# UNIVERSIDAD MAYOR DE SAN ANDRÉS
## FACULTAD DE CIENCIAS PURAS Y NATURALES
## CARRERA DE INFORMÁTICA


# PROYECTO DE GRADO

SIMULADOR 3D DE CINEMÁTICA PARA ESTUDIANTE DE NIVEL SECUNDARIO.
CASO: UNIDAD EDUCATIVA REP. DE ITALIA

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS

**POSTULANTE:** Bladimir Laruta Mollo

**TUTOR METODOLÓGICO:** Ph.D. Yohoni Cuenca Sarzuri

**REVISOR:** Lic. German Huanca Ticona

LA PAZ – BOLIVIA
2016

# DEDICATORIA

El presente trabajo va dedicado a mi padre: Leocadio Laruta, quien me apoyo durante toda mi trayectoria Universitaria, lo cual me permitió llegar hasta esta instancia de mi formación académica ya que él siempre ha estado presente para brindarme un apoyo moral y sicológicamente, gracias por estar en esos momentos difíciles brindándome su amor, paciencia y comprensión.

# AGRADECIMIENTOS

Primeramente agradezco a la Carrera de informática de la Universidad Mayor de San Andrés por haberme sido para te de mi formación científica como también a los diferentes docente que brindaron sus conocimientos y su apoyo para seguir adelante.

Agradezco a mi tutor metodológico Ph.D Yohoni Cuenca Sarzuri por su dedicación y guía para la elaboración de este trabajo y su valiosa orientación. Agradezco también al Lic. German Huanca Ticona por haber aceptado ser mi asesor y tomarse el tiempo de revisar este trabajo y por brindarme su apoyo.

A la directora de la Unidad Educativa Rep. De Italia Lic. Silvia a. Minaya Flores, por su confianza y aceptar que pueda desarrollar mi proyecto de grado es su establecimiento educativo y ser puente para la culminación de mi carrera universitario.

Al profesor de la asignatura de Física de dicha Unidad educativa, Ing. Felipe Pérez Mamani, por brindarme un apoyo incondicional y brindarme la información necesaria para la culminación de este trabajo.

Por supuesta agradezco a mi padre porque sin su apoyo y cariño durante toda mi vida, no habría logrado llegar hasta este momento de mi vida. Por ultimo agradezco a una persona especial muchas gracias Sheila Nigañez por tu apoyo constante.

A todas la personas que fueron parte de mi vida decirles gracias de todo corazón.

Bladimir Laruta Mollo

# RESUMEN

El uso de las nuevas tecnologías es indispensable para la formación de cualquier persona. Debido a la importancia que tiene la formación del tema de Cinemática del área de Física. La cual conlleva a un mejor entendimiento de los distintos tipos de movimientos que existe en la naturaleza, los jóvenes estudiantes deberán usar estas nuevas tecnologías para poder tener una mejor educación.

Una de las tecnologías crecientes es el uso de imágenes tridimensionales, ya que su uso es más didáctico e interactivo las cual nos expresa imágenes más reales a la realidad la cual ofrece contenidos en tres dimensiones. El presente proyecto de grado "*Simulador 3D de cinemática para estudiante de nivel secundario. Caso: unidad educativa Rep. De Italia*", describe la creación de una aplicación que sirve como herramienta educativa accesible y adecuada a las necesidades requeridas para el proceso de enseñanza y aprendizaje. Mediante modelos 3D la cual llevara a una mejor percepción del tema, que permitirá realizar pruebas o verificación de lo avanzado en el tema de Cinemática de la materia de física. Obteniendo la atención de los estudiantes por el uso de imágenes 3D a la hora de la enseñanza.

La metodología a seguir fue MEISE, metodología de ingeniería de software educativo la cual nos brinda los pasos a seguir para poder desarrollar aplicaciones educativas, conjuntamente se hizo uso de las herramientas: Unity 3D, Blender y el lenguaje de programación C#.

Una vez desarrollado la aplicación, se realizaron pruebas con las que se pudieron determinar la calidad y los beneficios que esta brinda, en un entorno social tecnológico y económico. Para los profesores y estudiantes. Y para la unidad educativa en general.

El uso de la aplicación por parte de profesores y estudiantes fue de alto agrado y recibió una excelente aceptación. Además los resultados demuestran el éxito de la implementación de nuevas tecnologías en el campo educativo y su beneficio en el proceso de enseñanza - aprendizaje.

**Palabras claves:**

Imágenes 3D, Cinemática, Enseñanza – Aprendizaje, simulador

# ABSTRACT

The use of new technologies is essential for the formation of people. Because of the importance of the formation, of the topic Kinematics of Physics. Which leads to a better understanding of the different types of movements that exist in nature, young students should use these new technologies to be better educated.

One of the growing technology is the use of three-dimensional images, as their use is more educational and interactive which expresses us the real reality which offers three-dimensional content images. This draft grade "*simulator of kinematic 3D for secondary student. Case: unit educational Rep. Italy* "describes the creation of an application that serves as accessible and appropriate to the needs required for the process of teaching and learning educational tool. Using 3D models which lead to a better perception of the subject, to perform testing or verification of the lateness in the subject matter of Kinematics of physics. Getting the attention of students through the use of 3D images when teaching.

The methodology followed was MEISE, engineering methodology educational software which gives us the steps to develop educational applications, jointly made use of the tools: Unity 3D, Blender and the programming language C #.

Once we developed the application, testing which could determine the quality and the benefits it provides, in a technological and economic social environment were made. For teachers and students. And for the educational unit in general.

The use of the application by teachers and students was high pleased and received an excellent reception. Furthermore, the results demonstrate the successful implementation of new technologies in education and its benefit in the process of teaching - learning

**Key Words:**

3D images, teaching – learning, Kinematics, simulator

# ÍNDICE

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# CAPÍTULO I
# MARCO PRELIMINAR

## 1.1. INTRODUCCIÓN

Hoy en día las Tecnologías de información y comunicación TIC. Facilitan muchas de las actividades que realizamos en nuestra vida cotidiana, haciendo más eficientes y de mejor calidad todas las tareas que realizamos. Ahora mismo es mucho más fácil acceder a la información debido al gran avance de las tecnologías lo que es ideal para las diferentes actividades que la sociedad realiza. Es por este motivo por la cual muchas de las instituciones hacen uso de las nuevas Tecnologías de Información y Comunicación para el desempeño de sus actividades para el manejo de la información.

El uso de las TIC en los diferentes niveles: Ya sea empresarial, institucional, etc. trae un gran beneficio. En el ámbito educativo puede tener un impacto muy significativo en el desarrollo del aprendizaje. Además de ser un gran apoyo para los educadores facilitando los desempeños que realiza. Pero en la realidad que vivimos, no tomamos la importancia que esta se merece.

El área educativa de la ciudad de La Paz presenta ciertos problemas que muchas veces acarean en el fracaso de los estudiantes. Algunos de estos problemas se debe: a la malla curricular, la falta de recursos, la masificación de las aulas, problemas familiares, el desinterés de los estudiantes o en otros casos. Con todos estos problemas cualquier herramienta que nos ayude a mejorar el proceso de enseñanza y aprendizaje marcara una gran diferencia, en especial tomando en cuenta el nivel secundario ya que en esta etapa los estudiantes tienden a prestar menos interés al proceso de aprendizaje ya sea por problemas familiares o la incapacidad que uno mismo siente, ya que se les hace más difícil comprender todo lo que el educador les inculca, más si se trata del área de "Física" ya que muchas veces no se puede comprender todo lo que el docente explica mediante gráficos en dos dimensiones, 2D. Es por tal motivo que en esta etapa se debe motivar a los estudiantes a que presten más atención a la educación que se les brinda.

Por tal motivo para que la educación mejore lo primero que deberá realizarse es, captar la atención de los estudiantes, lo cual se lo conseguiría realizando imágenes reales que se observa en la naturaleza con dibujos en tres dimensiones 3D, en la explicación del tema

impartido por el docente de física. Lo cual también presentaría un problema ya que realizar dibujos en 3D es mucho más complicado y moroso de realizar, por tal motivo disminuiría la eficiencia del educador y no cumpliría con la curricular asignada.

Así el presente trabajo va relacionado a brindar una herramienta para el proceso de enseñan y aprendizaje de los estudiantes de nivel secundario de la "*Unidad Educativa Rep. De Italia*". Que tiene como propósito la creación de una herramienta educativa "Aplicación" específicamente para el tema de "cinemática", que sea accesible. Haciendo uso de la programación grafica en 3D para que los estudiantes puedan adquirir fácilmente todo lo que se les enseña, por lo cual se brindaría una educación de calidad.

## 1.2.ANTECEDENTES

### 1.2.1. INSTITUCIONALES

La Unidad Educativa Rep. De Italia empezó sus actividades en la hacienda del famoso escrito Alcides Arguedas la cual estaba situada en la Av. Kollasuyo en la zona Alto Mariscal Santa Cruz de la ciudad de La Paz. Pero antes que la Unidad Educativa Rep. De Italia existiera, había una escuela llamada "Luis Uría" que realizaba sus actividades en dicha hacienda.

En 1961 fue su último año de funcionamiento de la escuela Luis Uría en dicha hacienda, como directora estaba la señora Betty Quiroga, en octubre de ese mismo año fue trasladado a la zona Munaypata. La zona Alto Mariscal Santa Cruz se quedó sin una unidad educativa, ante la falta de esta necesidad los padres de familia de la zona se movilizaron teniendo con representante al señor Hugo Altamirano los cuales estaban preocupado por dicha situación, recurre al Ministerio de Educación, para pedir la creación de una nueva escuela ya que se contaba con los ambientes necesarios con el nombre de: "Alcides Arguedas" pero fue rechazado, porque ya existía una escuela con ese nombre, fue en una segunda ocasión para poner el nombre de Mariscal Andrés de Santa Cruz pero nuevamente fue rechazado, el sénior Hugo Altamirano asistía a misa en la parroquia apostólica Santiago ubicado en la zona de Munaypara, el cual le comento sobre el problema que tenía al padre de dicha parroquia, el padre le sugirió el nombre de su patria natal "Italia". Entonces el consulto con los padres de familia, los cuales gustosamente aceptan el nombre sugerido y con esto se da inicio al funcionamiento de la escuela con el nombre "República de Italia". Para la inauguración de

la escuelita, se invita al embajador de la República de Italia y personajes renombrados de la ciudad de La Paz.

El colegio se funda el 4 de Junio de 1962, a sugerencia de los padres de familia. Se crea con la resolución ministerial Nº 1264 del 11 de diciembre de 1963. La escuela empieza con 500 alumnos y niveles de primero a sexto de primaria, la mayoría de los alumnos provenían de la zona de Alto Tejar.

La escuela empieza su funcionamiento con el profesor Juan Choque Villca, en la primera semana de clases. Ante la ausencia de profesores el señor Hugo Altamirano se puso a pasar clases alternando con juegos y entretenimientos, la segunda semana se presenta la profesora Carmen Vásquez que anteriormente fue la profesora de la escuela Luis Uría retorno con carácter voluntario, a la tercera semana se presentó la profesora Olga Archondo Palacios que anteriormente trabajó en la escuela Luis Uría. La profesora Carmen Vásquez abandona la escuela Italia por orden del supervisor René Silva.

En 1979 la Unidad Educativa Rep. De Italia tenía muchas deficiencias relativas a su capacidad académica ya que había una creciente población estudiantil, teniendo deficiencias estructurales, estaban en trámite la documentación de la escuela desde hace 17 años atrás. La directora del ciclo intermedio declaró al periódico presencia que el local de la escuela Italia no cuenta con servicios adecuados para los alumnos, la falta de mobiliario, un acordeón para las clases de música y además no tenía sanitario dijo además que el edificio no tenía patio. También contaba con laboratorio y biblioteca pero se testifica que se desaparecieron misteriosamente. Luego al pasar desde hace años este problema no fue reclamado por nadie.

Actualmente la Unidad Educativa "República de Italia" tiene su local propio en la zona "Alto Mariscal Santa Cruz", la directora Lic. Silvia Adriana Minaya Flores. Los docentes de la unidad educativa realizan sus actividades en aulas sin contar con laboratorios, ya que solamente cuentan con el laboratorio de computación. Haciendo algo complicado la comprensión de la materia de física al no tener medios para la comprobación o verificación de lo aprendido en el tema de cinemática que es el caso de estudio.

Por tal motivo que al no contar con un laboratorio de física para realizar las verificaciones. El presente trabajo pretende brindar una herramienta informática para poder subsanar de alguna manera la falta de un laboratorio de física. Con la herramienta informática se pretende lograr conseguir el interés de los alumnos en la materia de física del tema de cinemática con

la cual podrán realizar las comprobaciones o verificaciones de los problemas propuestos en clases.

### 1.2.2. TRABAJOS SIMILARES

En la carrera de Informática de la Universidad Mayor de San Andrés se pueden encontrar diversos proyectos y tesis de grado que se basan en resolver problemas educativos, en los diferentes campos relacionados al ámbito educativo. Algunos trabajos que se basan en resolver problemas en la educación. Las cuales se han centrado en la resolución de dificultades en diferentes campos.

"Diseño de espacios interactivos tridimensionales a través de la internet, aplicando realidad virtual inmersiva" (Arteaga, 2008): Tesis de grado que se enfoca en la creación de un entorno 3D usando una conexión a un servidor web, que concluye en las limitantes técnicas que se tienen en la creación de estos entornos, por la falta de herramientas y contenido 3D de ese año.

"Tutor inteligente para el aprendizaje de la matemática en tercero de primaria" (Chambi, 2008): Este sistema ayuda a que los niños de tercer siglo a prendan las matemáticas de forma virtual.

"Entrenamiento virtual para la olimpiada boliviana de física" (Coronel, 2010): El presente trabajo, que en si es un Software de Definición, Generación, Evaluación y Seguimiento de pruebas objetivas de selección múltiple, hasta ahí nada nuevo con la salvedad que dicho software está enmarcado y aprovecha recursos de un EVE/A (Entorno virtual de Enseñanza Aprendizaje)

## 1.3. PLANTEAMIENTO DEL PROBLEMA

En nuestro país todavía no se está dando un buen aprovechamiento del uso de las TIC en el sector educativo la cual lo podemos ver en los siguientes aspectos:

- La entrega de notebook a los profesores del sistema educativo en Bolivia por resolución ministerial 194/2011 el 21 de abril de 2011 por el ministerio de educación.
- Entrega de netbook por la empresa nacional QUIPUS a estudiantes del nivel secundario en todos los colegios fiscales de Bolivia q comenzó el 31 de julio del 2014.

- La materia de computación que no es considerada en la ley educativa Avelino-Siñani. Atribuyendo a que todos los profesores deberían complementar su materia con computación.

En estos últimos años en nuestro país se está tomando mucho más en cuenta el uso de las TIC en el área educativa, Aún falta bastante, porque aunque los docentes y estudiantes cuenten con material, aun no cuentan con el software necesario para su óptima educación, o no han recibido la capacitación para el uso del material brindado. El aprendizaje es diferente cuando se hace uso de las nuevas tecnologías ya que el estudiante se siente involucrado y en este sentido es que el ambiente esta mediado por tecnologías provocando un proceso de aprendizaje dinámico, no es la tecnología la que mejora la educación sino el uso de estas didácticamente combinados con la práctica, lo cual hace que las tecnologías sean un excelente aliado en la educación.

Todos los profesores cuenta con una NoteBook que el gobierno plurinacional de Bolivia brindo a cada uno, pero estas no son aprovechadas del todo. Como las Quas "NetBook las cuales fueron proporcionadas por la empresa Quipus" también no estas siendo usadas por la falta de herramientas informáticos o la falta de socialización de estas si existen. Y la falta de un laboratorio de física para la verificación de los ejercicios que realiza el docente. El desinterés de los estudiantes, son algunos problemas por la cuales atraviesa el Unidad Educativa Rep. De Italia.

### 1.3.1. PROBLEMA CENTRAL

¿De qué manera podemos captar el interés de los estudiantes de secundaria, de la Unidad Educativa Rep. De Italia, en el área de Cinemática?

### 1.3.2. PROBLEMAS SECUNDARIOS

- Dificultad en la comprensión del tema de cinemática, el cual causara el desinterés del tema.
- Problema en la resolución de ejercicios matemáticos lo cual provoca que no pueden resolver ejercicios de física.
- La falta de imaginación de los gráficos que realiza el docente de la materia de física del tema de cinemática, lo cual conlleva a la no comprensión de ejercicios de la materia.

- Poca interpretación de los datos brindados por el docente para la resolución de los ejercicios de Cinemática, lo cual contribuye a la mala resolución de los ejercicios.

## 1.4. DEFINICIÓN DE OBJETIVOS

### 1.4.1. OBJETIVO GENERAL

Desarrollar una aplicación que permita realizar simulaciones de cinemática basado en la programación grafica 3D para poder captar la atención de los estudiantes de nivel secundario de la Unidad Educativa Rep. De Italia.

### 1.4.2. OBJETIVOS ESPECÍFICOS

- Implementar criterios pedagógicos y tecnológicos en el desarrollo de la aplicación.
- Tomar en cuenta toda la bibliografía necesaria de cinemática.
- Capacitar a los usuarios del uso del simulador de cinemática.
- Desarrollar modelos en tres dimensiones para:
  - movimiento rectilíneo uniforme
  - movimiento rectilíneo uniformemente variado
  - caída libre
  - movimiento parabólico
  - movimiento circular uniforme.
- Realizar las funciones de:
  - movimiento rectilíneo uniforme.
  - movimiento rectilíneo uniformemente variado.
  - caída libre
  - movimiento parabólico.
  - movimiento circular uniforme.

## 1.5. JUSTIFICACIÓN

### 1.5.1. JUSTIFICACIÓN ECONÓMICA

Los beneficios que trae el simulador de cinemática es: agilizar el grado de desempeño del docente que puede impartir su clase haciendo uso de esta aplicación, lo cual reduce considerablemente el tiempo que emplea en realizar dibujos del escenario en el que se trabajara. Con la aplicación el docente podrá realizar su escenario de trabajo rápidamente

con el beneficio de ser en 3D. Por lo tanto se reducirá tiempo en la preparación del escenario de trabaja para la impartición del tema.

Para hacer uso del simulador se necesita un ordenador y si así amerita también un data show. Lo cual es un benefició, ya que la unidad educativa cuentan con estos equipos. Por lo tanto no se estaría haciendo gastos económicos al adquirir tales equipos. Por lo tanto la aplicación brinda beneficios al no realizar gastos económicos en la implementación de esta, también reduce considerablemente el tiempo que el profesor o alumnos realizan su escenario de trabajo.

### 1.5.2. JUSTIFICACIÓN SOCIAL

Cuando se haga uso de la aplicación en el aula, el grado de atención y captación del tema impartido por el profesor será mucho más óptimo y didáctico siendo más atractivo para el estudiante ya que se harán uso de objetos 3D en el escenario a trabaja.

Agilizara el desempeño del profesor, ya que no perderá mucho tiempo en realizar gráficos para impartir sus clases. Solo ara uso de los modelos tridimensionales que brinda el simulador de cinemática.

Por lo tanto si logramos captar la atención de los estudiantes con objetos 3D tendremos mejores resultados a la hora de la enseñanza y el aprendizaje, lo cual presenta un gran beneficio al momento de hacer uso del simulador de cinemática.

### 1.5.3. JUSTIFICACIÓN TECNOLÓGICO

Para que la aplicación funcione será necesario ordenadores con las siguientes especificaciones mínimas: CPU: compatible con el conjunto de instrucciones SSE2. Video de 1Gb, OS: Windows XP SP2+, Mac OS X 10.8+, Ubuntu 12.04+, SteamOS. Esto sería un obstáculo al momento de la implementación. Pero al contrario se hace beneficioso ya que en la unidad educativa se cuenta con estos equipos, ya sea el equipo del docente o los equipos de laboratorio de computación

Es un gran beneficio por qué se puede hacer uso de los ordenadores proporcionados por la empresa Quipus. Para la impartición de las clases.

## 1.6.ALCANCES Y LÍMITES

### 1.6.1. ALCANCES

El simulador que se creara permitirá realizar simulaciones de los ejercicios con la el profesor imparte su clase del tema de cinemática, también se podrá hacer uso del simulador para realizar talleres en el aula, de los siguientes temas:

- Movimiento rectilíneo uniforme.
- Movimiento rectilíneo uniformemente variado.
- Caída Libre
- Movimiento Parabólico.
- Movimiento Circular Uniforme.

### 1.6.2. LÍMITES

- El simulador solo se lo podrá usar en el tema de cinemática y no así en otros temas
- El simulador no realizar cálculos adicionales, de variables adicionales para la simulación.
- Se deberá introducir los datos de acuerdo al Sistema Internacional, no realizara conversiones.
- Indirectamente los beneficios del simulador serán limitador por la calidad de enseñanza del docente

## 1.7. APORTES

### 1.7.1. PRÁCTICO

La implementación del simulador será un aporte al ámbito educativo ya que les proporcionara una herramienta que les permitirá tener una mejor educación para los estudiantes del área de cinemática. La cual servirá para agilizar el proceso de enseñanza y aprendizaje.

### 1.7.2. TEÓRICO

El simulador en cuanto a lo pedagógico apoya al proceso de aprendizaje ya que facilita la comprensión de los ejercicios a realizarse, haciendo uso de objetos 3D lo cual al mismo tiempo lleva a la motivación de los estudiantes ya que al hacer uso de estos objetos 3D logramos captar la atención de los estudiantes. Por lo cual si tenemos la atención de los estudiantes nos llevaría a un mejor rendimiento de estos.

## 1.8. METODOLOGÍA

La metodología a seguir para el desarrollo del simulador será la Metodología de Ingeniería de Software Educativo MeISE. Ya que esta metodología aparte de ser una metodología para el desarrollo de software educativo se divide en dos etapas la etapa de definición y la etapa de desarrollo. Lo cual facilitara en gran manera y agilizar el desarrollo del simulador ya que en la primera etapa se centrara en la definición de los requisitos el diseño preliminar y la iteraciones, y en la segunda etapa se centrara a lleno al desarrollo, toman todas las iteraciones hechas en la primera etapa. Y así poder cumplir con nuestros objetivos propuestos.

Para su desarrollo usaremos Unity la cual es un motor de ideo Juegos que nos permite desarrollar aplicación 3D. La cual hace uso del lenguaje de programación CSharp C#, JavaScript y Boo. Pero en este caso aremos uso de C#. Ya que permite realizar la programación orientada a objetos P.O.O.

Pero esto no es todo lo que se usara para el desarrollo del simular también aremos uso de un software que permite realizar modelos en 3D. Que es Blender, la cual permitirá realizar todos los modelos 3D junto con su texturizado y la animación.

También se hará uso indirectamente de Open GL la cual permite desarrollar aplicación en 3 dimensiones. Se dice indirectamente ya que Unity como Blender hacen uso de este lenguaje de programación.

También se seguirá la metodología que el docente usa para la enseñanza del tema, tomando en cuenta los tipos de ejercicios, para que los estudiantes no tengan dudas en cuanto a la utilización del simulador, y así lograr una aplicación familiar a lo que ellos logran ven con el docente de la materia

# CAPÍTULO II
# MARCO TEÓRICO

## 2.1. INTRODUCCIÓN

En el presente capítulo se presentara una descripción breve sobre los principios y conceptos básicos para el desarrollo del proyecto, acerca de la metodología y técnicas y las herramientas que se usaran, para tener una mejor comprensión en el transcurso del desarrollo del proyecto.

