio virtual de física muy extenso éste tiene un costo elevado para poder hacer uso de ella, sin embargo nuestro simulador puede ser usado por cualquier usuario ya que no cuenta con ningún costo para su uso.

Una de las principales limitaciones de este proyecto fue el tiempo para el desarrollo, también cabe mencionar que no se contó con un equipo para el desarrollo de contenido. En este proyecto se abarcan tres temas que

contienen lo básico para la simulación. Como futuros trabajos se pretende ampliar los temas de física, también se planea implementar componentes como fricción,

F. Conclusión

Se logró crear una herramienta de simulación usando Unity como plataforma de desarrollo. Los datos de altura, velocidad, fuerza y magnitud que se obtuvieron al momento de la simulación tienen aproximados a los datos que se obtienen a través de las fórmulas teóricas de la física. Sin embargo nuestra herramienta de simulación puede ser utilizada para el apoyo en el aprendizaje y la enseñanza en alumnos que requieran de una representación más realista de la física para los temas mencionados en este proyecto.

V. Referencias

[1] J. Valverde, "Aprendizaje de la Historia y Simulación Educativa," *Tejuelo*, vol. 9, pp. 83-89, 2010.

[2] M. S Elizondo Treviño, "Dificultades en el proceso enseñanza aprendizaje de la Física," Presencia Universitaria, pp. 70-77, Junio 2013. [Online]. http://eprints.uanl.mx/3368/1/Dificultades_en_el_proceso_ense%C3%B1anza_aprendizaje_de_la_F%C3%ADsica.pdf

[3] J. Serrano y M. Paz, "La enseñanza y el aprendizaje de la física y el trabajo colaborativo con el uso de las TIC," *REALTEC*, vol. 11, pp. 95-107, 2012.

[4] M. Lazo y M. Vénere C. García Bauza, "INCORPORACIÓN DE COMPORTAMIENTO FÍSICO EN MOTORES GRÁFICOS ," *Asociación Argentina de Mecánica Computacional*, vol. XXVII, pp. 3023-3039, Noviembre 2008.

[5] M. Lazo y M. Vénere C. García Bauza, "INCORPORACIÓN DE COMPORTAMIENTO FÍSICO EN MOTORES GRÁFICOS ," *Asociación Argentina de Mecánica Computacional*, vol. XXVII, pp. 3023-3039, Noviembre 2008.

[6] A. Garcia Berneto y M.R. Gil Martin, "Entornos constructivistas de aprendizaje basados en simulaciones informáticas.," *Electrónica de Enseñanza de las ciencias*, vol. %, no. 2, 2006.

[7] N. Arrioja, Users Unity, 1st ed. Buenos Aires: Fox Andina, 2013.

[8] A. Thorn, *Animation Fundamentals in Unity Animation Essentials*. Mumbai, India: Packt Publishing, 2015.

[9] T. Norton, Learning C# by Developing Game with Unity 3D. Birmingham, West Midlands: Packt Publishing, 2013, vol. 1.

[10] Ton Roosendaal. Blender. [Online]. https://www.blender.org/

[11] I. Aguilar Juárez y J. R. Heredia Alonso, "Simuladores y laboratorios virtuales para Ingeniería en Computación," Revista Iberoamericana para la Investigación y el Desarrollo Educativo, Enero-Junio 2013.

[12] Facultat de Informàtica de Barcelona. Retro Informática. [Online]. http://www.fib.upc.edu/retro-informatica/avui/realitatvirtual.html

[13] F. J Perez Martínez, "Presente y Futuro de la Tecnología de la Realidad Virtual," Creatividad y Sociedad, vol. XVI, Marzo 2010.

[14] y C. Vuille A. Raymond, Fundamentos de la Física. México DF.: Cengage Learning Editores, S.A. de C.V., 2012.

[15] F. Walter Mora, "Vectores, Rectas y Planos," Revista digital, pp. 9, 10, agosto 2014.

[16] M. Belmonte y JRodriguez, "Simulación por ordenador y enseñanza de la física," Comunicación, Lenguaje y Educación, pp. 63-72, 1995.

[17] H. E. Medellín y J. R. Martínez G. Ortega, "Influencia en el aprendizaje de los alumnos usando simuladores de física," in *Iopciennce*, vol. 4, San Luis Potosí, 2010.

[18] (2010) newtonlab. [Online]. http://www.newtonlab.com/English/newton/

[19] (2016) Algodoo. [Online]. http://www.algodoo.com

[20] Thorn Alan, Unity Animation Essentials, Nadeem N. Bagban, Ed.: Packt Publishing, June 2015 , vol. 1.