En este capítulo se hablara sobre la metodología de ingeniería de software educativo, la cual ayudara en el desarrollo del simulador, los conceptos de los Movimientos que abarca el tema de Cinemática y las herramientas que se usaran serán: Blender, Unity y el lenguaje de programación CSharp.

## 2.2. METODOLOGÍA DE INGENIERÍA DE SOFTWARE EDUCATIVO

Como afirma Galvis (1992) cuando se ha determinado que es deseable contar con un apoyo informático para resolver un problema o conjunto de ellos, dependiendo de las necesidades que fundamentan esta decisión, cabe optar por un tipo de apoyo informático u otro.

Habrá necesidades que se pueden resolver usando herramientas informáticas de productividad, tales como un procesador de texto, una hoja de cálculo, un graficador, un manejador de bases de datos, o combinación de ellos. Por ejemplo, si interesa que los alumnos desarrollen sus habilidades de expresión verbal o de expresión gráfica y que se concentren en lo que generan antes que en la forma como lo hacen, siendo editable lo que hagan, el uso de un procesador de texto o de uno gráfico, pueden ser la solución más inmediata y adecuada. Si de lo que se trata es de facilitar el procesamiento de datos numéricos para que de ese modo puedan concentrarse en el análisis de los resultados procesados, una hoja de cálculo electrónico será un magnífico apoyo. Si interesa que los alumnos puedan alimentar, consultar, cruzar y analizar datos que cumplen con ciertos criterios, en un sistema manejador de bases de datos se tendrá un magnífico aliado.

Pero si las posibilidades que brindan las herramientas de propósito general no son adecuadas o son insuficientes, habrá que pensar en qué otro tipo de ambiente educativo informático es

conveniente. Tratándose de necesidades educativas relacionadas con el aprendizaje, según la naturaleza de éstas, se podrá establecer qué tipo de MEC conviene usar (Galvis, 1992).

- Un ***sistema tutorial*** se amerita cuando, siendo conveniente brindar el conocimiento al alumno, también interesa que lo incorpore y lo afiance, todo esto dentro de un mundo amigable y ojalá entretenido.
- Un ***simulador*** podrá usarse también para que el aprendiz llegue al conocimiento mediante trabajo exploratorio, conjetural, a través de aprendizaje por descubrimiento, dentro de un micromundo que se acerca razonablemente, en su comportamiento, a la realidad o a aquello que se intenta modelar.
- Un ***juego educativo*** será conveniente cuando, ligado al componente lúdico, interesa desarrollar algunas destrezas, habilidades o conceptos que van ligados al juego mismo.
- Los ***sistemas expertos*** se ameritan cuando lo que se desea aprender es lo que sabe un experto en la materia, conocimiento que no siempre está bien definido, ni siempre es completo, pero que es complejo y combina reglas de trabajo con reglas de raciocinio, con metaconocimiento. Por consiguiente, no se puede encapsular rígidamente, ni se puede transmitir el conocimiento en forma directa; se requiere interactuar con ambientes vivenciales que permitan desarrollar el criterio del aprendiz para la solución de situaciones en la forma com lo haría un experto.
- Un ***sistema tutor inteligente*** se ameritará cuando, además de desear alcanzar algún nivel de experticia en un área de contenido, interesa que el MEC asuma adaptativamente las funciones de orientación y apoyo al aprendiz, en forma semejante a como lo haría un experto en la enseñanza del tema.

Abud (2009) explica que La Metodología de Ingeniería de Software Educativo MeISE propone un ciclo de vida dividido en dos etapas. En la primera etapa se contempla la definición de requisitos y el análisis y diseño preliminar, durante los cuales se determinan en forma global las características que se pretende alcanzar con el producto, los requisitos pedagógicos, de comunicación y la arquitectura sobre la cual se construirá el software, y se termina con un plan de iteraciones las cuales se programan teniendo cuidado de que el producto que se libera al término de cada una está didácticamente completo, es decir que cubpre completamente algunos de los objetivos didácticos del software. Una vez establecidos estos lineamientos, inicia la segunda etapa, en la cual se procede a desarrollar el producto, de modo que el equipo toma cada iteración, la diseña, la construye, la prueba y

la implementa, evaluando al final la conveniencia de proseguir con subsecuentes iteraciones hasta obtener un producto completo.

**Las fases propuestas para la etapa de definición son:**

- la fase conceptual, durante la cual se identifican los requerimientos del sistema, se conforma el equipo de trabajo y se elabora el plan de desarrollo.
- la fase de análisis y diseño inicial, en la que se propone la arquitectura que servirá de base para la solución del problema y se establecen las características pedagógicas y de comunicación que regirán el desarrollo de software.
- finalmente la fase de plan de iteraciones, en la cual se divide el proyecto en partes funcionales que permitan mejor control en su desarrollo.

**En la etapa de desarrollo se tienen:**

- la fase de diseño computacional, en la que se realizará un diseño computacional detallado de un incremento específico del software.
- la fase de desarrollo, durante la cual se implementa la arquitectura en forma incremental (iteración por iteración).
- y la fase de despliegue, donde se realiza la transición del producto ejecutable al usuario final. Estas tres últimas etapas se repiten iterativamente para cada incremento del software. El modelo se ilustra en la Figura 2.1. etapa de definición

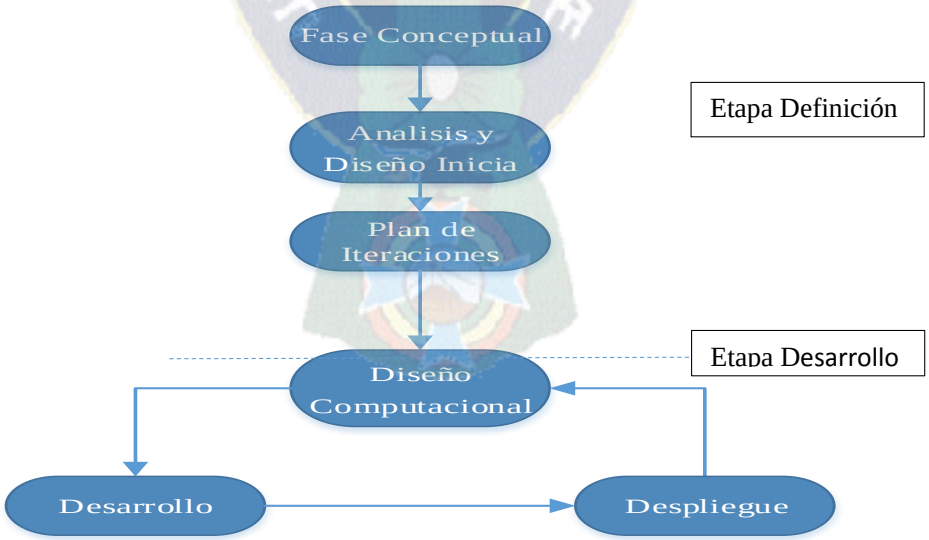



**Figura 2.1:** Ciclo de vida de la metodología.
Tomado de Abud (2009)

## 2.3. DESCRIPCIÓN DE FASES

### 2.3.1. ETAPA CONCEPTUAL

Esta etapa inicia con una investigación sobre los requerimientos que se cubrirán con el producto a desarrollar, delimitando su alcance. Se desarrolla el plan del proyecto, se evalúan riesgos y se establecen los criterios de éxito. En la tabla 2.1 se muestran las actividades a realizar y los artefactos que se generan en esta fase (Abud, 2009).

**Tabla 2.1:** Actividades y Artefactos de la Fase Conceptual.
Tomado de Abud (2009)

| ACTIVIDAD | ARTEFACTO |
|---|---|
| Analizar las necesidades educativas | Modelo instruccional (incluye temática a atender, objetivos, conocimientos previos, fuentes de información, modelo educativo a utilizar, elementos de motivación y formas de evaluación)<br>Glosario (descripción de los términos que pueden causar confusión o duda) |
| Revisar alternativas de solución | Estudio de alternativas (establece las diferentes alternativas que se tienen para el desarrollo del software, se determina el tipo de modelo educativo y se justifica la elección) |
| Elaborar un estudio de riesgos | Lista de riesgos (establece los riesgos relativos al desarrollo y a los aspectos pedagógicos y la forma de atenderlos) |
| Conformar del equipo de trabajo y el plan inicial de desarrollo | Plan Inicial (se conforma el equipo de trabajo, se elabora la programación de actividades, se asignan responsables a cada una y se determinan los tiempos estimados para llevarlas a cabo) |
| Identificar la funcionalidad que se pretende alcanzar con el software | Modelo de actores (identifica los tipos de usuario que utilizarán el software y describe<br>sus características) Modelo de casos de uso (establece un modelo general de las funciones que cubrirá el sistema a través de diagramas de casos de uso y su especificación) |
| Establecer los criterios de medición de calidad del proceso, considerando aspectos tanto técnicos como pedagógicos | Modelo de aceptación (incluye las características mínimas que deben cumplirse para que el producto se acepte) |

### 2.3.2. ANÁLISIS Y DISEÑO INICIAL

En la fase de análisis y diseño inicial se analiza el dominio del problema y se establece la arquitectura del sistema. En este punto se describen a detalle los requisitos del software y las características educativas y de comunicación que el producto debe contemplar. En la tabla 2.2 se detallan estas actividades (Abud, 2009).

**Tabla 2.2:** Actividades y Artefactos de la Fase de Análisis y Diseño Inicial.
Tomado de Abud (2009)

| ACTIVIDAD | ARTEFACTO |
|---|---|
| Identificar los requisitos funcionales y no funcionales que se cubrirán con el software | Modelo de requisitos (Se determinan los requisitos que debe cumplir el software en cuanto a funcionalidad, comunicación, interfaz y docencia.) |
| Establecer la arquitectura del software | Descripción de la arquitectura (establecer la arquitectura base sobre la cual se desarrollará el software; se debe considerar que dicha arquitectura sea capaz de atender adecuadamente las tareas de aprendizaje que se van a manejar) |
| Elaborar el diseño educativo | Modelo educativo (Se definen el objetivo terminal y los sub-objetivos, y en base a éstos se establecen las tareas de aprendizaje apegadas al tipo de modelo educativo) |
| Elaborar el diseño de comunicación general del producto | Modelo de interfaz (diseño de las zonas de comunicación y pantallas que se seguirán a lo largo del desarrollo) Modelo de navegación (diseño de los caminos de navegación generales que se presentarán al usuario) Prototipo de la interfaz de usuario (establecer las plantillas de diseño que se seguirán a lo largo del desarrollo) |

### 2.3.3. PLAN DE ITERACIONES

Una vez identificados los requisitos a cubrir con el software se procede a analizar cuántos subproductos funcionales pueden producirse de modo que se puedan liberar partes operativas del sistema final, con el objetivo de llevar un mejor control en el desarrollo. Una vez identificados los incrementos se priorizan y se colocan con mayor prioridad aquellos que cubren los conocimientos base. En la tabla 2.3 se muestran los resultados de esta fase.

Una vez identificados los requisitos a cubrir con el software se procede a analizar cuántos subproductos funcionales pueden producirse de modo que se puedan liberar partes operativas del sistema final, con el objetivo de llevar un mejor control en el desarrollo. Una vez identificados los incrementos se priorizan y se colocan con mayor prioridad aquellos que cubren los conocimientos base. En la tabla 2.3 se muestran los resultados de esta fase.

**Tabla 2.3:** Actividades y Artefactos de la Fase del Plan de Iteraciones.
Tomado de Abud (2009)

| ACTIVIDAD | ARTEFACTO |
|---|---|
| Diseñar las iteraciones de forma que las versiones ejecutables cubran objetivos didácticos bien planeados, de acuerdo a la secuencia de temas. | Plan de iteraciones (dividir el desarrollo en iteraciones, cuidando de que cada iteración cubre requisitos y objetivos educativos completos) |
| Priorizar las iteraciones, de modo que las que contienen conocimientos básicos que se requieren como base para aprendizajes posteriores se ejecuten primero. | Lista de Iteraciones Priorizadas (ordenar las iteraciones programadas de forma lógica de acuerdo a los contenidos) |

### 2.3.4. DISEÑO COMPUTACIONAL

Para cada iteración se debe elaborar el diseño computacional detallado, de modo que sirva de base para el desarrollo. Los artefactos y actividades propios, los cuales se muestran en la tabla 2.4.

**Tabla 2.4:** Actividades y Artefactos de la Fase de Diseño Computacional.
Tomado de Abud (2009)

| ACTIVIDAD | ARTEFACTO |
|---|---|
| Realizar el plan de trabajo de la iteración | Plan de trabajo (se determinan las tareas que se realizarán en el diseño del software, se asignan a los miembros del equipo y se calendarizan) |
| Elaborar el diseño computacional | Modelo de diseño (detallar el diseño a través de diagramas de clases y secuencia, incluir la descripción de clases y métodos; para los desarrollos que requieren bases de datos, incluir la especificación de diccionario de datos y diagramas entidad relación.) |
| Refinar el diseño de navegación | Modelo de navegación refinado (diseñar los caminos de navegación específicos para la iteración en desarrollo) |
| Refinar prototipo de interfaz | Modelo de navegación refinado (diseñar los caminos de navegación específicos para la iteración en desarrollo) |
| Refinar prototipo de interfaz | Modelo de interfaz usuario (desarrollar las pantallas específicas para los elementos de la iteración en desarrollo) |

### 2.3.5. DESARROLLO

Se desarrolla en esta fase el producto, implementando la arquitectura de manera que se obtiene una versión del software lista para que sea utilizada por los usuarios finales. En la tabla 2.5 se incluyen sus elementos a detalle.

**Tabla 2.5:** Actividades y Artefactos de la Fase de Desarrollo.
Tomado de Abud (2009)

| ACTIVIDAD | ARTEFACTO |
|---|---|
| Desarrollar los componentes | Modelo de desarrollo (Determinar los componentes a desarrollar y documentarlos.) |
| Probar los componentes | Modelo de pruebas unitarias (Realizar pruebas de los componentes contra los criterios previamente establecidos. Estas pruebas deben incluir las pruebas del diseño instruccional) |
| Integrar al desarrollo previo | Modelo de Integración (establecer un plan para incorporar el nuevo desarrollo a la liberación previa si es el caso) |
| Realizar pruebas de integración | Pruebas de integración (realizar pruebas para verificar que la incorporación del nuevo incremento no ha inducido fallas al sistema) |

### 2.3.6. FASE DE DESPLIEGUE

En la fase de despliegue se realiza la transición del producto a los usuarios. Aquí se culmina con una versión ejecutable del producto. Las actividades y artefactos de esta fase se describen en la tabla 2.6. Al finalizar esta etapa se evalúa la conveniencia de continuar los desarrollos, y en su caso regresar a la etapa de diseño computacional para continuar con el siguiente incremento.

**Tabla 2.6:** Actividades y Artefactos de la Fase de Despliegue.
Tomado de Abud (2009)

| ACTIVIDAD | ARTEFACTO |
|---|---|
| Entregar producto al usuario | Producto (Se debe entregar el producto debidamente empacado, etiquetado y con información sobre su contenido, aplicación, población objetivo y requerimientos de instalación) Manual de Usuario (Debe contener información<br>detallada de cómo utilizar el software)<br>Manual de Instalación (información de los requerimientos para su funcionamiento y procedimiento de instalación) |
| Evaluar las características de calidad y satisfacción de los usuarios | Aceptación del Usuario (realizar pruebas con los usuarios finales y comprobar su grado de satisfacción y efectividad del software) |
| Evaluar la conveniencia de continuar con otro incremento al producto | Evaluación de despliegue (analizar los resultados de la prueba de aceptación del usuario y determinar si es conveniente seguir con otra iteración.) |

## 2.4. SIMULADOR

Real Academia Española. (2006). Diccionario de la lengua española define que: un simulador es un aparato que produce el comportamiento de un sistema en determinadas condiciones, aplicado generalmente para el entrenamiento de quienes deben manejar dicho sistema.

Goldsmith y Ray (2016) consideran que: "Simulación es una técnica numérica para conducir experimentos en una computadora digital. Estos experimentos comprenden ciertos tipos de relaciones matemáticas y lógicas, las cuales son necesarias para describir el comportamiento y la estructura de sistemas complejos del mundo real a través de largos períodos."

### 2.4.1. SIMULACIÓN EN INFORMÁTICA

Goldsmith y Ray (2016) coinciden que: En informática la simulación tiene todavía mayor significado especializado: Alan Turing usó el término "*simulación*" para referirse a lo que pasa cuando una computadora digital corre una tabla de estado "corre un programa" que describe las transiciones de estado, las entradas y salidas de una máquina.

## 2.5. CINEMÁTICA

Galileo Galilei (1642) Expresa que: La observación y el estudio de los movimientos ha atraído la atención del hombre desde tiempos remotos. Así, es precisamente en la antigua Grecia en donde tiene su origen la sentencia «Ignorar el movimiento es ignorar la naturaleza», que refleja la importancia capital que se le otorgaba al tema. Siguiendo esta tradición, científicos y filósofos medievales observaron los movimientos de los cuerpos y especularon sobre sus características. Los propios artilleros manejaron de una forma práctica el tiro de proyectiles de modo que supieron inclinar convenientemente el cañón para conseguir el máximo alcance de la bala. Sin embargo, el estudio propiamente científico del movimiento se inicia con Galileo.

### 2.5.1. MOVIMIENTO

A partir de la experiencia cotidiana nos damos cuenta que el movimiento representa el cambio continuo en la posición de un objeto. La física estudia tres tipos de movimiento: traslacional, rotacional y vibratorio (Tipler, 2006).

### 2.5.2. PARTÍCULA

Expresa que: Una partícula es un punto individual de masa, como un electrón; pero también designamos un objeto cuyas partes se mueven exactamente de la misma manera. Incluso los objetos complejos pueden ser considerados como partículas, si no existen movimientos internos como rotación o la vibración de sus partes (Tipler, 2006).

### 2.5.3. POSICIÓN ($\vec{r}$ )

Dice que: Para describir el movimiento de una partícula en el espacio, primero hay que poder describir su posición. Consideremos una partícula que está en el punto P en un cierto instante. El vector de posición r de la partícula en ese instante es un vector que va desde el origen del sistema de coordenadas al punto P (Tipler, 2006).

### 2.5.4. TRAYECTORIA

Es la sucesión de posiciones que ocupa la partícula durante su movimiento. Podemos decir que la curva que describe la partícula se llamará trayectoria. A la longitud de la trayectoria, se le denomina "distancia recorrida".

### 2.5.5. DESPLAZAMIENTO $\Delta\vec{r}$

El vector desplazamiento $\Delta\vec{r}$, se define como el cambio de posición que se realiza en este intervalo.

$$\Delta\vec{r} = \overrightarrow{r_f} - \overrightarrow{r_i}$$

Para describir el movimiento de una partícula, introduciremos la cantidad física velocidad y aceleración, que tienen definiciones sencillas en la física, aunque son más precisas y un poco distintas de las empleadas en el lenguaje cotidiano. Si pone atención a estas definiciones trabajará mejor con éstas y otras cantidades físicas importantes. Un aspecto importante de las definiciones de velocidad y aceleración es que son vectores, esto implica que tienen magnitud y dirección.

### 2.5.6. VELOCIDAD

- Velocidad Media $\overrightarrow{V_m}$

Se define a la velocidad media de la partícula durante el intervalo de tiempo $\Delta t$ como la razón entre el desplazamiento y el intervalo de tiempo.

$$\overrightarrow{V_m} \equiv \frac{\Delta \vec{r}}{\Delta \vec{t}}$$

Según esta definición, la velocidad media tiene dimensiones de longitud dividida por el tiempo (L/T)-m/s, en unidades del SI.

- La velocidad media es una cantidad vectorial dirigida a lo largo de r .
- Advierta que la velocidad media entre dos puntos es independiente de la trayectoria entre los dos puntos.
- En general la velocidad media depende del intervalo de tiempo escogido.

❖ Velocidad Instantánea $\vec{V}$

Se define que el límite de la velocidad media, $\Delta\vec{r}/\Delta\vec{t}$, conforme $\Delta\vec{t}$ tiende a cero; es igual a la razón instantánea de cambio de posición con el tiempo.

$$\overrightarrow{V_m} = \lim_{\Delta t \to 0} \frac{\Delta \vec{r}}{\Delta \vec{t}} = \frac{\partial \vec{r}}{\partial \vec{t}}$$

Según esta definición, la velocidad instantánea tiene dimensiones de longitud dividida por tiempo [L/T]- m/s, en unidades del SI.

- La dirección del vector velocidad instantánea, en cualquier punto en una trayectoria de la partícula, está a lo largo de la línea que es tangente a la trayectoria en ese punto y en dirección del movimiento.
- A la magnitud del vector velocidad instantánea, se le conoce como rapidez.
- El velocímetro de un automóvil indica la rapidez, no la velocidad.
- La palabra instante tiene un significado un poco distinto en física que en el lenguaje cotidiano. Podemos decir "duró un instante" para referirnos a algo que "duró un intervalo de tiempo muy corto"; pero, en física, un instante no tiene duración; es un solo valor del tiempo.

### 2.5.7. MOVIMIENTO RECTILÍNEO UNIFORME (M.R.U)

Características:

1. La velocidad permanece constante en magnitud y dirección.
2. En tiempos iguales se recorren distancias iguales.
3. La distancia recorrida es directamente proporcional al tiempo transcurrido.

Se puede ver una representación de tal movimiento en la figura 2.2.


**Figura 2.2:** Representación del MRU.
Tomado de Tipler (2006)

Formula: $$V = \frac{d}{t}$$

Donde:

V: rapidez constante del movimiento.

d: distancia recorrida.

t: Tiempo transcurrido.

### 2.5.8. MOVIMIENTO RECTILÍNEO UNIFORMEMENTE (M.R.U.V)

El movimiento de una partícula es rectilíneo, cuando su trayectoria es una recta. Si el movimiento rectilíneo, tiene una aceleración constante o uniforme se dice que el movimiento es rectilíneo uniformemente acelerado. Es importante tener en cuenta que si la aceleración de una partícula permanece constante, su magnitud y dirección permanecen invariables durante el movimiento.

En el movimiento unidimensional, con aceleración constante, la aceleración media es igual a la aceleración instantánea, en consecuencia la velocidad aumenta o disminuye a la misma tasa durante todo el movimiento lo cual se puede observar en la Figura 2.3.

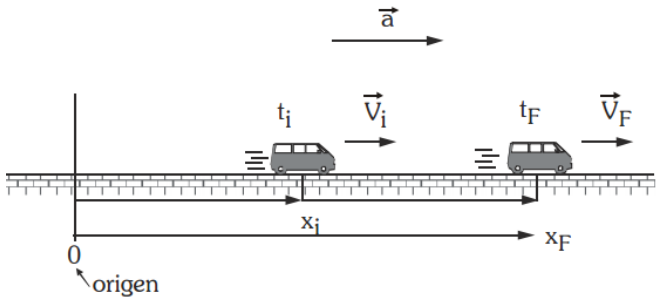



**Figura 2.3:** Representación del MRUV.
Tomado de Tipler (2006)

Características del MRUV.

- La trayectoria es una línea recta.
- La velocidad varía uniformemente.
- La aceleración es constante.

Ecuaciones que nos permiten el cálculo de tal movimiento:

Velocidad en un determinado instante.

$$V_f = f(t)$$

$$V_f = V_i \pm at$$

Velocidad en un determinado instante.

$$V_f = f(d)$$

$$V_f^2 = V_i^2 \pm 2ad$$

Distancia en un determinado instante. $d = f(t)$

$$d = V_i t \pm \frac{1}{2} a t^2$$

Donde:

$V_f$: Velocidad Final.

$V_i$: Velocidad Inicial.

a: Aceleración.

d: Distancia.

### 2.5.9. CAÍDA LIBRE (C.L)

Caída libre es el movimiento rectilíneo en dirección vertical con aceleración constante realizado por un cuerpo cuando se deja caer en el vacío. Este tipo de movimiento sigue las mismas ecuaciones del MRUV. Con la diferencia de que envés de la aceleración "a" cambiamos por otra aceleración la cual sería la gravedad "g".

La gravedad llega a ser una aceleración constate con el valor de 9.81 $m/s^2$.

### 2.5.10. MOVIMIENTO PARABÓLICO (M.P)

El movimiento parabólico, también conocido como tiro oblicuo, consiste en lanzar un cuerpo con una velocidad que forma un ángulo α con la horizontal. En la siguiente figura puedes ver una representación de la situación.

El movimiento parabólico está compuesto por dos tipos de movimientos; uno es el M.R.U. y el otro es C.L, que podemos obsérvalos en la Figura 2.4.

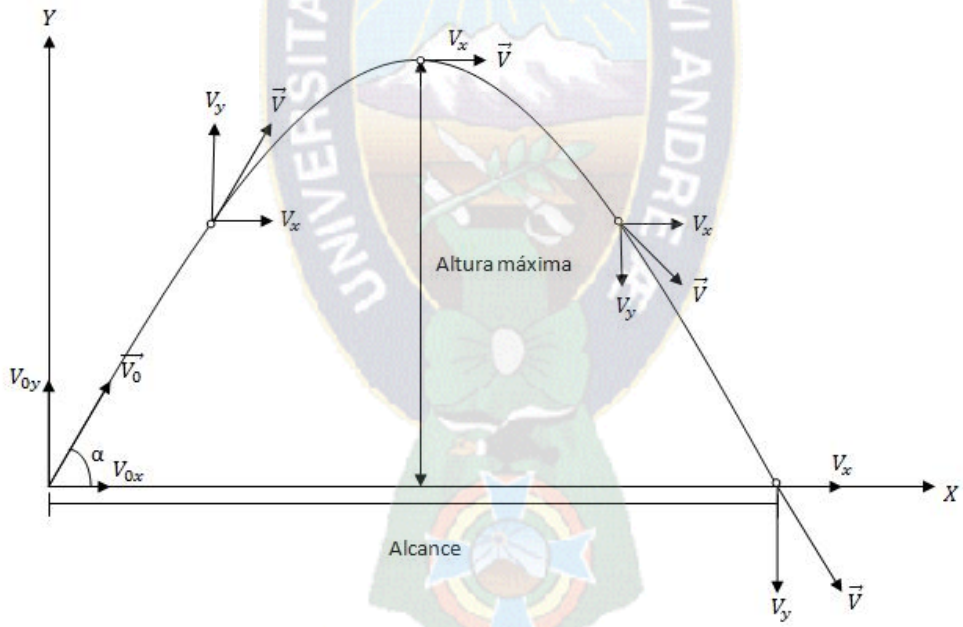
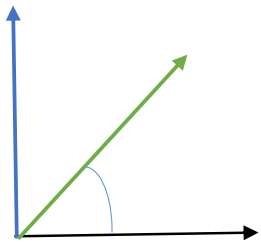



**Figura 2.4:** Composición del movimiento parabólico.
Tomado de Tipler (2006)

En cuanto a las ecuaciones a usar en este tipo de movimiento se hace uso tanto las ecuaciones de M.R.U y C.L. Con el plus de que en esta ocasión la velocidad tiene una diferencia, la cual varía tanto en el eje X y en el eje Y la cual se puede ver en la Figura 2.5.



$$V_x = V * \cos\alpha$$

$$V_y = V * \sin\alpha$$

$$\tan\alpha = \frac{V_y}{V_x}$$

**Figura 2.5:** Descomposición de la velocidad.
Tomado de Tipler (2006)

Según el teorema de Pitágoras

$$V = \sqrt{V_x^2 + V_y^2}$$

### 2.5.11. MOVIMIENTO CIRCULAR UNIFORME (M.C.U)

El movimiento circular uniforme es un movimiento de trayectoria circular en el que la velocidad angular es constante. Esto implica que describe ángulos iguales en tiempos iguales. En él, el vector velocidad no cambia de módulo pero sí su dirección, es tangente en cada punto a la trayectoria. Esto quiere decir que no tiene aceleración tangencial ni aceleración angular, aunque sí aceleración normal.

Donde:

| | |
|---|---|
| $F = \frac{Numero\ de\ oscilaciones}{tiempo}$ | Frecuencia |
| $T = \frac{1}{F}$ | Periodo |
| $V = \omega * R$ | Velocidad lineal o Tangencial |
| $\omega = \frac{2\pi}{T}$ | Velocidad angular |

## 2.6. HERRAMIENTAS

A continuación se dará una descripción breve de las herramientas que usaremos para el desarrollo de la aplicación.

### 2.6.1. UNITY 5

La plataforma de desarrollo es flexible y poderosa para la creación de video juegos y experiencias interactivos 3D y 2D multiplataforma. Es un ecosistema completo para todo aquel que busque desarrollar un negocio a partir de la creación de contenido de alta gama y conectarse con sus jugadores y clientes más fieles y entusiastas. (Unity, 2016)

Unity 3D es un motor gráfico desarrollado por Unity Technologies desde 2001 con el objetivo de permitir a todo el mundo crear atractivos entornos 3D. En los últimos años ha ganado varios premios, entre ellos el "Wall Street Journal Technology Innovation Award", y se encuentra entre las 5 mejores compañías de juegos de 2009 según Gamasutra junto a Apple, Epic Games, Valve y Zynga. Entre sus clientes podemos encontrar a Bigpoint, Cartoon Network, Coca-Cola, Disney y Electronic Arts (Unity, 2016).

Unity 3D permite desarrollar software para un amplio número de plataformas, de modo que es sumamente atractivo para un amplio rango de desarrolladores de videojuegos, desde grandes compañías que pretendan desarrollar un AAA, hasta pequeños equipos de estudiantes.

Estos datos de por sí ya nos pueden dar una ligera idea del potencial del Unity.

### 2.6.2. BLENDER

Es un Programa de modelado en 3D, apoyado por varias herramientas, es multiplataforma "corre en windows XP, Vista 32 y 64 bits, Linux 32 y 64 bits, MacOS, solaris, etc.". Fue creado por la empresa Not a Number "NaN" (Blender, 2016).

Está orientado a artistas y profesionales del diseño y multimedia, puede ser usado para crear, visualizaciones 3D estáticas o vídeos de alta calidad. También incorpora un motor de 3D en tiempo real el cual permite la creación de contenido tridimensional interactivo que puede ser reproducido de forma independiente.

Blender se desarrolla como Software Libre, con el código fuente disponible bajo la licencia GNU GPL, su descarga y su uso es completamente gratuito. Aun así recomendaría que si haces dinero con el programa dones una cantidad a la fundación o compres algunos de sus productos (como el manual oficial) para que siga el desarrollo.

**Características principales:**

- Software libre, gratuito y multiplataforma
- Potente y versátil
- Importa y exporta de múltiples formatos 3D
- Soporte gratuito vía blender3d.org
- Manual multilenguaje en línea
- Una comunidad mundial creciente.
- Un archivo ejecutable pequeño que permite una fácil distribución
- Te puedes olvidar de números de serie y activaciones
- Múltiples plugins también gratuitos que expanden las posibilidades del programa
- Si sabes programar puedes usar el código fuente para hacer modificaciones

La interfaz de Blender puede intimidar de buenas a primeras ya que no es como el del común de programas en 3D, tiene una gran cantidad de elementos que lo hacen ver complejo pero siguiendo los ejemplos y tutoriales de su página un usuario común de programas en 3D tendrá un estupendo manejo del programa en unos cuantos días.

### 2.6.3. LENGUAJE DE PROGRAMACIÓN C#

El lenguaje de programación C# fue creado por el Danés Anders Hejlsberg que diseño también los lenguajes Turbo Pascal y Delphi. El C# (pronunciado en inglés "C sharp" o en español "C sostenido") es un lenguaje de programación orientado a objetos. Con este nuevo lenguaje se quiso mejorar con respecto de los dos lenguajes anteriores de los que deriva el C, y el C++ (Archer, 2001).

Con el C# se pretendió que incorporase las ventajas o mejoras que tiene el lenguaje JAVA. Así se consiguió que tuviese las ventajas del C, del C++, pero además la productividad que posee el lenguaje JAVA y se le denomino C#.

Algunas de las características del lenguaje de programación C# son: Su código se puede tratar íntegramente como un objeto. Su sintaxis es muy similar a la del JAVA. Es un lenguaje orientado a objetos y a componentes. Armoniza la productividad del Visual Basic con el poder y la flexibilidad del C++. Ahorramos tiempo en la programación ya que tiene una librería de clases muy completa y bien diseñada (Archer, 2001).

A pesar que el lenguaje C# forma parte de la plataforma .NET, que es una interfaz de programación de aplicaciones. C# es un lenguaje independiente que originariamente se creó para producir programas sobre esta plataforma .NET.

Esta plataforma se creó, entre otras razones, porque el Visual Basic era uno de los lenguajes de programación que se encargaban de desarrollar estas aplicaciones. Pero el Visual Basic es un lenguaje orientado a objetos algo pobre, porque se quiso que fuese desde su creación un lenguaje fácil de aprender para los programadores novatos. Por esto, surgió el C#, para suplir esta deficiencia del Visual Basic (Archer, 2001).

El Visual Basic no tiene algunas de las características necesarias como la herencia, los métodos virtuales, la sobrecarga de operadores, etc. Que se han conseguido con el C# y la plataforma .NET.

# CAPÍTULO III
# MARCO APLICATIVO

## 3.1. INTRODUCCIÓN

En este capítulo se realizara la descripción lógica y el modelamiento de la aplicación para alcanzar los objetivos definidos para la culminación del proyecto, y así conseguir que los requerimientos se estructuren.

Se propone la creación de una aplicación 3D con fines educativos, es decir una herramienta educativa computarizado, para poder motivar y fortalecer el proceso de aprendizaje del tema de cinemática, de la materia de física, apoyado en la teoría de la materia. Para esto se hace uso de un ordenador aplicando la tecnología tridimensional para motivar el aprendizaje y enriquecer la experiencia del alumno.

## 3.2. FASE CONCEPTUAL

### 3.2.1. ANÁLISIS DE LAS NECESIDADES EDUCATIVAS

En la enseñanza del nivel secundario surgen muchas dificultades relacionadas a los ámbitos de aprendizaje, desinterés de los jóvenes y la falta de técnicas de enseñanza entre otros. Una de las más grandes es la complejidad y practicas necesarias, que se requiere en cuanto a la comprensión de la materia de física en el tema de cinemática ya que muchas veces el alumno no llega a entender claramente lo que el profesor pretende explicar. Es por tal motivo que a menudo se hace uso de materiales educativos didácticos como ser: el huso de laboratorios, exposiciones y otros.

Otra necesidad educativa radica en el hecho de que uno de los impulsores en cuanto a la enseñanza es la motivación. Lograr motivar al estudiante para que capte claramente lo que se le explica sobre la materia, lo cual puede ser complicado para los profesores.

Por lo tanto los problemas encontrados en el proceso de aprendizaje de los estudiantes de nivel secundario son los siguientes:

- **Motivación**: Este es uno de los elementos más importantes para la adquisición del conocimiento. Es una tarea que los profesores (as) deben realizar para que los estudiantes puedan interesarse más en lo que están aprendiendo.

- **Mal aprovechamiento del material tecnológico**: Actualmente la mayoría de las unidades educativas de Bolivia cuenta con laboratorios de computación, física, biología y química. Donde cuentan con equipos para la enseñanza de la materia. Además el gobierno del Presidente Evo Morales Ayma hiso entrega a todos los colegios del sector público de computadoras personales Kuaas a los estudiantes del sexto de secundaria, las cuales no son aprovechadas ya que muchas unidades educativas siguen con las temáticas tradicionales o simplemente no existe el software adecuado para que la enseñanza sea más dinámica e interesante para el estudiante.
- **Complejidad**: Debido a la dedicación que debe darse al proceso de enseñanza-aprendizaje no solo en física si no en las demás materias de la curricula educativa, suele ser un proceso complejo y arduo hasta llegar a una comprensión eficiente por parte del estudiante ya que es la base de su formación profesional en un futuro.

### 3.2.2. ALTERNATIVAS DE SOLUCIÓN

Con los problemas anteriormente listados sobre el proceso de enseñanza y aprendizaje se propone las siguientes soluciones:

- **Motivación:** La aplicación a realizar estimulara la motivación en el estudiante, mediante gráficos 3D didácticos y desplegar la simulación con los datos proporcionados mediante el problema propuesto. Esperando que el uso del simulador 3D motive aún más a los estudiantes de nivel secundario.
- **Mal aprovechamiento del material tecnológico:** La aplicación aprovechara el uso de las Kuaas u ordenadores con las cuales cuenta la Unidad Educativa Rep. De Italia. Convirtiendo un aula tradicional en un ambiente de aprendizaje didáctico apoyándose en las nuevas tecnologías.
- **Complejidad:** La aplicación reducirá la complejidad de enseñanza en el área de física tema cinemática, al ser un material didáctico y fácil de entender.

### 3.2.3. ESTUDIO DE RIESGOS

De manera general el contenido de la aplicación está basado solo en el tema de Cinemática a nivel colegiatura. Por lo tanto solo se podrá hacer uso en este ámbito.

El contenido básico de la aplicación toca los siguientes puntos;

- Movimiento rectilíneo uniforme.

- Movimiento rectilíneo uniformemente variado.
- Caída libre.
- Movimiento parabólico.
- Movimiento circular uniforme.

Los estudiantes podrán hacer uso de la aplicación, de manera individual o grupal de manera que todos tengan la posibilidad de contar con el material brindado.

### 3.2.4. FUNCIONALIDAD QUE SE PRETENDE ALCANZAR CON EL SOFTWARE

Se tendrá como usuario general a los estudiantes y profesores, como se describe en la Tabla 3.1 las cuáles serán específicamente del área de física y solo en el tema de cinemática.

**Tabla 3.1:** Roles y tareas de usuario.

| ROL | DESCRIPCIÓN |
|---|---|
| Usuario general (Estudiante- profesor) | Tendrá acceso completo a la aplicación de forma directa, y será el encargado de introducir los datos necesarios para realizar la simulación. |

Ya establecidos los roles de uso podemos proceder a la realización de las funcionalidades del simulador, la cual se representara mediante diagramas de casos de usos.

### 3.2.5. DIAGRAMA DE CASOS DE USOS

**Módulo principal**. En este módulo se imprimirá desde la pantalla de inicio, conjuntamente con las opciones y detalle de la aplicación, como se muestra en la figura 3.1.

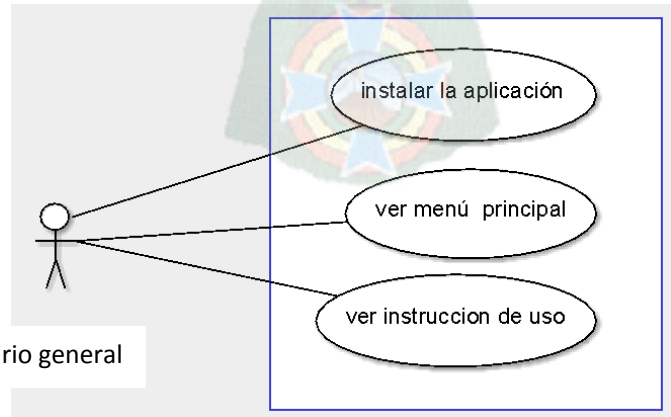



**Figura 3.1:** Diagrama de casos de uso principal.

**Módulo de simulación.** Este módulo contempla todos lo relacionado a la generación de la simulación de los distintos tipos de movimientos que se tocara en el proyecto. Como ser movimiento rectilíneo uniforme, movimiento rectilíneo uniformemente variado, etc. Las cuales se denota en la figura 3.2.

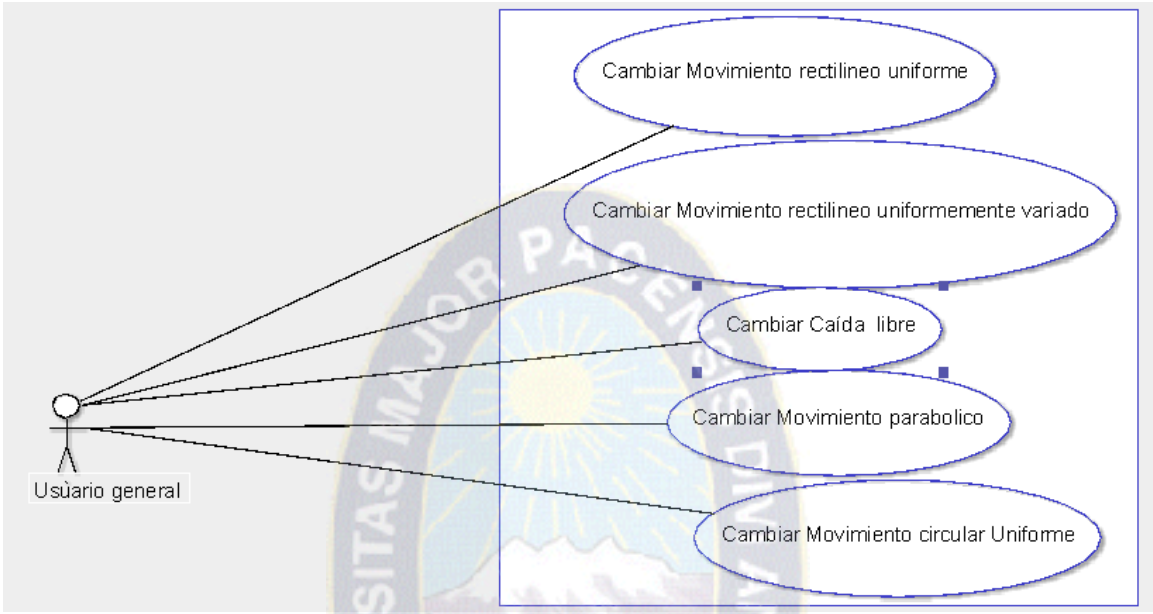



**Figura 3.2:** Diagrama de casos de uso. Módulo de simulación. Cambiar.

## 3.3. CASOS DE USO EXPANDIDO

La descripción detallada de cada casos de uso tanto del menú principal y el Modulo de simulación se las explica en las tablas Tabla 3.2 hasta la Tabla 3.9.

**Tabla 3.2:** Casos de uso Instalar la aplicación.

| Casos de uso: **Instalar la aplicación** | | | Nro.: **1** |
|---|---|---|---|
| Descripción: **casos de uso para instalar el archivo simulador.exe** | | | |
| Actores: | Usuario general (Estudiante-profesor) | | |
| Condiciones de entrada | Una vez identificada las características de hardware y software del ordenador, podrá hacer la instalación | | |
| Flujo de evento normal | **Paso** | **Estrada del actor** | **Respuesta del sistema** |
| | 1 | Escoger el icono y nombre respecto al simulador de cinemática | Se inicia la instalación, si no existe dificultades se instala correctamente |
| Post condición | La aplicación se inicia exitosamente | | |

**Tabla 3.3:** Casos de uso Ver Menú principal.

| Casos de uso: **Ver menú principal** | | | Nro.: **2** |
|---|---|---|---|
| Descripción: **casos de uso para visualizar l menú principal de la aplicación** | | | |
| Actores: | Usuario general (Estudiante-profesor) | | |
| Condiciones de entrada | Instalación correcta y terminada la presentación de la aplicación. | | |
| Flujo de evento normal | **Paso** | **Estrada del actor** | **Respuesta del sistema** |
| | 1 | Escoger el icono y nombre respecto al simulador de cinemática | Se inicia la instalación, si no existe dificultades se instala correctamente |
| | 2 | | Se inicia la aplicación con la interfaz principal |
| | 3 | | Luego de la interfaz principal se mostrara el menú con las opciones a realizar. |
| Post condición | La aplicación muestra un menú con opciones que nos permitirán interactuar con la aplicación. | | |

**Tabla 3.4:** Casos de uso ver instrucciones de uso.

| Casos de uso: **Ver instrucciones de uso** | | | Nro.: **3** |
|---|---|---|---|
| Descripción: **casos de uso para ver el uso correcto de la aplicación** | | | |
| Actores: | Usuario general (Estudiante-profesor) | | |
| Condiciones de entrada | En el menú principal debe haberse elegido la opción “ayuda” | | |
| Flujo de evento normal | **Paso** | **Estrada del actor** | **Respuesta del sistema** |
| | 1 | Escoger el icono y nombre respecto al simulador de cinemática | Se inicia la instalación, si no existe dificultades se instala correctamente |
| | 2 | | Se inicia la aplicación con la interfaz principal |
| | 3 | | Luego de la interfaz principal, se mostrara el menú de opciones. |
| | 4 | El usuario debe elegir la opción “Ayuda” del menú principal | Al elegir la opción Ayuda nos mostrará una interfaz con los pasos correctos para el uso de la aplicación. |
| Post condición | Por consiguiente se hará uso correcto de la aplicación | | |

**Tabla 3.5:** Casos de uso cambiar a movimiento rectilíneo uniforme.

| **Casos de uso: Cambiar a movimiento rectilíneo uniforme** | Nro.: **4** | | |
|---|---|---|---|
| Descripción: **casos de uso para pasar al modo de simulación MRU** | | | |
| Actores: | Usuario general (Estudiante-profesor) | | |
| Condiciones de entrada | En el menú principal debe haberse elegido la opción “MRU” | | |
| Flujo de evento normal | **Paso** | **Estrada del actor** | **Respuesta del sistema** |
| | 1 | Escoger el icono y nombre respecto al simulador de cinemática | Se inicia la instalación, si no existe dificultades se instala correctamente |
| | 2 | | Se inicia la aplicación con la interfaz principal |
| | 3 | El usuario debe escoger la opción MRU | Al elegir la opción se generara una interfaz que nos permitirá realizar la respectiva simulación. |
| | 4 | El usuario en las nueva interfaz deberá llenar los datos necesarios para realizar la simulación | Se detecta los datos y se realiza una validación |
| | 5 | El usuario deberá presionar el botón play | Se realiza la simulación |
| Post condición | Por consiguiente se valida los datos y se realiza la simulación | | |

**Tabla 3.6:** Casos de uso cambiar a movimiento rectilíneo uniformemente variado.

| **Casos de uso: Cambiar a movimiento rectilíneo uniformemente variado** | **Nro.: 5** | | |
|---|---|---|---|
| Descripción: **casos de uso para pasar al modo de simulación MRUV** | | | |
| Actores: | Usuario general (Estudiante-profesor) | | |
| Condiciones de entrada | En el menú principal debe haberse elegido la opción “MRUV” | | |
| Flujo de evento normal | **Paso** | **Estrada del actor** | **Respuesta del sistema** |
| | 1 | Escoger el icono y nombre respecto al simulador de cinemática | Se inicia la instalación, si no existe dificultades se instala correctamente |
| | 2 | | Se inicia la aplicación con la interfaz principal |

| | | | |
|---|---|---|---|
| | 3 | El usuario debe escoger la opción MRUV | Al elegir la opción se generara una interfaz que nos permitirá realizar la respectiva simulación. |
| | 4 | El usuario en las nueva interfaz deberá llenar los datos necesarios para realizar la simulación | Se detecta los datos y se realiza una validación |
| | 5 | El usuario de presionar el botón play | Se realiza la simulación |
| Post condición | Por consiguiente se valida los datos y se realiza la simulación | | |

**Tabla 3.7:** Casos de uso cambiar a Caída Libre.

| Casos de uso: **Cambiar a caída libre** | | | Nro.: **6** |
|---|---|---|---|
| Descripción: **casos de uso para pasar al modo de simulación caída libre** | | | |
| Actores: | Usuario general (Estudiante-profesor) | | |
| Condiciones de entrada | En el menú principal debe haberse elegido la opción “Caída Libre” | | |
| Flujo de evento normal | **Paso** | **Estrada del actor** | **Respuesta del sistema** |
| | 1 | Escoger el icono y nombre respecto al simulador de cinemática | Se inicia la instalación, si no existe dificultades se instala correctamente |
| | 2 | | Se inicia la aplicación con la interfaz principal |
| | 3 | El usuario debe escoger la opción “Caída Libre” | Al elegir la opción se generara una interfaz que nos permitirá realizar la respectiva simulación. |
| | 4 | El usuario en las nueva interfaz deberá llenar los datos necesarios para realizar la simulación | Se detecta los datos y se realiza una validación |
| | 5 | El usuario de presionar el botón play | Se realiza la simulación |
| Post condición | Por consiguiente se valida los datos y se realiza la simulación | | |

**Tabla 3.8:** Casos de uso cambiar a movimiento parabólico.

| Casos de uso: **Cambiar a movimiento parabólico** | | | Nro.: **7** |
|---|---|---|---|
| Descripción: **casos de uso para pasar al modo de simulación MP** | | | |
| Actores: | Usuario general (Estudiante-profesor) | | |
| Condiciones de entrada | En el menú principal debe haberse elegido la opción "MP" | | |
| Flujo de evento normal | **Paso** | **Estrada del actor** | **Respuesta del sistema** |
| | 1 | Escoger el icono y nombre respecto al simulador de cinemática | Se inicia la instalación, si no existe dificultades se instala correctamente |
| | 2 | | Se inicia la aplicación con la interfaz principal |
| | 3 | El usuario debe escoger la opción MP | Al elegir la opción se generara una interfaz que nos permitirá realizar la respectiva simulación. |
| | 4 | El usuario en las nueva interfaz deberá llenar los datos necesarios para realizar la simulación | Se detecta los datos y se realiza una validación |
| | 5 | El usuario de presionar el botón play | Se realiza la simulación |
| Post condición | Por consiguiente se valida los datos y se realiza la simulación | | |

**Tabla 3.9:** Casos de uso cambiar a movimiento circular uniforme.

| Casos de uso: **Cambiar a movimiento circular uniforme** | | | Nro.: **8** |
|---|---|---|---|
| Descripción: **casos de uso para pasar al modo de simulación MCU** | | | |
| Actores: | Usuario general (Estudiante-profesor) | | |
| Condiciones de entrada | En el menú principal debe haberse elegido la opción "MCU" | | |
| Flujo de evento normal | **Paso** | **Estrada del actor** | **Respuesta del sistema** |
| | 1 | Escoger el icono y nombre respecto al simulador de cinemática | Se inicia la instalación, si no existe dificultades se instala correctamente |
| | 2 | | Se inicia la aplicación con la interfaz principal |
| | 3 | El usuario debe escoger la opción MCU | Al elegir la opción se generara una interfaz que nos permitirá realizar la respectiva simulación. |

| | | | |
|---|---|---|---|
| | 4 | El usuario en las nueva interfaz deberá llenar los datos necesarios para realizar la simulación | Se detecta los datos y se realiza una validación |
| | 5 | El usuario de presionar el botón play | Se realiza la simulación |
| Post condición | Por consiguiente se valida los datos y se realiza la simulación | | |

### 3.3.1. ASPECTOS TÉCNICOS

La aplicación funcionara en ordenadores ya sea de escritorio o portátiles. Y un data show si se desea proyectar. Para que el proceso funcione adecuadamente serán necesario las siguientes características de Hardware:

- Memoria RAM mínima 512 Mb.
- Tarjeta de video con capacidades DX9 "shader modelo 2.0".
- CPU: compatible con el conjunto de instrucciones SSE2.
- Memoria libre mínima de 200 Mb El software recomendado deberá cumplir con el siguiente requisito:
- Sistema operativo Windows XP SP2 o superior.

## 3.4. ANÁLISIS Y DISEÑO INICIAL

### 3.4.1. REQUISITOS FUNCIONALES Y NO FUNCIONALES

La aplicación deberá permitir realizar simulación de los distintos tipos de movimiento que abarca el tema de Cinemática. Con tan solo introducir los datos necesarios para realizar esta acción, entre los requisitos están:

- Deberá ser amigable para los usuarios y no ser complejo para no llegar a confundir usuario.
- Deberá recolectar datos numéricos.
- Nos tendrá que proporcionar objetos 3D para realizar la simulación
- Tener opciones para iniciar pausar y restaurar la simulación.
- Tendrá q reportar datos de la simulación
- El simulador hará uso del sistema internacional en cuanto al manejo de las unidades

En cuanto a las no funcionales tenemos:

- Solo realizara simulaciones del nivel de aprendizaje en colegiatura.
- No realizara conversiones de unidades
- La simulación no se realizara en tiempo real.

### 3.4.2. DISPOSITIVOS DE ENTRADA Y SALIDA

Para el desarrollo de la aplicación se necesitara la ayuda del motor de videojuegos UNITY 3D y el del framework que proporciona Unity 3D permite el uso de dispositivos de entrada y salida las cuales son:

**Dispositivos de entrada**

Teclado y mouse: son dispositivos que sirven para la entrada de información, el usuario podrá interactuar mediante selección en las distintas opciones que se brindaran.

**Dispositivos de salida**

Pantalla: Para la salida de información, el usuario podrá visualizar el menú inicio, modelos 3D y otras características de la aplicación.

### 3.4.3. ARQUITECTURA DEL SOFTWARE

Arquitectura base sobre la cual se desarrollara el simulador.

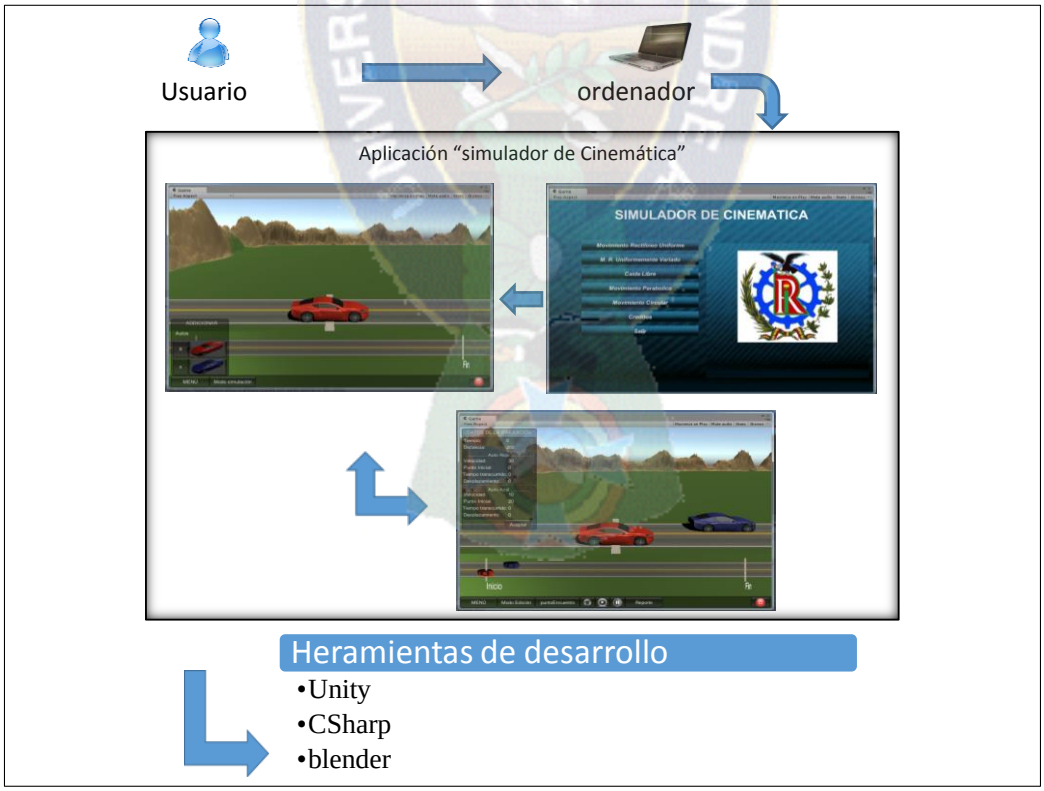



**Figura 3.3:** Arquitectura de desarrollo.

### 3.4.4. DISEÑO DE COMUNICACIÓN GENERAL DEL PRODUCTO

Se realizara el diseño de las interfaces que serán zonas de comunicación en las que se realiza la interacción entre el usuario y la aplicación, teniendo así una mejor comprensión del tema que elija el estudiante con los modelos 3D además para el diseño de las mismas se tomaran en cuenta los dispositivos de entrada y salida de la aplicación, haciendo uso de botones y cuadros de texto como se observa en la figura 3.4.


**Figura 3.4:** Diseño comunicativo.

Descripción de la interface:

Menú principal. En su interfaz se muestra las opciones a elegir para realizar los distintos tipos de simulación que realizaras la aplicación entre las cuales esta:

- Opción “MRU ”
- Opción “MRUV”
- Opción “CL”
- Opción “MP”
- Opción “MC”

Tales opciones permiten realizar una simulación y verificar lo que sucede en el problema propuesto, la actualización de las variables simultáneamente al realizar la simulación. Y por lo tanto tener una mejor comprensión del ejercicio realizado.

## 3.5. PLAN DE ITERACIONES

### 3.5.1. DISEÑO DE LAS ITERACIONES

**Primera iteración**

En la primera iteración se realizaran las tareas que permitan desarrollar el modulo del movimiento rectilíneo uniforme, las cuales son las siguientes:

- Realizar los modelos 3D necesarios
- Crear una escena en el que se trabajara la simulación del MRU
- Incluir una cámara con su respectivo script
- Realizar los script necesarios que permita realizar la simulación.

**Segunda iteración**

En la segunda iteración se realizara las tareas que permitan realizar simulaciones del módulo de movimiento rectilíneo uniformemente variado, las cuales son las siguientes:

- Realizar los modelos 3D necesarios
- Crear una escena en el que se trabajara la simulación del MRUV
- Incluir una cámara con su respectivo script
- Realizar los script necesarios que permita realizar la simulación.

**Tercera iteración**

Esta iteración está destinada al desarrollo del módulo de caída Libre. Las cuales conllevan a la realización de las siguientes tareas:

- Realizar los modelos 3D necesarios.
- Crear una escena en el que se trabajara la simulación de caída libre.
- Incluir una cámara con su respectivo script.
- Realizar los script necesarios que permita realizar la simulación.

**Cuarta iteración**

La cuarta iteración está destinada para el módulo de movimiento parabólico la cual permitirá realizar simulaciones de tal movimiento, para lo cual se deberá realizar las siguientes tareas:

- Realizar los modelos 3D necesarios.

- Crear una escena en el que se trabajara la simulación del movimiento parabólico.
- Incluir una cámara con su respectivo script.
- Realizar los script necesarios que permita realizar la simulación.

**Quinta iteración**

En la quinta iteración se realizara las tareas que permitan realizar la simulación del movimiento circular uniforme.

- Realizar los modelos 3D necesarios.
- Crear una escena en el que se trabajara la simulación del movimiento circular uniforme.
- Incluir una cámara con su respectivo script.
- Realizar los script necesarios que permita realizar la simulación.

**Sexta iteración**

En una sexta iteración se procede a la integración todos los módulos que permiten realizar las distintas simulaciones, para tal caso se realizara las siguientes tareas.

- Realizar un menú que permita seleccionar los distintos tipos de simulación
- Realizar una opción que brinde ayuda sobre el uso de la aplicación

### 3.5.2. PRIORIZACIÓN DE LAS ITERACIONES

Para que la aplicación sea funcional en cada avance que se realiza deberemos priorizar las iteraciones listados anteriormente. Para que cada avance realizado esta sea totalmente funcional de los respectivos módulos, por lo tanto las iteraciones a realizarse primeramente serán:

1. Sexta iteración
2. Primera iteración
3. Segunda iteración
4. Tercera iteración
5. Cuarta iteración
6. Quinta iteración.

Esta priorización permite manipular la aplicación con cada módulo terminado. Ya que en sexta iteración se realiza el menú principal, y las otras iteración están ordenadas en el orden que el profesor dicta su clase.

## 3.6. DISEÑO COMPUTACIONAL

### 3.6.1. PLAN DE TRABAJO DE LA ITERACIÓN

En cada iteración se debe realiza las siguientes tareas para la buena culminación de estas, para lo cual se necesitara un programador y un diseñador.

**El diseñador**. Deberá realizar el diseño de los modelos 3D necesarios en cada módulo el cual ara uso de la herramienta Blender para su desarrollo.

**El programador**. Realizara los script necesarios que permita gestionar los objetos 3D para poder realizar la respectiva simulación. Entre las cuales tenemos scripts que permiten realizar las siguientes acciones.

- Manipular cámaras
- Manipular GameObject
- Manipular el terreno en el cual se realiza la simulación

Una de las tareas que realizara el programador es el menú principal la cual integrara todos los modules de simulaciones, y brindara una ayuda respecto al uso de la misma esta acción la realizara conjuntamente con el diseñador, el cual brindara del material para su desarrollo.

### 3.6.2. ELABORAR EL DISEÑO COMPUTACIONAL

En el desarrollo de la aplicación se ase el uso de la programación orientado a objetos la cual permite realizar un control de cada objeto utilizado en el desarrollo del proyecto, a continuación se presenta la relación entre las clases utilizadas en el simulador que se puede observar en la figura 3.5.

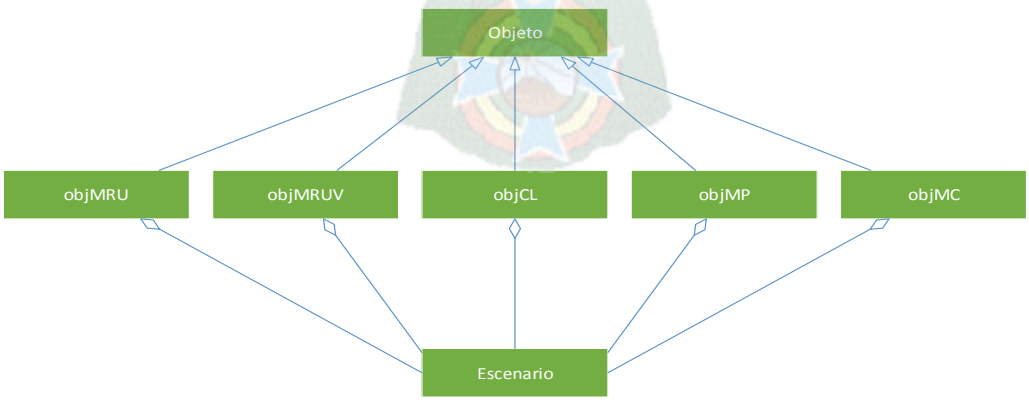



**Figura 3.5:** Relación de clases.

La descripción de cada una de las clases utilizados en el desarrollo del simulador con sus respectivos atributos y métodos son las siguientes:

**Clase Objeto.** Esta clase nos servirá para almacenar los datos que usaremos al realizar la simulación, la cual cuenta con los siguientes atributos que se puede ver en la Figura 3.6:


**Figura 3.6:** Clases Objeto.

Los atributos de la clase son:

- Velocidad**:** Es de tipo flotante que se usara para saber la velocidad que tiene el objeto a simular
- tiempoTranscurido: este atributo proporcionara el tiempo que transcurrió desde el inicio de la simulación, la cual es de tipo float.
- Desplazamiento: brinda la distancia recorrida desde el inicio de la simulación, la cual es de tipo float.
- posicionInicial: brinda la información de la posición en la que se encuentra el objeto a simular desde el origen, la cual será de tipo float.
- Sentido: brinda en qué dirección ara su recorrido.

Los métodos utilizados son los siguientes:

- Objeto(). Es el constructor de la clase la cuan inicializas las variables.
- getVelocidad() retorna la velocidad del objeto.

- getTiempo() retorna el tiempo transcurrido.
- getDistancia() retorna el desplazamiento del objeto.
- getPosicionInicial() retorna la posición inicial del objeto.
- getSentido() retorna la dirección del objeto.
- setVelocidad() retorna la velocidad del objeto.
- setTiempo() retorna el tiempo transcurrido.
- setDistancia() retorna el desplazamiento del objeto.
- setPosicionInicial() retorna la posición inicial del objeto.
- setSentido() retorna la dirección del objeto.

**Clase objMRU**. Sus atributos y métodos se muestran en la Figura 3.7 y la descripción de cada uno es la siguiente:


**Figura 3.7:** Clase ObjMRU.

Atributos:

- velocidadFinal. Proporciona la velocidad final que tiene el objeto durante la simulación de tipo float.

Métodos:

- ObjMRU(). Es el constructor de la clase.
- Leer(). El método recolecta los datos necesarios para realizar la simulación.
- validarDato(). Valida los datos recolectados cumpliendo con la condición que el dato introducido sea de tipo numérico.

- prepararDatos(). Realiza los respectivos cálculos para poder realizar la simulación.
- Simular(). Procede a realizar la simulación. actualizando los datos cada vez que se actualización los frames.
- Reporte().Brinda los datos actualizados después de la simulación.

**Clase objMRUV.** Los atributos y métodos se muestran en la Figura 3.8 y la descripción de cada uno es la siguiente:

Atributos:

- velocidadFinal. Proporciona la velocidad final que tiene el objeto durante la simulación la cual es de tipo float.
- Aceleración. Es la variable que almacenar la aceleración del objeto a simular de tipo float.
- Acelerado. Variable de tipo boleano que servirá para saber si se realizara una tramo de simulación acelera o no.
- Uniforme. Variable de tipo boleano que servirá para saber si se realizara una tramo de simulación uniforme o no.
- Desacelerado. Variable de tipo boleano que servirá para saber si se realizara una tramo de simulación desacelerado o no.

Métodos:

- ObjMRUV(). Es el constructor de la clase.
- Leer(). El método recolecta los datos necesarios para realizar la simulación.
- validarDato(). Valida los datos recolectados cumpliendo con la condición que el dato introducido sea de tipo numérico.
- prepararDatos(). Realiza los respectivos cálculos para poder realizar la simulación.
- Simular(). Procede a realizar la simulación. actualizando los datos cada vez que se actualización los frames.
- Reporte().Brinda los datos actualizados después de la simulación.


**Figura 3.8:** Clase ObjMRUV.

**Clase objCL.** Los atributos y métodos se muestran utilizados en esta clases se las puede observar en la Figura 3.9 y la descripción de cada uno es la siguiente:

Atributos:

- velocidadFinal. Proporciona la velocidad final que tiene el objeto durante la simulación la cual es de tipo float.
- gravedad. Es la variable que almacenar el valor de la gravedad de tipo float.
- tiempoSubida. Variable de tipo float que servirá para almacenar el valor te tiempo de subida empleado.
- tiempoBajda. Variable de tipo float que servirá para almacenar el tiempo de bajada empleado del objeto.
- alturaMaxima. Variable de tipo float que servirá para saber la altura máxima que se alcanso.

Métodos:

- ObjCL(). Es el constructor de la clase.
- Leer(). El método recolecta los datos necesarios para realizar la simulación.
- validarDato(). Valida los datos recolectados cumpliendo con la condición que el dato introducido sea de tipo numérico.
- prepararDatos(). Realiza los respectivos cálculos para poder realizar la simulación.
- Simular(). Procede a realizar la simulación. actualizando los datos cada vez que se actualización los frames.
- Reporte(). Brinda los datos actualizados después de la simulación.


**Figura 3.9:** Clase ObjCL.

**Clase objMP.** Esta clase cuenta con atributos y métodos que se muestran en la Figura 3.10 y la cual permite gestionar las variables que permitirán realizar la simulación de dicho movimiento:

Atributos:

- velocidadFinal. Proporciona la velocidad final que tiene el objeto durante la simulación la cual es de tipo float.
- gravedad. Es la variable que almacenar el valor de la gravedad de tipo float.
- tiempoSubida. Variable de tipo float que servirá para almacenar el valor te tiempo de subida empleado.
- tiempoBajda. Variable de tipo float que servirá para almacenar el tiempo de bajada empleado del objeto.
- alturaMaxima. Variable de tipo float que servirá para saber la altura máxima que se alcanso.
- Angulo. De tipo float, la cual sirve para saber con qué inclinación se realizas el disparo.

Métodos:

- ObjMP(). Es el constructor de la clase.
- Leer(). El método recolecta los datos necesarios para realizar la simulación.
- validarDato(). Valida los datos recolectados cumpliendo con la condición que el dato introducido sea de tipo numérico.
- prepararDatos(). Realiza los respectivos cálculos para poder realizar la simulación.

- Simular(). Procede a realizar la simulación. actualizando los datos cada vez que se actualización los frames.
- Reporte().Brinda los datos actualizados después de la simulación.


**Figura 3.10:** Clase ObjMP.

**Clase objMC.** Los atributos y métodos de esta clase se muestran en la Figura 3.11 y la descripción de cada uno es la siguiente:

Atributos:

- Frecuencia. De tipo float brinda información del número de oscilaciones por segundo.
- velocidadAngular. Proporciona la velocidad angular que tiene el objeto durante la simulación la cual es de tipo float.
- DesplazamientoAngular. De tipo float, la cual sirve para saber el desplazamiento angular.

Métodos:

- ObjMC(). Es el constructor de la clase.
- Leer(). El método recolecta los datos necesarios para realizar la simulación.
- validarDato(). Valida los datos recolectados cumpliendo con la condición que el dato introducido sea de tipo numérico.
- prepararDatos(). Realiza los respectivos cálculos para poder realizar la simulación del movimiento circular.

- Simular(). Procede a realizar la simulación. actualizando los datos cada vez que se actualización los frames.
- Reporte(). Brinda los datos actualizados después de la simulación.

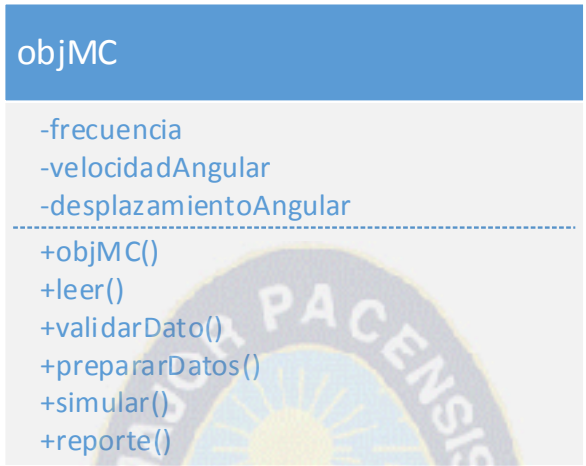



**Figura 3.11:** Clase ObjMC.

**Clase escenario.** Permite la creación de un terreno en la cual se realizar todas las simulaciones la cual cuenta con los atributos y métodos necesarios que permita realizar un control de esta que se muestran en la Figura 3.12

Atributos:

- tiempo. De tipo float, que brinda el tiempo que se realizará la simulación.
- distancia. De tipo float, que brinda la distancia que recorrerá al realizar la simulación.
- Objeto3D. Variable de tipo GameObject que el modelo que se usara para la simulación.

Métodos:

- escenario(). Es el constructor de la clase.
- Leer(). El método nos recolecta los datos necesarios para realizar la simulación.
- validarDato(). Valida los datos recolectados cumpliendo con la condición que el dato introducido sea de tipo numérico.
- prepararDatos(). Realiza los respectivos cálculos para poder realizar la simulación.
- Reporte(). Brinda los datos actualizados después de la simulación.


**Figura 3.12:** Clase escenario.

### 3.6.3. PROTOTIPO DE INTERFAZ

La aplicación usara varias interfaces. Muchas de ellas cambian dependiendo de la opción escogida. Para facilitar el diseño y creación del escenario deseado, se crearon mockups. En la creación de los mismos se tiene en cuenta los siguientes puntos:

- Correcta gramática y ortografía: Cualquier falla gramatical u ortográfica es imperdonable para una aplicación con fines educativos.
- Uso de colores cálidos y claros: Se eligieron colores como el rojo y verde, en tonos cálidos para que sean llamativos para el usuario.
- Selección de fuentes adecuadas: Las selección de las fuentes se hizo tras una comparación y búsqueda de las mismas, para lo cual se eligieron las fuentes "calibri". A continuación se describen las interfaces:

**Menú principal:** Una vez finalizada la carga de la aplicación, se muestra el menú principal con siete botones la cual se observa en la figura 3.13.

- Botón "MRU": Permitirá cambiar a la interfaz del Movimiento rectilínea uniforme.
- Botón "MRUV": Permitirá cambiar a la interfaz del MRUV.
- Botón "CL": Permitirá pasar a la interfaz de caída libre.
- Botón "MP": Permite cambiar a la interfaz del movimiento parabólico.
- Botón "MC": Permite pasar a la interfaz del movimiento circular uniforme.
- Botón "ayuda": Brinda una descripción de cómo usar la aplicación.
- Botón "salir": Opción que permite cerrar la aplicación.


**Figura 3.13:** Mockup del menú principal.

**Pantalla de ayuda:** Se accederá tras presionar el botón de "Ayuda" del menú principal. La ayuda o instrucciones sobre el uso del sistema lo cual se puede observar en la Figura 3.14

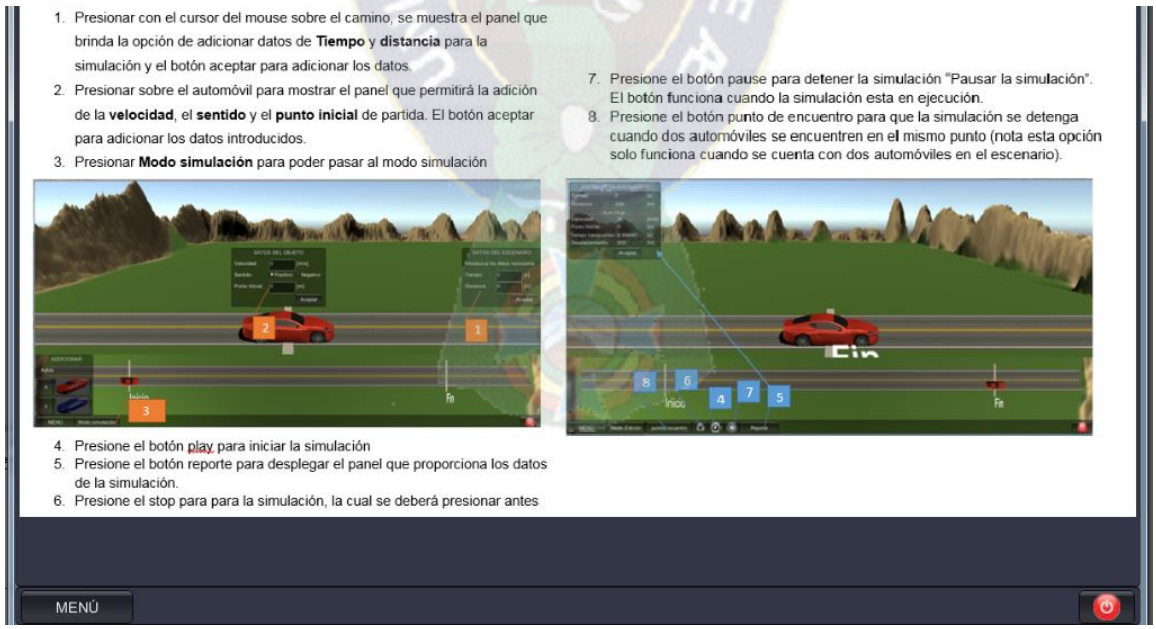



**Figura 3.14:** Mockup de la pantalla de ayuda.

**Pantalla de MRU:** Se accederá tras presionar el botón de "Movimiento rectilíneo uniforme" del menú principal. La cual proporcionara la funcionalidad de poder realizar simulaciones, la cual se muestra en la Figura3.15.


**Figura 3.15:** Mockup de la pantalla del MRU.

- El panel ADICIONAR permite adicionar y quitar modelos 3D de autos
- El botón menú brinda la opción de regresa a la pantalla principal.
- El botón con el icono que representa apagar, cierra la aplicación.
- El botón Modo simulación pasa a otra pantalla con opciones para realizar la simulación, la cual se muestra en la Figura 3.16.


**Figura 3.16:** Mockup de la pantalla del MRU en modo Simulación.

- El panel DATOS DEL OBJETO brinda la opción de poder introducir los datos del objeto la cual sirve para realizar la simulación.
- El panel DATOS DEL ESCENARIO brinda la opción de introducir datos a la escena para realizar la simulación.
- Modo edición. Permite regresa a la pantalla principal del MRU que se ve en la Figura 3.15.
- Punto de encuentro. Es la opción que permite parar la simulación cuando dos objetos se encuentra en puntos iguales en un determinado tiempo.
- Stop. Permite parar y restaurar la simulación.
- Play. Permite iniciar la simulación.
- Pause. Pausa la ejecución.
- Reporte. Brinda la información de la simulación.

**Pantalla de MRUV:** Se accederá tras presionar el botón de "Movimiento rectilíneo uniformemente variado" del menú principal. La cual proporcionara la funcionalidad de poder realizar simulaciones, la cual se muestra en la Figura 3.17.


**Figura 3.17:** Mackup de la pantalla del MRUV.


**Figura 3.18:** Mockup de la pantalla del MRUV en modo Simulación.

- El panel ADICIONAR permite adicionar y quitar modelos 3D de autos
- El botón menú regresa a la pantalla principal.
- El botón con el icono que representa apagar, cierra la aplicación.
- El botón Modo simulación pasa a otra pantalla con opciones para realizar la simulación, la cual se muestra en la Figura 3.18.
- El panel DATOS DEL OBJETO. Brinda la opción de poder introducir los datos del objeto la cual sirve para realizar la simulación.
- El panel DATOS DEL ESCENARIO. Brinda la opción de introducir datos a la escena para realizar la simulación.
- Modo edición. Regresa a la pantalla principal del MRUV que se ve en la Figura 3.17.
- Punto de encuentro. la opción de parar la simulación cuando dos objetos se encuentra en puntos iguales en un determinado tiempo.
- Stop. Permite parar y restaurar la simulación.
- Play. Permite iniciar la simulación.
- Pause. Pausa la ejecución.
- Reporte. Brinda la información de la simulación.

**Pantalla de CL:** Se accederá tras presionar el botón de "caída libre" del menú principal. La cual proporcionara la funcionalidad de poder realizar simulaciones para el movimiento de caída libre, la cual se observa en la Figura 3.19.

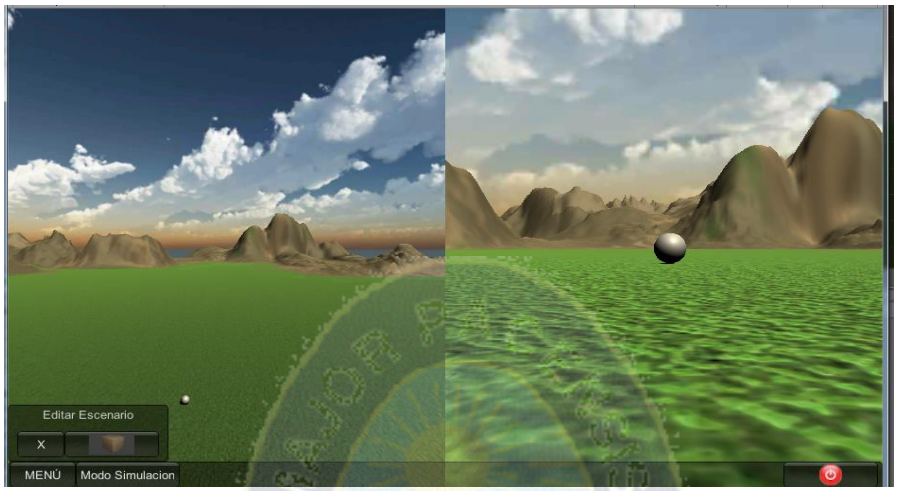



**Figura 3.19:** Mockup de la pantalla de CL.

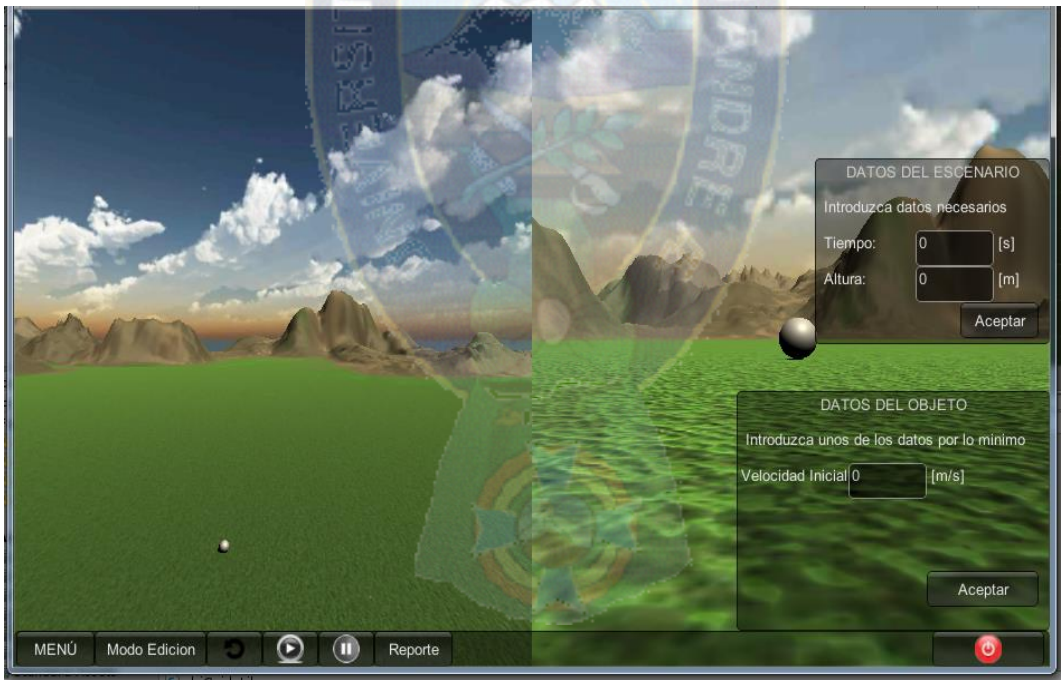



**Figura 3.20:** Mockup de la pantalla de CL en modo Simulación.

- El panel ADICIONAR permite adicionar y quitar modelos 3D
- El botón menú regresa a la pantalla principal.
- El botón con el icono que representa apagar, cierra la aplicación.

- El botón Modo simulación pasa a otra pantalla con opciones para realizar la simulación, la cual se muestra en la Figura 3.20.
- El panel DATOS DEL OBJETO brinda la opción de poder introducir los datos del objeto la cual sirve para realizar la simulación.
- El panel DATOS DEL ESCENARIO brinda la opción de introducir datos a la escena para realizar la simulación.
- Modo edición regresa a la pantalla principal del CL que se ve en la Figura 3.19.
- Restaurar  permite parar y restaurar la simulación.
- Play permite iniciar la simulación.
- Pause. Pausa la ejecución.
- Reporte brinda la información de la simulación.

**Pantalla de MP:** Se accederá tras presionar el botón de "Movimiento parabólico" del menú principal. La cual proporcionara la funcionalidad de poder realizar simulaciones, la cual se muestra en la Figura 3.21.

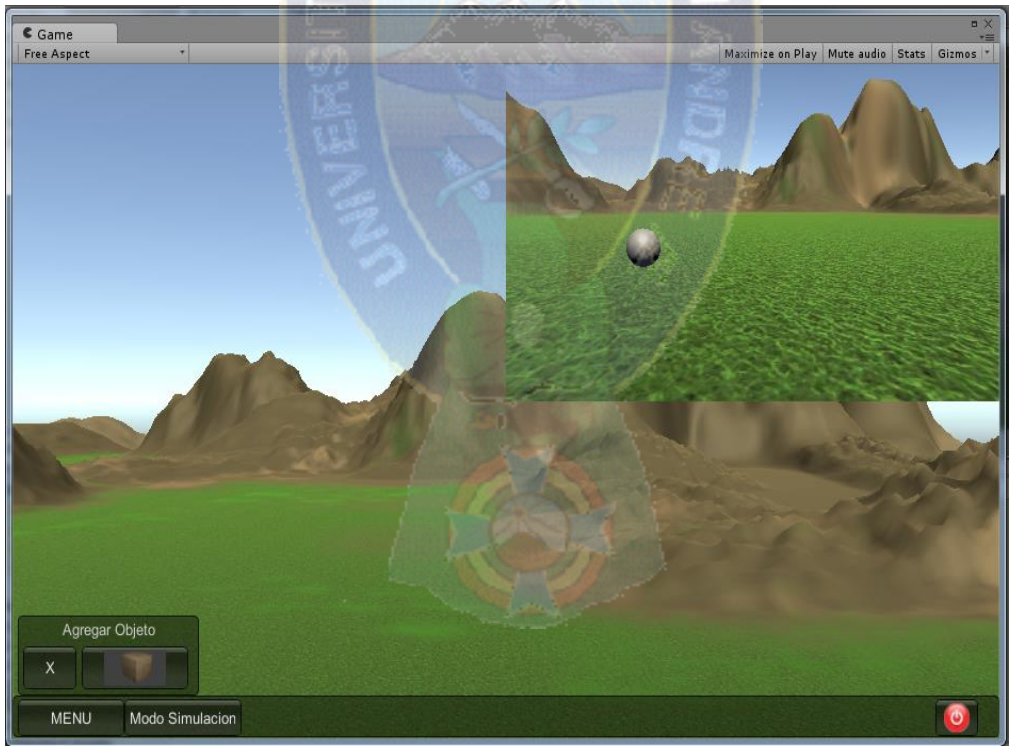



**Figura 3.21:** Mockup de la pantalla de MP.

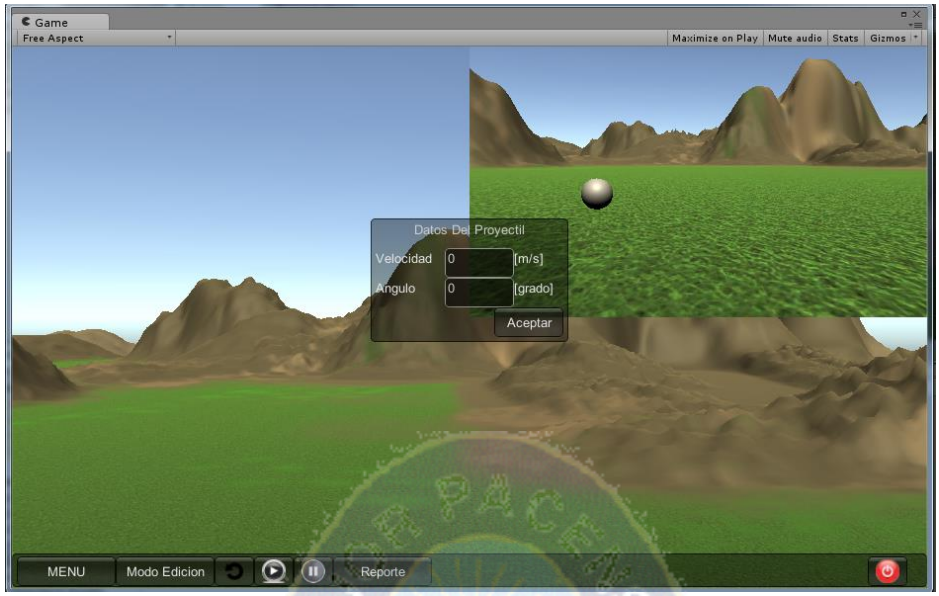



**Figura 3.22:** Mockup de la pantalla de MP en modo Simulación.

- El panel ADICIONAR Permite adicionar y quitar modelos 3D
- El botón menú regresa a la pantalla principal.
- El botón con el icono que representa apagar, cierra la aplicación.
- El botón Modo simulación pasa a otra pantalla con opciones para realizar la simulación, la cual se muestra en la Figura 3.22.
- El panel DATOS DEL PROYECTIL brinda la opción de poder introducir los datos del objeto la cual sirve para realizar la simulación.
- El panel DATOS DEL ESCENARIO brinda la opción de introducir datos a la escena para realizar la simulación.
- Modo edición regresa a la pantalla principal del MP que se ve en la Figura 3.21.
- Restaurar permite parar y restaurar la simulación.
- Play permite iniciar la simulación.
- Pause. Pausa la ejecución.
- Reporte brinda la información de la simulación.

**Pantalla de MCU:** A esta opción se accederá tras presionar el botón de "Movimiento circular" del menú principal. La cual proporcionara la funcionalidad de poder realizar simulaciones, la cual se muestra en la Figura 3.23.

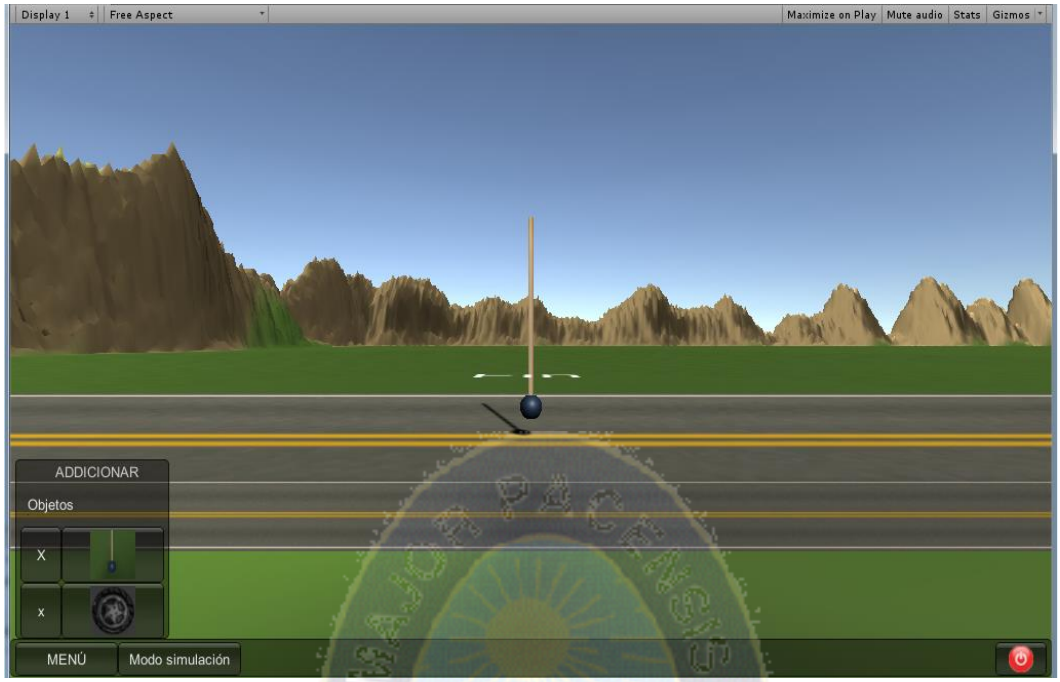



**Figura 3.23:** Mockup de la pantalla de MCU.


**Figura 3.24:** Mockup de la pantalla de MCU en modo Simulación.

- El panel ADICIONAR permite adicionar y quitar modelos 3D

- El botón menú regresa a la pantalla principal.
- El botón con el icono que representa apagar, cierra la aplicación.
- El botón Modo simulación pasa a otra pantalla con opciones para realizar la simulación, la cual se muestra en la Figura 3.24.
- El panel DATOS DEL PROYECTIL brinda la opción de poder introducir los datos del objeto la cual sirve para realizar la simulación.
- El panel DATOS DEL ESCENARIO brinda la opción de introducir datos a la escena para realizar la simulación.
- Modo edición regresa a la pantalla principal del MCU que se ve en la Figura 3.23.
- Restaurar permite parar y restaurar la simulación.
- Play permite iniciar la simulación.
- Pause Pausa la ejecución.
- Reporte brinda la información de la simulación.

## 3.7. DESARROLLO

### 3.7.1. DESARROLLO DE COMPONENTES

**Creación de los Modelos 3D:**

Los modelos 3D que serán mostrados en los diferentes módulos del simulador. El diseño del modelo 3D está basado en el dibujo de objetos necesarios para la aplicación. Se tomaron en cuenta las siguientes consideraciones para la creación de los modelos 3D:

- Fáciles de reconocer: Los modelos son simples y con un diseño claro, sin elementos innecesarios que compliquen su reconocimiento y visualización.
- Diseños simples: Los modelos no exceden el uso de polígonos, puntos y superficies subdivididas, para cuidar el rendimiento de la aplicación.
- Materiales coloreados: Se usan materiales coloreados en lugar de texturas, tomando en cuenta que no es necesario que los modelos tengan apariencia realística, y también para evitar consumir demasiada memoria por el almacenamiento de las texturas.

En la Figura 3.25. Se describe el modelo de un automóvil que será mostrado en el simulador. Para empezar el modelado se tomó como base figuras primitivas de un plano para el diseño del automóvil, por siguiente se realiza las divisiones de superficies para luego darle forma y color de un automóvil real.

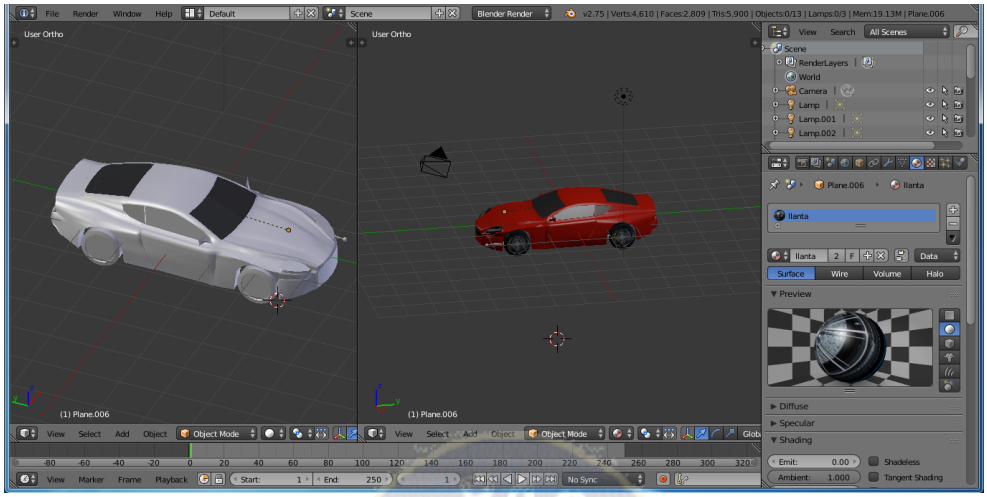


**Figura 3.25:** Modelo 3D del automóvil.

Al concluir el modelamiento del automóvil se desarrollamos el script necesario para que el automóvil tenga la animación necesaria para la simulación. La cual nos permitirá interactuar con el usuario, ya que el script ara uso de los datos proporcionados por el usuario y esta las procesara para tener el resultado deseado.

También se realiza los modelos del escenario la cual se muestra en la Figura 3.26 que se modelo con el framework que proporciona Unity, la cual servirá para realizar las simulaciones dentro de este escenario.

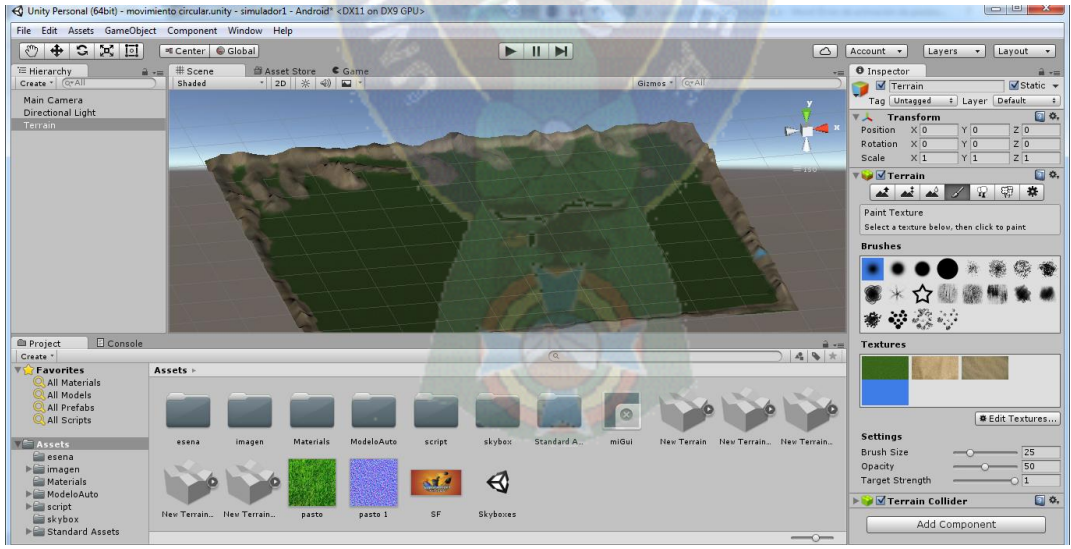


**Figura 3.26:** Modelo 3D del escenario de trabajo.

Modelando una carretera la cual se ve en la Figura 3.27 este modelo se la realizo con Unity haciendo uso de un plano, con una material de una carretea.

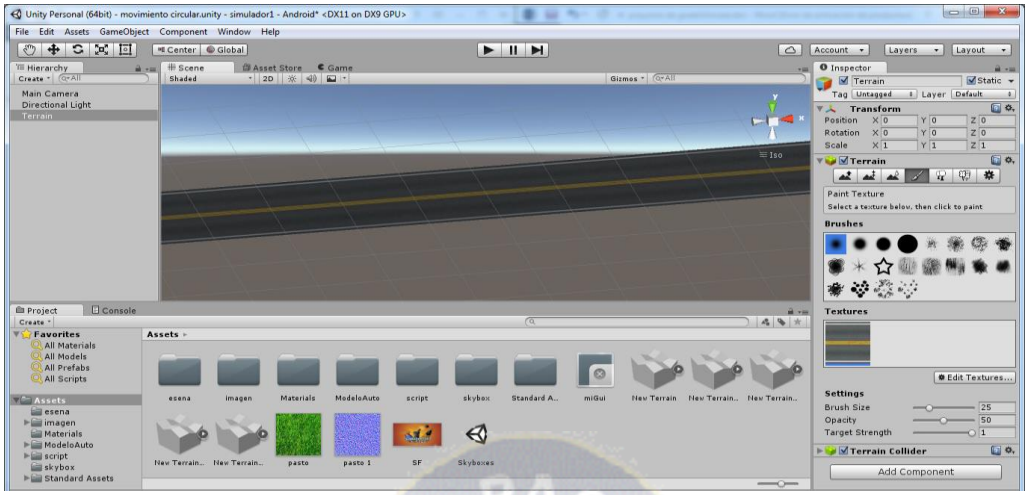


**Figura 3.27:** Modelo de carretera.

Y por último procedemos a crear una cámara que se muestra en la Figura 3.28, la cual servirá para poder visualizar todos los modelos que se utilizara al ejecutarlo. A su vez se realiza el script que se encargara de controlar tal cama la cual generara una interfaz de usuario con la cual se podra controlar los modelos 3D y realizar la simulación.

**Figura 3.28:** Cámara que permitirá visualizar el escenario.

### 3.7.2. PRUEBA DE COMPONENTES

Al culminar con el desarrollo de los componentes que permitirán cumplir con los objetivos del proyecto. Ahora se procede a comprobar la ejecución de cada uno de ellos, las cuales podemos verlos des de las Figuras 3.29 hasta la Figura 3.32.

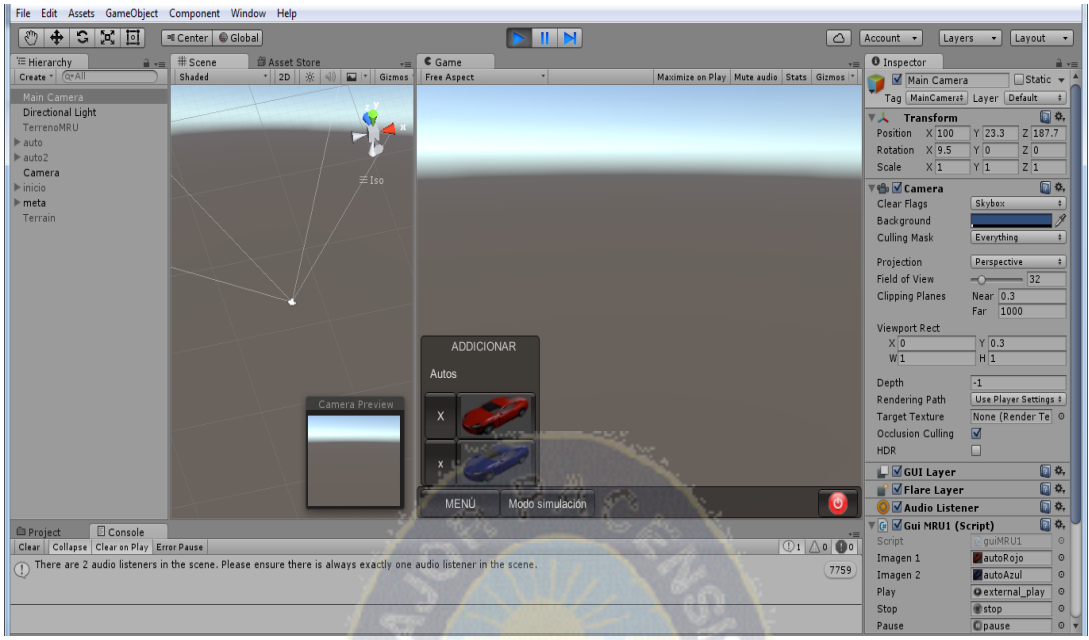



**Figura 3.29:** Ejecución de la cámara con el respectivo GUI.


**Figura 3.30:** Ejecución de los modelos de Automóviles.

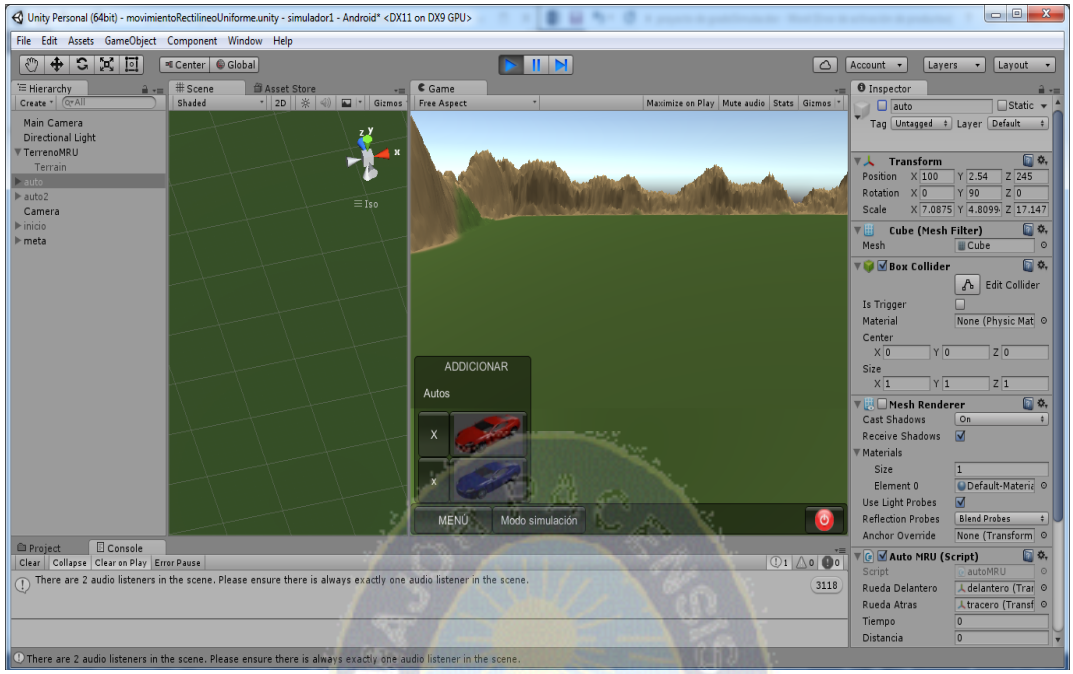



**Figura 3.31:** Ejecución de escenario sobre la cual se ara la simulación.

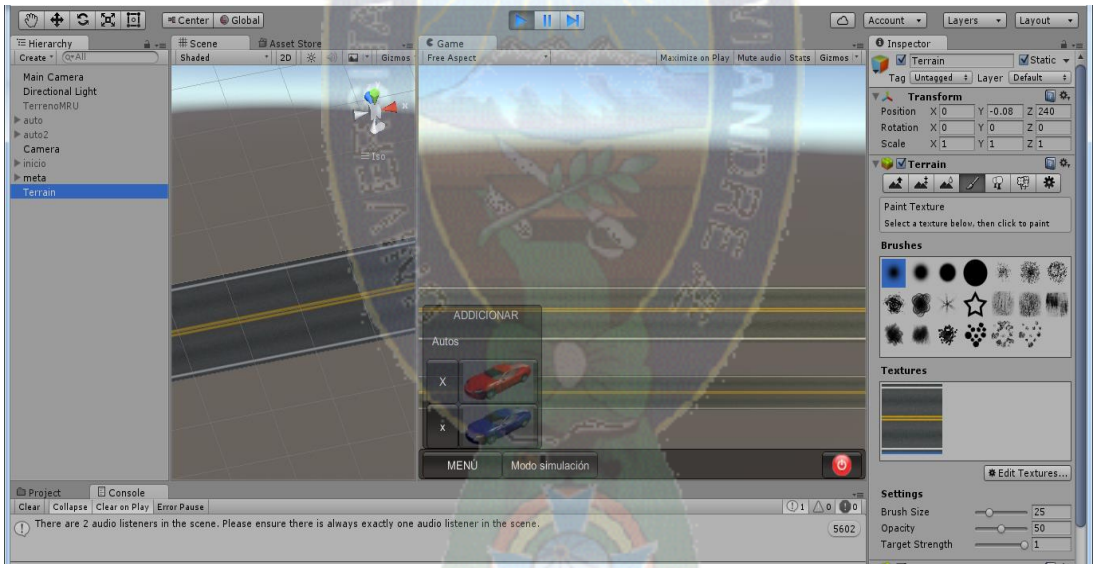



**Figura 3.32:** Ejecución de la carretera con vista ortográfica y perspectiva.

### 3.7.3. INTEGRAR EL DESARROLLO PREVIO

Al haber culminado con las pruebas de cada componente la siguiente tarea a realizar es la integración de cada componente en uno solo para tener el funcionamiento deseado. La integración se realizara sobre el escenario que se mostró en la figura 3.26 la cual está en la página 62. Para lo cual se pondrá todos los componentes en el escenario, el resultado de esta acción se la puede ver en la Figura 3.33.

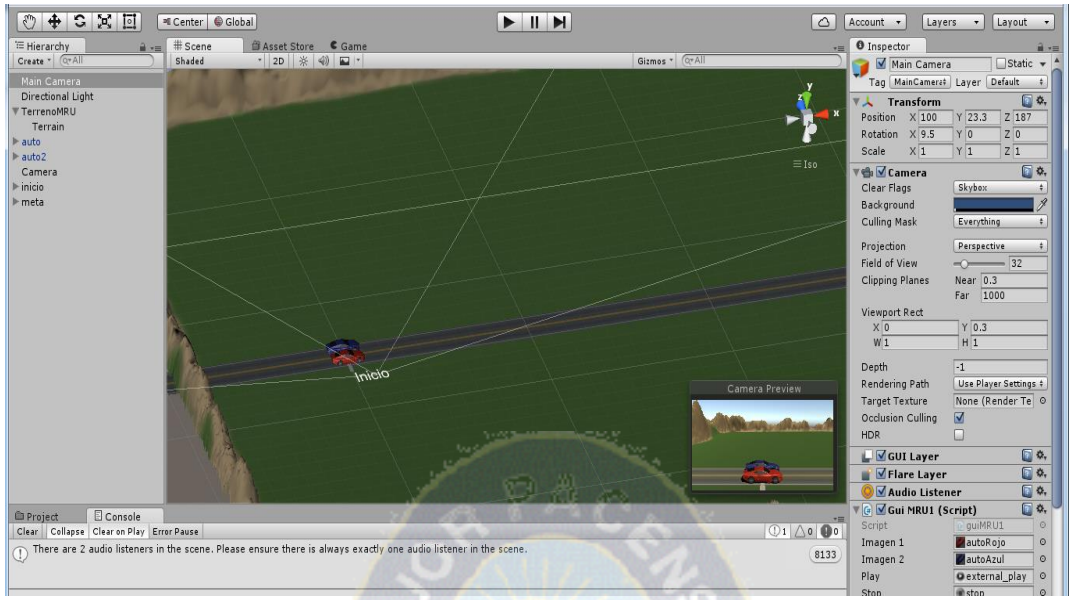



**Figura 3.33:** Integración de los componentes en el escenario.

### 3.7.4. REALIZAR PRUEBAS DE INTEGRACIÓN

La acción a realizar tras hacer realizado la respectiva integración de componentes es proceder a la verificación de esta para ver si se cumple con los objetivos esperados. La cual la podemos observar en las Figura 3.34 que brinda la opción de personalizar el escenario con la que se trabajara y en la Figura 3.35 proporciona los controladores para poder realizar la simulación.


**Figura 3.34:** Ejecución integrada en modo Edición.


**Figura 3.35:** Ejecución de la integración en modo simulación.

## 3.8.FASE DE DESPLIEGUE

Para esta fase, se tiene el proyecto prácticamente terminado por lo cual se prueba para verificar la existencia de errores en proceso de simulación, si los datos son correctamente procesados, el análisis del rendimiento y la usabilidad de la aplicación, asegurando así que sea capaz de brindar un tiempo de respuesta eficiente con el contenido educativo del tema "Cinemática".

### 3.8.1. PRUEBA DE PROCESAMIENTO DE LOS DATOS

Teniendo la aplicación terminado para cada contenido del tema "Cinemática", se procede con la prueba de procesamiento de datos a través del computador.

Con los siguientes datos se procede a realizar la simulación:

$$D = 200 \text{ [m]}$$

Datos del auto Rojo:

$$V = 30 \text{ [m/s]}$$

$$\text{Punto Inicial} = 40 \text{ [m]}$$

Datos del auto azul:

$$V = 40 \text{ [m/s]}$$

$$\text{Punto Inicial} = 0 \text{ [m]}$$

En la figura 3.36 se muestra la aplicación en funcionamiento, la simulación del MRU con sus respectivos modelos 3D. No presenta demora en el procesamiento de los datos, ya que se aseguró que la aplicación muestre datos exactos a la hora de realizar una simulación.


**Figura 3.36:** Prueba de procesamiento de datos.

### 3.8.2. PRUEBA DE LA CAJA NEGRA

Esta prueba se enfoca para identificar la existencia de problemas en cuanto al desempeño de los casos de uso definidos. Es decir aquellos que serán ejecutados, sin verificar el código fuente.

**Caso Uno:** Caso de uso Modulo Principal

Para este módulo, el actor no requiere de mucho recurso procesamiento en los tres primeros casos con los que interactúa. Simplemente la instalación y las 3 primeras interfaces de usuario.

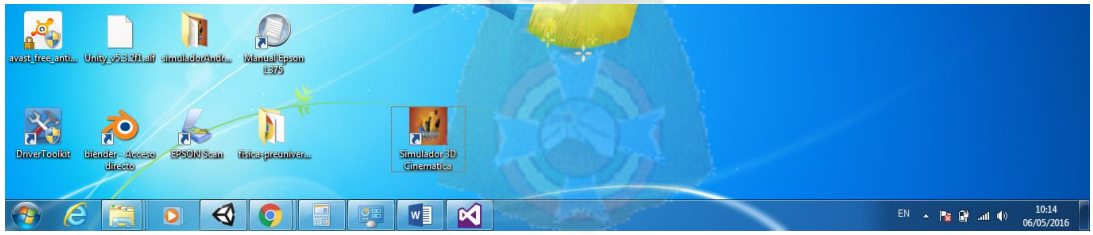


**Figura 3.37:** Aplicación correctamente Instalada.

Se observar en la Figura 3.37 que la aplicación se instaló correctamente como un icono de presentación lista para ser iniciada.

Antes de realizar las primeras simulaciones es necesario ver el contenido con el que el usuario se va encontrar la cual se la muestra en la Figura 3.38.


**Figura 3.38:** Contenido de la aplicación.

**Caso Dos:** Caso de uso simulación

El usuario interactúa directamente con la aplicación se realiza las simulaciones requeridas. Introduciendo datos necesarios para realizar la simulación. Teniendo como salida la simulación respectiva y los datos tras haber concluida la simulación, la cual se las puede ver en la figura 3.39 y en la Figura 3.40

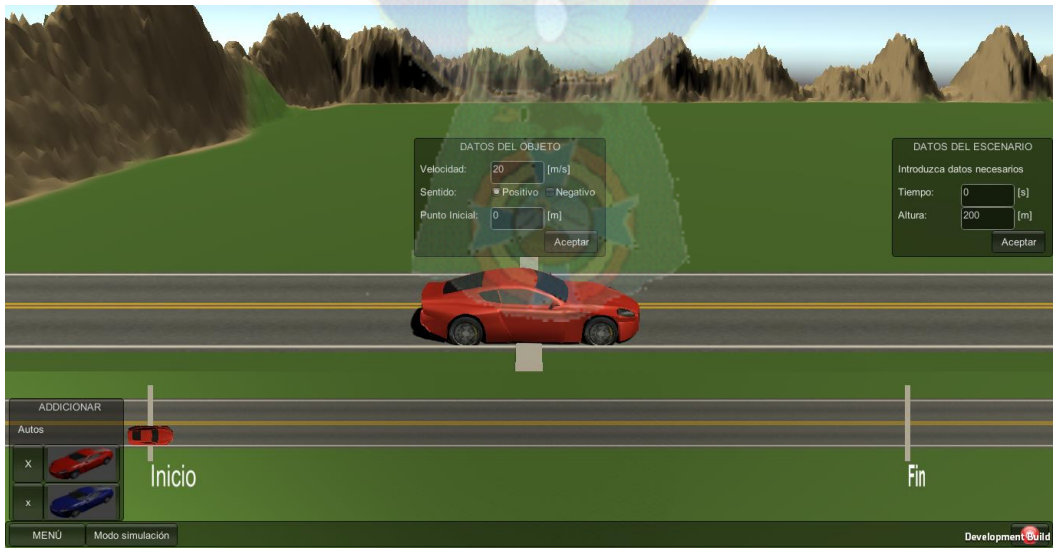



**Figura 3.39:** Introducción de datos necesarios para la simulación.

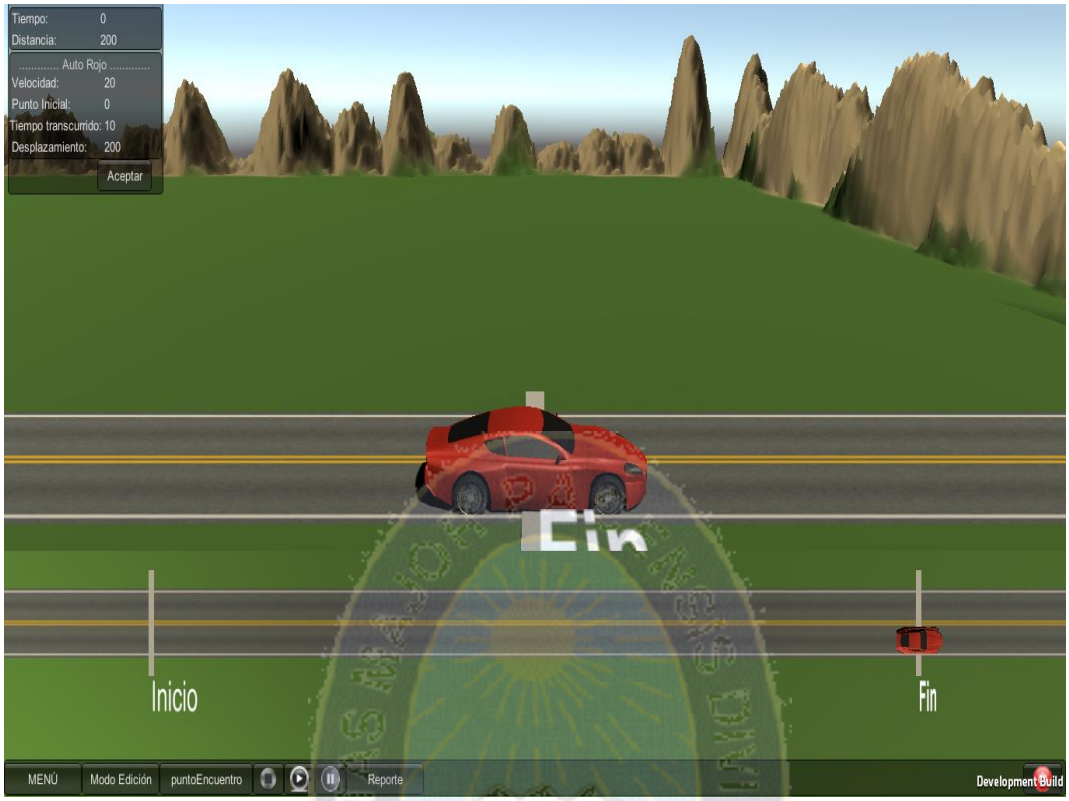



**Figura 3.40:** Resultado tras haber culminado la simulación.

### 3.8.3. PRUEBA DE LA CAJA BLANDA

Unity reconoce tres tipos de lenguaje de programación para el desarrollo de aplicaciones

- Java Script
- CSharp
- Boo

Para el desarrollo de la aplicación se optó por utilizar código CSharp. Por tener conocimientos de como desarrollo aplicaciones con tal lenguaje de programación.

La prueba de la caja blanca se realizó a los script definidos ya que se tiene que realizar pruebas al código del software, para la interacción entre las interfaces y el reconocimiento de los modelos 3D.

Código CSharp la cual lo se muestra en la Figura 3.41, un trozo de la función que permite realizar todo el proceso de simulación, la cual trabaja en función al tiempo, haciendo una actualización cada 0,01 segundo.

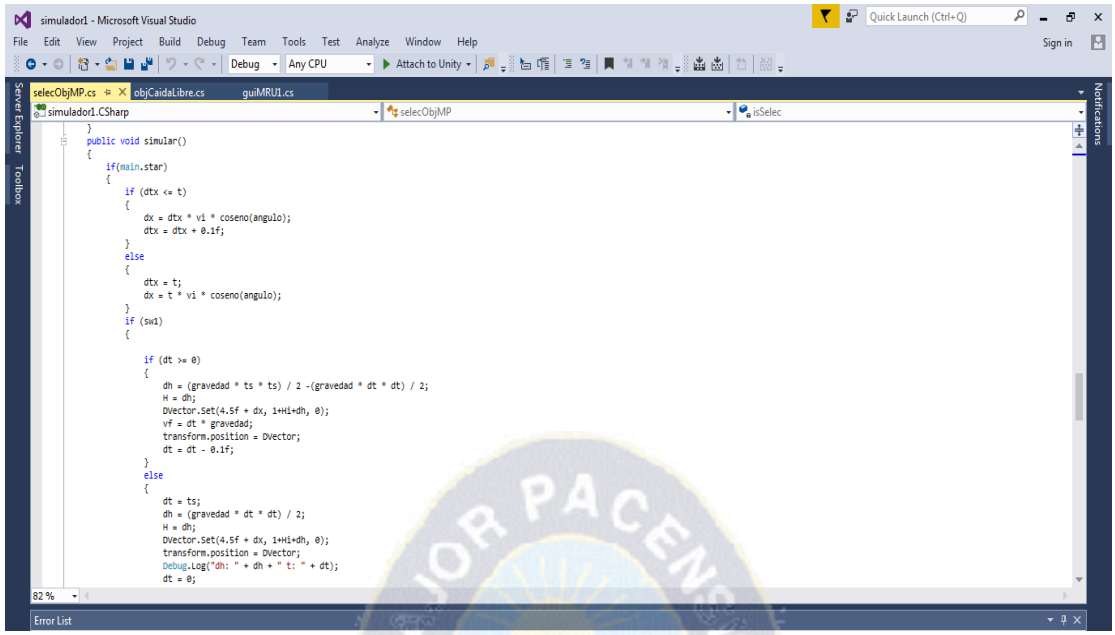



**Figura 3.41:** Código del Script ObjMP, que muestra parte del código de simulación.

Se utiliza la estructura "if "para la redirección de una escena a otra

Cada script se crea en el directorio script, los modelos 3D están en el directorio modelo auto, las imágenes usadas en el directorio imagen, las escenas creadas en el directorio escena, y los tipos de cielos usados están el directorio skylbox, lo cual se lo puede observar en la figura 3.42

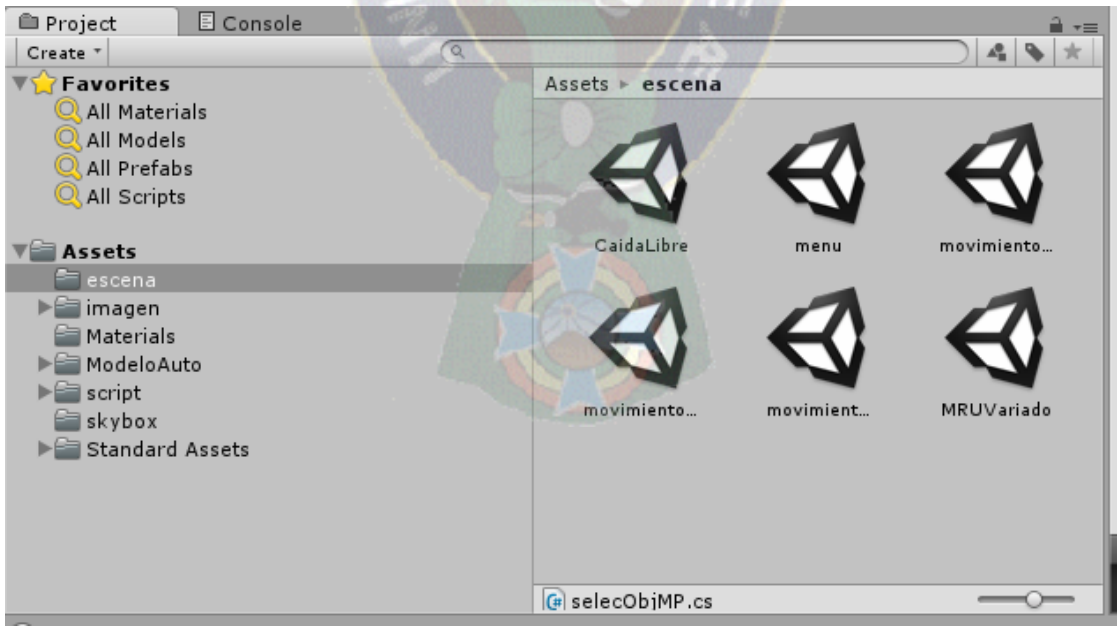



**Figura 3.42:** Organización de los archivos del proyecto.

# CAPITULO IV
# CALIDAD

## 4.1. INTRODUCCIÓN

En el presente capítulo se procede al cálculo de calidad de la aplicación educativa, tomando en cuenta el conjunto de problemáticas ya establecidas. Se propone métricas orientadas a evaluar la calidad de productos de software educativo con la utilización del estándar o norma ISO 9126.

Donde se adoptó el conjunto de características básicas de calidad del software y se agregaron aspectos relevantes a ser evaluados, entre estos están:

- Pedagógicos. Orientados a evaluar las características referentes a la enseñanza - aprendizaje.
- Contenido. Que profundizaran en los aspectos relacionados con la información que se presentara atreves de la aplicación.
- Interfaz Usuario – aplicación. Que contempla los puntos a evaluar en cuanto a la presentación de la tema educativa.

Junto a estos aspectos, se consideran los factores que sugiere la norma ISO-9126, de los cuales se tomaron aquellas que se apegan al software educativo. Los cuáles serán: funcionalidad, usabilidad, confiabilidad, facilidad de uso, eficiencia, mantenibilidad y transportabilidad de la aplicación. El objetivo es alcanzar el nivel de calidad necesario y suficiente para la evaluación del proyecto y cumplir satisfactoriamente las necesidades del usuario. Pressman, (2002) define la satisfacción como:

**Satisfacción de usuario** = producto satisfactorio + buena calidad + entrega del producto dentro de lo propuesto y tiempo establecido.

Una vez establecidos los factores a evaluar, se genera la tabla 4.1 para identificar un orden de acuerdo al grado de influencia en el producto de software educativo.

**Tabla 4.1:** Relevancia de los factores de calidad y los aspectos del software educativo. Tomado de Abud (2009)

| **Factores de calidad según estándar ISO 9126** | |
|---|---|
| **Mayor importancia** | 1. Usabilidad<br>2. Funcionalidad<br>3. Eficiencia |
| **Menor importancia** | 4. Confiabilidad<br>5. Mantenibilidad<br>6. Transportabilidad |
| **Aspectos del software educativo** | |
| **Mayor importancia**<br>**Menor importancia** | 1. Pedagógico<br>2. Interfaz U – App<br>3. contenido |

## 4.2. COMPARACIÓN CON OTROS SIMULADORES

Para poder tener en cuenta la calidad que brinda en cuanto a la funcionalidad el simulador de cinemática 3D realizar en el presente trabajo. Se procederá a realizar una comparación con dos distintas aplicaciones que permiten realizar simulaciones del área de física del tema de cinemática, las cuales son las siguientes:

**eduMedia** es una sitio Web *"www.edumedia-sciences.com"*que brinda recursos innovadores e interactivos para el aprendizaje de las ciencias la cual pretende estimular la curiosidad y el placer del aprendizaje. La cual está dirigido a profesores, alumnos o para todo público.

En este sitio se puede encontrar dos apartados, una va dirigido al nivel primario y la otra al de secundaria, en el último se puede encontrar el área de física en la cual se proporciona las opciones para realizar simulaciones de cinemática las cuales se puede observar desde la figura 4.1 hasta la figura 4.3.

En las opciones que brinda eduMedia para realizar simulaciones del área de física del tema de cinemática se puede observar que esta no brindan la opción de personalizar. Ya que no brinda la opción de adicionar los datos con la cual el usuario pretenderá realizar la simulación. El simulador es estático ya que funciona con valores predeterminados.

eduMedia no brinda todas las opciones que abarca el tema de cinemática del nivel secundario. Solo proporciona opciones para realizar simulaciones de: Movimiento rectilíneo

uniforme, Movimiento rectilíneo uniformemente variado, caída libre y movimiento parabólico. Y no así el movimiento circular.

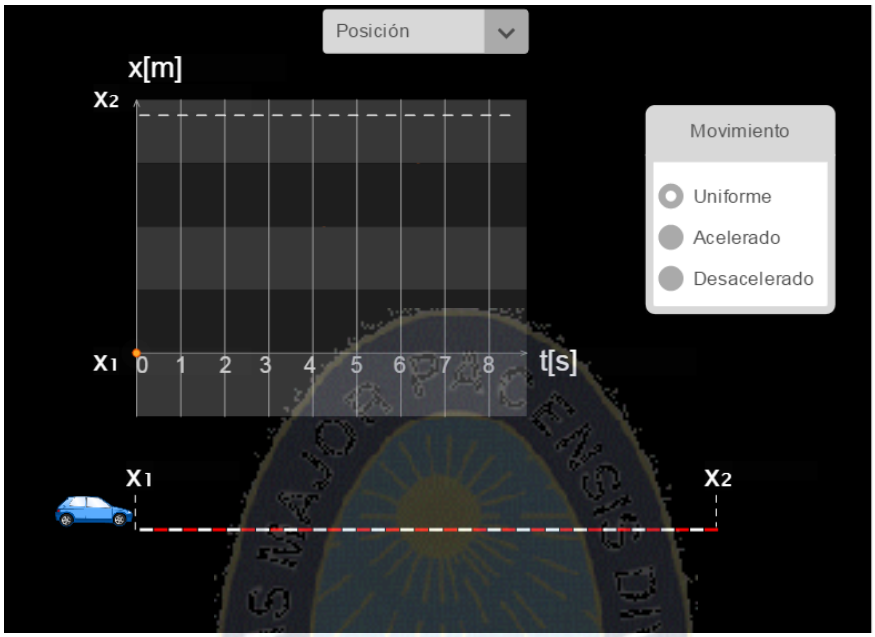



**Figura 4.1:** Entorno de trabajo para realizar simulación del MRU y MRUV. Tomado de eduMedia (2016)

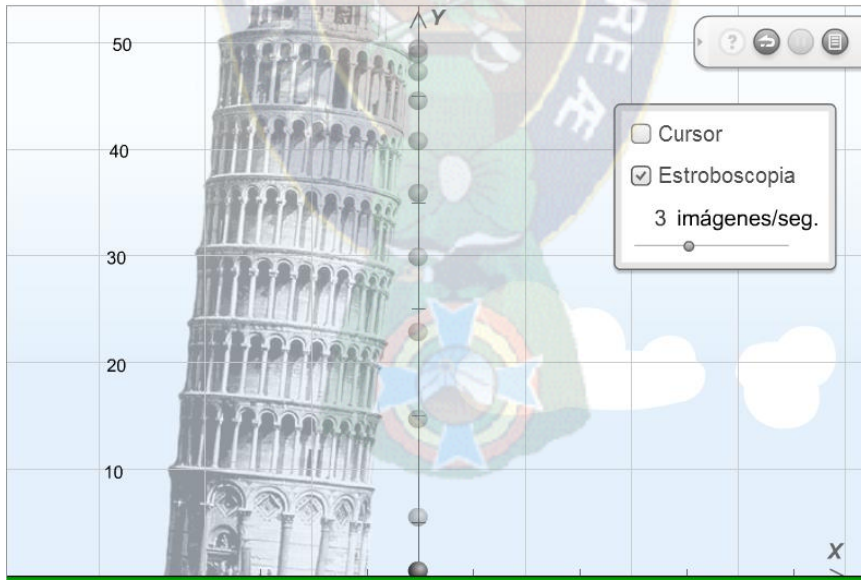



**Figura 4.2:** Entorno de trabajo para las simulaciones de Caída Libre. Tomado de eduMedia (2016)

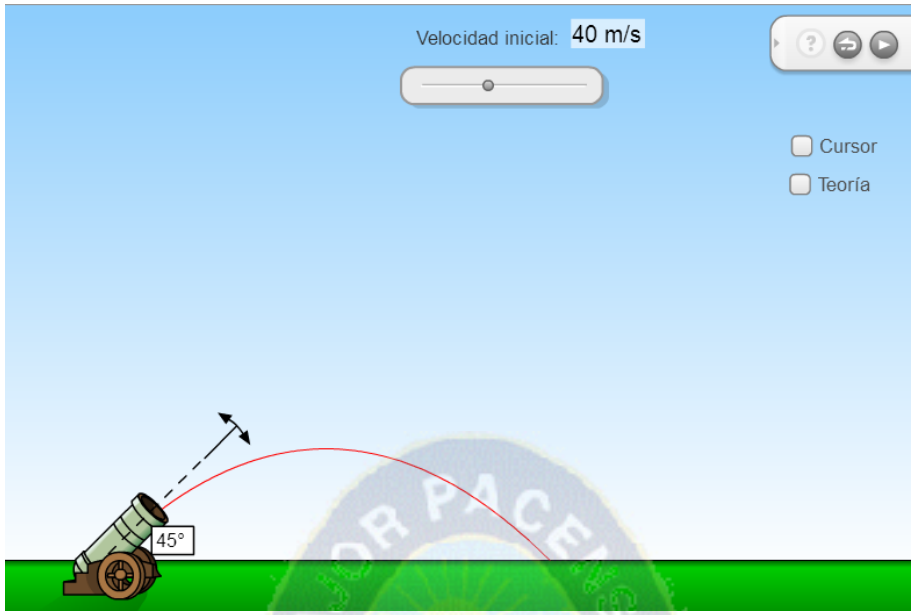



**Figura 4.3:** Entorno de trabajo para realizar simulaciones del Movimiento Parabólico. Tomado de eduMedia (2016)

**Educarex** brinda contenidos educativos digitales, la cual puede ser accedida desde su sitio Web *"http://www.educarex.es/"*. En la opción Contenidos Educativos digitales, la cual proporciona un laboratorio virtual en la cual se puede realizar simulaciones del tema de cinemática que se puede observar en la figura 4.4.

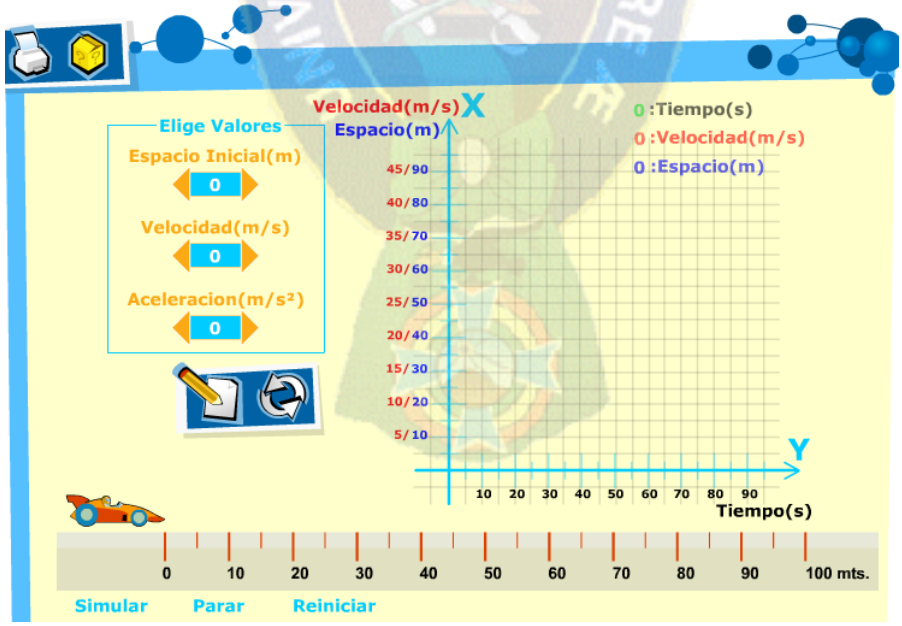



**Figura 4.4:** Entorno que permite realizar simulaciones del MRUV. Tomado de Educarex (2016)

Educarex solo brinda una sola opción para realizar simulaciones la cual es del movimiento rectilíneo uniformemente variado, en la cual se permite elegir valores del espacio inicial, velocidad y aceleración y brinda una distancia estática de 100 metros en la cual se realiza la aceleración. Los parámetros que permite variar hasta cierto punto son limitados con una velocidad máxima de 10 [m/s], espacio inicial de 10 y una aceleración máxima de 5[$m/s^2$].

El simulador de Cinemática 3D realizado en el presente trabajo proporciona opciones para realizar simulaciones de: movimiento rectilíneo uniforme, movimiento rectilíneo uniformemente variado, caída libre, movimiento parabólico y movimiento circular. Claramente se nota que el simulador 3D presenta todo el contenido temático del tema de cinemática a nivel colegiatura. Proporciona personalizar las simulaciones con datos proporcionados por el usuario.

Haciendo una comparación con el simulador que brinda eduMedia la cual carece de dar la opción de personalizar las simulaciones. También le falta la opción del movimiento circular. Referente al simulador que proporciona Educarex, esta solo da una sola opción la cual es la del Movimiento rectilíneo uniformemente variado la cual solo considera el movimiento acelerado y no así el movimiento desacelerado, brinda hasta cierto punto la opción de personalizar la simulación con ciertas limitaciones, en la tabla 4.2 en las cuales se ve el cumplimientos de las especificaciones solicitados por el profesor de la materia.

**Tabla 4.2:** Comparación de funcionalidad de los simuladores.

| | Simulador 3D Cinemática | eduMedia | educarex |
|---|---|---|---|
| **MRU** | Si | Si | No |
| **MRUV** | Si | Si | Si |
| **Caída Libre** | Si | Si | No |
| **Movimiento Parabólico** | Si | Si | No |
| **Movimiento Circular** | Si | No | No |
| **Opción de personalizar la simulación** | Si | No | No |

Por lo tanto se concluye que el simulador realizado en este trabajo supero considerablemente a las otras dos en cuando a la funcionalidad. Por lo tanto se con concluye que la aplicaciones es de muy buen calidad.

## 4.3. FACTORES DE CALIDAD SEGÚN ESTÁNDAR ISO - 9126

ISO 9126 es un estándar internacional para la avaluación del software, originalmente desarrollado en 1991 para proporcionar un esquema para la evaluación de calidad. La normativa define seis características que son:

### 4.3.1. USABILIDAD (FACTIBILIDAD DE USO)

La usabilidad es el fuerzo requerido por el usuario para utilizar el producto satisfactoriamente. Viene reflejada en la facilidad de comprensión, facilidad de aprendizaje y facilidad de operabilidad. Para comprobar la usabilidad se considera el método propuesto por Jacob Nielsen, para quien, la usabilidad es una medida abstracta.

### 4.3.2. TEST DE USUARIO FINAL

Consiste en una evaluación escrita después de las pruebas finales con un valor minino de 0 y un valor máximo de 100, para ver el grado de uso de los usuarios respecto a la aplicación, se utilizó preguntas mostradas en la tabla 4.3.

**Tabla 4.3:** Valor de ajuste según (Pressman, 20022).

| Nro. | Factor de ajuste (respecto al software) | Valor obtenido % |
|---|---|---|
| **1** | Es entendible | 90 |
| **2** | Es operable | 90 |
| **3** | Esta acorde al contenido pedagógico del tema de cinemática | 90 |
| **4** | Ayuda a promover e incentivar el aprendizaje de los estudiantes. | 100 |
| **5** | Tiene buena presentación(atractivo a la vista) | 100 |
| **6** | Puede ser usado fácilmente | 95 |
| | **PROMEDIO** | $\bar{x} = 94.17\%$ |

Por lo tanto, la software obtuvo un valor de usabilidad de 94.17 %, se interpreta como la facilidad de uso de la aplicación mediante las interfaces, es decir, que de 100 usuarios 94 pudieron entender y operar la aplicación sin ninguna dificultad.

### 4.3.3. FUNCIONABILIDAD

Es la capacidad del software de proveer los servicios necesarios para cumplir con los requisitos funcionales. Es decir, que estén disponibles a la funcionalidad y su conformidad al comportamiento deseado por el usuario.

Se obtiene mediante "Punto función" que se realiza en la notación de medida cuantitativas del dominio de información del software.

Tomando en cuenta la tabla 4.4, con la ponderación de las variables que serán evaluadas.

**Tabla 4.4:** Valor de complejidad.

| Categoría | Nivel de ajuste |
|---|---|
| **Sin influencia** | 0 |
| **Menor importancia** | 1 |
| **Moderado** | 2 |
| **Medio** | 3 |
| **Significativo** | 4 |
| **Importante** | 5 |

Se asignan valores a las preguntas de complejidad según Pressman lo cual se observa en la tabla 4.5.

**Tabla 4.5:** Valores de complejidad, según Pressman.

| Nro. | Factor de ajuste(respecto a al software) | Valor |
|---|---|---|
| **1** | ¿Requiere copias de seguridad? | 3 |
| **2** | ¿Requiere comunicación de datos? | 5 |
| **3** | ¿Es crítico el rendimiento? | 3 |
| **4** | ¿Existe funciones de procesamiento distribuido? | 3 |
| **5** | ¿Requiere entrada de datos? | 5 |
| **6** | ¿Requiere la entrada de datos que las transacciones se realicen sobre múltiples interfaces u operaciones? | 5 |
| **7** | ¿Se ejecuta el software en un entorno operativo existente y utilizado? | 5 |
| **8** | ¿Se ejecuta archivos maestros de forma interactiva? | 3 |
| **9** | ¿Son complejas las entradas, salidas, los archivos o las peticiones? | 2 |
| **10** | ¿Es complejo el procesamiento interno? | 4 |

| | | |
|---|---|---|
| **11** | ¿Se ha diseñado el código para ser reutilizable? | 5 |
| **12** | ¿Están incluidas en el diseño la conversación y la instalación? | 5 |
| **13** | ¿Se ha diseñado para soportar múltiples instalaciones en diferentes organizaciones (Unidades educativas) | 5 |
| **14** | ¿Se ha diseñado para ser fácilmente utilizable por el usuario? | 5 |
| **Valor Total** | | $\sum x_i = 58$ |

a) Número de Entradas de usuario, referido a cada entrada que proporciona datos a la aplicación, reflejando el grado de complejidad de las entras se muestra en la Tabla 4.6.

**Tabla 4.6:** Total entrada de Usuario.

| Nro. | Entrada de Usuario | Complejidad | | |
|---|---|---|---|---|
| | | Baja | Media | Alta |
| **1** | Ingreso a la aplicación | 1 | 0 | 0 |
| **2** | Ingreso a las interfaces de usuario | 0 | 0 | 7 |
| **3** | Ingreso al seleccionar una opción para ver el contenido del tema “Cinemática” | 0 | 5 | 0 |
| | **Total** | 1 | 5 | 7 |

b) Número de salidas de usuario, referido a cada salida que proporciona la aplicación al usuario con el grado de complejidad se la ilustra en la Tabla 4.7.

**Tabla 4.7:** Total salida de Usuario.

| Nro. | Salida de Usuario | Complejidad | | |
|---|---|---|---|---|
| | | Baja | Media | Alta |
| **1** | Reportes de resultado de la simulación | 0 | 5 | 0 |
| **2** | Generación de la simulación | 0 | 3 | 2 |
| **Total** | | 0 | 8 | 2 |

c) Número de Peticiones de usuario, se define como entrada interactiva que produce la generación de alguna respuesta, la cual podemos observarla en la Tabla 4.8

**Tabla 4.8:** Total Peticiones de Usuario.

| Nro. | Petición de Usuario | Complejidad | | |
|---|---|---|---|---|
| | | Baja | Media | Alta |
| **1** | Interfaces (Pantallas) | 0 | 0 | 7 |
| **2** | Interacción con Modelos 3D | 2 | 0 | 2 |
| **Total** | | 2 | 0 | 9 |

d) Número de Archivos, se considera a los archivos maestros, pueden ser: grupo lógico de datos o archivos independientes, lo cual se muestra en la Tabla 4.9 tanto la complejidad y el tipo de archivo usado en el desarrollo de la aplicación.

**Tabla 4.9:** Total Archivos Maestros.

| Nro. | Archivos | Complejidad | | |
|---|---|---|---|---|
| | | Baja | Media | Alta |
| **1** | Modelos 3D | 2 | 0 | 2 |
| **2** | Animación incorporada al modelos | 0 | 1 | 5 |
| **3** | Modelos del cielo (skybox) | 0 | 3 | 0 |
| **Total** | | 2 | 4 | 7 |

e) Numero de Interfaces externas, prácticamente son las interfaces de hardware o software para trasferir información a otra aplicación, la cual podemos observarlo en la tabla 4.10.

**Tabla 4.10:** Total Interfaces externas.

| Nro. | Interfaces externas | Complejidad | | |
|---|---|---|---|---|
| | | Baja | Media | Alta |
| **1** | Memoria de almacenamiento interno | 1 | 1 | 2 |
| **Total** | | 1 | 1 | 2 |

Teniendo todos los parámetros de medición. Se calculó el punto función sin ajusta obteniendo un valor de 370, lo cual podemos observarlo en la Tabla 4.11.

**Tabla 4.11:** Punto función sin ajustar.

| Referencia | Parámetro de medición | Factor de Ponderación | | | Valor Obtenido |
|---|---|---|---|---|---|
| | | Bajo | Medio | Alto | |
| **a)** | Entradas de Usuario | 1*3=3 | 5*4=20 | 7*6=42 | 65 |
| **b)** | Salidas de Usuario | 0*4=0 | 8*5=40 | 2*7=14 | 54 |
| **c)** | Peticiones | 2*3=6 | 0*4=0 | 9*6=54 | 60 |
| **d)** | Archivos | 2*7=14 | 4*10=40 | 7*15=105 | 159 |
| **e)** | Interfaces | 1*5=5 | 1*7=7 | 2*10=20 | 32 |
| **Total Punto función sin ajustar** | | | | | 370 |

Calculo Punto Función (PF).

$PF$ = PFSA($grado\ de\ confiabilidad + Tasa\ de\ error * \sum xi$)

Donde:

| | |
|---|---|
| PFSA | Total de puntos función sin ajustar |
| Grado de Confiabilidad | Valor de 0.65, (pie de corrección) |
| Tasa de Error | Valor 0.01, (Error de confiabilidad de la aplicación) |
| $\sum(\boldsymbol{xi})$ | Valor Total de la complejidad de la aplicación la cual se la obtiene de la Tabla 4.4 |

Reemplazando valores:

$PF$ = 370 (0.65 + 0.01 $*$ 58)

$PF$ = 455.1

Obtenemos el punto función ideal, calculado con el valor máximo de $\sum Xi$

$PFideal$ = 370 (0.65 + 0.01 $*$ 70)

$PFideal$ = 499.5

Por ultimo obtenemos la funcionalidad deseada:

$$Funcionalidad = \frac{PF}{PFideal} = \frac{455.1}{499.5} = 0.9111$$

Por lo tanto, se obtuvo una funcionalidad de 91.11%, tomando en cuenta el punto función máximo. Lo que quiere decir que la aplicación móvil cumple satisfactoriamente con los requisitos funcionales.

### 4.3.4. EFICIENCIA

Es la relación entre las prestaciones del software y los requisitos necesarios para su utilización. Indicada por los tiempos de uso y recursos utilizados.

Se definieron cuatro preguntas mostradas en la Tabla 4.12, para evaluar la eficiencia de la aplicación y se pondero con valores de 0 – 100.

**Tabla 4.12:** Factores de Eficiencia.

| Nro. | Factores a Evaluar | Cantidad |
|---|---|---|
| **1** | ¿El tiempo promedio de respuestas a las consultas es adecuado? | 90 |
| **2** | Tiene rendimiento de acuerdo a los factores que utiliza | 95 |
| **3** | Brinda respuestas adecuadas a las consultas dadas | 98 |
| **Promedio** | | **94.33** |

Se brinda una eficiencia de 94.33 % con la aplicación.

### 4.3.5. CONFIABILIDAD (FIABILIDAD)

Pressman (2002) define como: "Probabilidad de operación libre de fallos de un programa de computadora en un entorno determinado y en un tiempo específico".

Una medida de confiabilidad es el tiempo medio entre fallos:

$$\boldsymbol{TMEF} = TMDF + TMDR$$

Donde:

TMEF – Tiempo medio entre fallos

TMDF – Tiempo medio de fallo

TMDR – Tiempo medio de reparación

Reemplazando valores:

$\boldsymbol{TMEF}$ = 6 $hrs\ de\ trabajo$ + 0.5 $hrs\ de\ reparacion$ = 6.5 $horas$

Medida de disponibilidad de la aplicación, que será la probabilidad de que la aplicación funcione de acuerdo a los requisitos dados en un momento determinado.

$$\text{Disponibilidad } = \frac{\text{TMDF}}{\text{TMDF } + \text{ TMDR}} = \frac{6\text{ hrs}}{6\text{hrs } + \text{ 0.5 hrs}} = 0.9230$$

Finalmente se obtiene una confiabilidad de 92.30 %.

### 4.3.6. MANTENIBILIDAD

Es el esfuerzo necesario para adaptarse a las nuevas especificaciones y requisitos del software. No existe forma de medir directamente la facilidad de mantenimiento, Pressman sugiere medidas indirectas para medir el grado de Mantenibilidad.

Para que se pueda realizar una modificación sin alterar la funcionabilidad de la aplicación se tiene q tomar los siguientes puntos: Facilidad de Análisis, Facilidad de Cambio, Estabilidad y Facilidad de prueba. Además de que el desarrollador debe realizarse una seria de preguntas descritas en la tabla 4.13.

Se tiene un 93.4% de Mantenibilidad de software, lo que quiere decir que un desarrollador puede realizar cambios en el código y estructura interna y externa de la aplicación desarrollada.

**Tabla 4.13:** Factores de Mantenibilidad.

| NRO. | FACTORES DE AJUSTES | VALOR |
|---|---|---|
| **1** | Se pueden identificar las partes que deben ser modificadas. | 90 |
| **2** | Existe facilidad de realizar cambios | 95 |
| **3** | Es fácil analizar un fallo de error | 90 |
| **4** | Los cambios mejoran la facilidad de pruebas. | 95 |
| **5** | Los cambios permiten una mejor estabilidad | 97 |
| **Promedio** | | 93.4 |

### 4.3.7. TRANSPORTABILIDAD

Definida como la capacidad del software ser transferido de un entorno a otro. Toma en cuenta los siguientes puntos: Adaptabilidad, Instalabilidad, Coexistencia y capacidad para ser reemplazable. Además de los requerimientos de Software y Hardware para ser transportado la cual se figura en la Tabla 4.14.

**Tabla 4.14:** Factores de Transportabilidad.

| Nro. | Factor de ajuste | Valor |
|---|---|---|
| **1** | Fácil de identificar el instalador de la aplicación | 95 |
| **2** | Puede ser transferido de un ordenador a otro | 95 |
| **3** | Necesita requerimiento de SW y HW | 85 |
| **4** | Se brinda ayuda para la instalación | 90 |
| **Promedio** | | 91.2 |

Así la aplicación tiene 91.2% de poder ser transferido e instalado en más de un ordenador a otro, sin tener dificultades de adaptabilidad o reinstalación.

### RESULTADO DE LOS ASPECTOS DE CALIDAD

El aspecto de calidad está relacionado directamente con el grado de satisfacción del usuario. Lo cual la tabla 4.15 muestra el resultado de calidad de la aplicación.

**Tabla 4.15:** Resultado del aspecto de calidad.

| CARACTERÍSTICAS ISO-9126 | RESULTADO |
|---|---|
| **Usabilidad** | 94.17% |
| **Funcionabilidad** | 91.11% |
| **Eficiencia** | 94.33% |
| **Confiabilidad** | 92.30% |
| **Mantenibilidad** | 93.40% |
| **Transportabilidad** | 91.20% |
| **PROMEDIO** | 92.75% |

## 4.4. ASPECTOS DEL SOFTWARE EDUCATIVO

Abud (2005) propone: Una escala para criterios binarios y multinivel en la que se asigna 40% para el aspecto pedagógico, 36% para la interfaz humano-computadora, 12% para el

contenido y 12% para el aspecto técnico la cual se la observa en la Tabla 4.16. Por último se establece una escala de aceptabilidad que van del 0 a 3, basada también en los rangos de satisfacción que establece el estándar ISO 9126 que se la ilustra en la tabla 4.17.

**Tabla 4.16:** Niveles de calidad para software educativo.
Tomado de Abud (2005)

| Nivel de calidad | Puntaje | Equivalencia |
|---|---|---|
| **0** | $u_k = 0$ | Ausencia de calidad |
| **1** | $u_k = 70$ | Calidad regular |
| **2** | $u_k = 90$ | Calidad aceptable |
| **3** | $u_k = 100$ | Calidad excelente |

**Tabla 4.17:** Nivel de Aceptabilidad de los valores de preferencia.
Tomado de Abud (2005)

| Nivel de calidad | Categoría | Puntaje |
|---|---|---|
| **0** | Mal | 0-50 |
| **1** | Regular | 50-70 |
| **2** | Bien | 70-90 |
| **3** | Excelente | 90-100 |

Para el cálculo de la calidad global, se utilizó el modelo de atributos múltiples:

$$U = \left[\sum_{k=1}^{n} w_k u_x\right]/100 \qquad (*)$$

Donde:

**U** es la calidad global.

$\boldsymbol{u_k}$, es el peso para el factor de calidad.

$\boldsymbol{w_k}$, es el puntaje obtenido para la alternativa k

Una vez establecidos los aspectos relevantes para la evaluación del software educativo, se estableció una escala de aceptación basada en los rangos de satisfacción del estándar ISO 9126 la cual la observamos en la Tabla 4.18.

**Tabla 4.18:** Métricas para calidad del software educativo en el aspecto pedagógico.

| Aspecto | Factor | Atributo | Peso y criterio |
|---|---|---|---|
| **Pedagógico 40%** | Facilidad de uso | Facilidad de aprendizaje | 4 binario |
| | | Diversidad en las actividades propuestas. | 4 binario |
| | | Actividades adecuadas para reforzar el aprendizaje. | 4 multinivel |
| | | Actividades motivadoras para el alumno. | 4 binario |
| | | Existe relación con lo que el profesor enseña. | 4 binario |
| | Funcionalidad | Seguridad del alcance de los objetivos educativos. | 4 binario |
| | | Manejo automático del historial académico | 4 binario |

Tomando en cuenta $wk$= 90, y el resultado total de los aspectos de calidad reemplazamos en la ecuación (*).

$$U = \frac{90 * 92.75}{100}$$

$$U = 83.475$$

Finalmente podemos decir que la Calidad Global del software educativo se encuentra en el rango.

$$\mathbf{90 < U = 83.475 < 100}$$

Por lo que se encuentra en la categoría "Excelente", siendo una categoría óptima para la calidad de la aplicación educativa "Simulador de Cinemática"

# CAPÍTULO V

# ANÁLISIS DE COSTO Y BENEFICIO

## 5.1. ESTIMACIÓN DE COSTO

La estimación de costo para el desarrollo de software es un factor muy importante en el análisis de los proyectos, constituye un tema estratégico contar con métricas para medir el costo de los mismos, garantizando la eficiencia, excelencia, calidad y la competitividad. El análisis de costo es el proceso de identificación de los recursos necesarios para llevar a cabo el trabajo o proyecto eficientemente. En la actualidad existen un conjunto de métricas que no se utilizan, y que pueden ser aplicables a cualquier tipo de proyecto de software para calcular el costo de los mismos.

### 5.1.1. FACTORES QUE INFLUYEN EN EL COSTO DE SOFTWARE

Se hace una estimación durante el estudio preliminar del problema y se revisa durante el análisis de factibilidad del proyecto. Una estimación mejorada se presenta durante las especificaciones del software y la estimación final durante la revisión del diseño. Los factores que principalmente afectan el costo del software son:

- Capacidad del Programador
- Complejidad del Producto
- Tamaño del Programa
- Tiempo Disponible
- Confiabilidad Requerida

### 5.1.2. ANÁLISIS DE COSTO BENEFICIO

Pressman (2002) sugiere el uso de: COCOMO II "Constructive Cost Model", que surge como una alternativa para incluir componentes de incerteza en las estimaciones conforme al nivel de información disponible.

Este es un modelo paramétrico que establece ecuaciones matemáticas para describir las relaciones entre el tamaño del software.

- Factor primario de costo representado en términos de puntos de función.

- Factores secundarios, buscan capturar particularidades de producto, proceso, persona y plataforma. Denominados Cost Drivers.

El nuevo modelo incorporado en el año 1990, tiene características de los modelos COCOMO 81 y Ada COCOMO. EL modelo COCOMO II, post-arquitectura cubre el actual desarrollo y mantenimiento de un producto de software. En esta etapa de ciclo de vida procede más a un costo efectivo.

Está compuesto por tres modelos denominados

- Composición de Aplicación
- Diseño Temprano
- Post – Arquitectura

En el cálculo de coste de la aplicación se ara unos del modelo post-arquitectura.

#### 5.1.2.1. MODELO POST-ARQUITECTURA

El Modelo Post-Arquitectura incluye el actual desarrollo y mantenimiento de un producto software. Esta fase avanza rentablemente si se desarrolla una arquitectura de ciclo de vida software válida con respecto a la misión del sistema, al concepto de operación y al riesgo, y establecido como marca de trabajo del producto. El modelo correspondiente de COCOMO II tiene aproximadamente la misma granularidad que los anteriores modelos COCOMO y AdaCOCOMO. Utiliza instrucciones fuente y/ó Puntos de Función para medir, con modificadores para reutilización y objetos software; un conjunto de 17 drivers de coste multiplicativos; y un conjunto de 5 factores que determinan el exponente de escala del proyecto. Estos factores sustituyen los modos de desarrollo (Orgánico, Semilibre y Rígido) del modelo original COCOMO y refina los 4 factores de exponente-escala en Ada COCOMO.

La fórmula básica para obtener una estimación de esfuerzo es:

$$MM_{NOMINAL} = A * (\text{KSLOC})^{B}$$

- $\boldsymbol{MM_{NOMINAL}}$, es el esfuerzo nominal para un proyecto de un tamaño dado expresado en Meses-persona.
- **A**, es una constante que captura los efectos multiplicadores de esfuerzo en proyectos de tamaño incremental. Provisionalmente se la ha estimado un valor de 2.45.

- $KSLOC$, es el tamaño del software expresado en líneas de código fuente.
- **B,** explica el ahorro ó gasto relativo de escala encontrado en proyectos software de distintos tamaños.

### 5.1.2.2. TAMAÑO DE LA APLICACIÓN EN MILES DE LÍNEAS DE CÓDIGO

El tamaño de una aplicación se mide en unidades de líneas de código fuente (KSLOC). Al igual que en la versión inicial del COCOMO, este valor se deriva de la medida de módulos software que constituirán el programa de aplicación, sin embargo, en la nueva versión COCOMO II puede estimarse también a partir de Puntos de Función sin ajustar convirtiendo a SLOC y luego dividiendo por 1000.

**Tabla 5.1:** Conversión de puntos de Función a Líneas de código.
**Fuente:** Tomado de ingeniería S. (2009)

| Lenguaje | SLOC/UFP |
|---|---|
| **Ada** | 71 |
| **Al Shell** | 49 |
| **APL** | 32 |
| **Assembly** | 320 |
| **Assembly (Macro)** | 213 |
| **ANSI/Quick/Turbo Basic** | 64 |
| **Basic – Compiled** | 91 |
| **Basic Interpreted** | 128 |
| **C** | 128 |
| **C++** | 29 |
| **Visual Basic** | 32 |
| **ANSI Cobol 85** | 91 |
| **Fortran 77** | 105 |
| **Forth**<br>**Jovial** | 64<br>105 |
| **List** | 64 |
| **Modula 2** | 80 |
| **Pascal** | 91 |
| **Prolog** | 64 |
| **Peport Generator** | 80 |
| **Spreadshet** | 6 |

Utilizando Punto Función sin Ajustar para determinar el tamaño del proyecto, éstos deben convertirse en líneas de código fuente en el lenguaje de implementación "ensamblador, lenguajes de alto nivel, lenguajes de cuarta generación, etc...", para evaluar la relativamente concisa implementación por Puntos de Función ver Tabla 5.1. COCOMO II realiza usando tablas que traducen Puntos de Función sin ajustar al equivalente SLOC.

Por lo tanto calculando con el valor obtenido del punto de función sin ajustar PFSA=370 ver Tabla 4.8 tenemos que:

$$SLOC = 370 * 29 \ = 10730 \ (Lineas\ de\ codigo\ fuente)$$

$$KSLOC = 10730 \ /1000 \ = 10.73 \ (Miles\ de\ Lineas\ de\ codigo\ fuente)$$

### 5.1.2.3. VARIABLE B

**Tabla 5.2:** Factores de escalamiento SF.
Tomado de Ingeniería S. (2009)

| Factores de Escala($SF_j$) | Muy Bajo | Bajo | Nominal | Alto | Muy Alto | Extra Alto |
|---|---|---|---|---|---|---|
| PREC | Completamente sin precedentes | Prácticamente sin precedentes | Casi sin precedentes | Algo familiar | Muy familiar | Completamente familiar |
| FLEX | Riguroso | Relajación ocasional | Algo de relajación | Conformidad general | Algo de conformidad | Metas generales |
| RESL* | Poco (20%) | Algo (40%) | A menudo (60%) | General-mente (75%) | En su mayor parte (90%) | Por completo (100%) |
| TEAM | Interaccion-es muy difíciles | Algo de dificultad en las interacciones | Interacciones básicamente cooperativas | Bastante cooperativo | Altamente cooperativo | Completas interacción |
| PMAT | Peso medio de respuesta "Si" para el cuestionario de Madurez CMM | | | | | |

$$B = 0.91 + 0.01 * \sum_{j=1}^{5} SF_j$$

El exponente B se obtiene mediante los denominados drivers de escala. La selección de drivers de escala se basa en la razón de que ellos son un recurso significante de variación exponencial en un esfuerzo ó variación de la productividad del proyecto. Cada driver de escala tiene un rango de niveles de valores desde Muy Bajo hasta Extra Alto (tabla 5.2). Cada nivel de valores tiene un peso, SF, y el valor específico del peso se llama factor de escala. Un factor de escala de un proyecto, SFj (ver tabla 5.3), se calcula sumando todos los factores y se usa para determinar el exponente de escala, B.

**Tabla 5.3:** Valores de los Factores de escala.
Tomado de Ingeniería S. (2009)

| Factores de Escala($SF_j$) | Muy Bajo | Bajo | Nominal | Alto | Muy Alto | Extra Alto |
|---|---|---|---|---|---|---|
| **PREC** | 6.20 | 4.96 | 3.72 | 2.48 | 1.24 | 0.00 |
| **FLEX** | 5.07 | 4.05 | 3.04 | 2.03 | 1.01 | 0.00 |
| **RESL*** | 7.07 | 5.65 | 4.24 | 2.83 | 1.41 | 0.00 |
| **TEAM** | 5.48 | 4.38 | 3.29 | 2.19 | 1.10 | 0.00 |
| **PMAT** | 7.80 | 6.24 | 4.68 | 3.12 | 1.56 | 0.00 |

Para el cálculo de la variable B usaremos los factores de nivel muy bajos ya que son las que se ajustan en cuanto a la elaboración del proyecto.

Por lo tanto tenemos:

$$B = 0.91 + 0.01 * \sum_{j=1}^{5} SF_j$$

Remplazando los valores tenemos:

$$B = 0.91 + 0.01 * (6.20 + 5.07 + 7.07 + 5.48 + 7.80)$$

$$B = 0.91 + 0.01 * (6.20 + 5.07 + 7.07 + 5.48 + 7.80)$$

$$B = 1.2262$$

Por lo tanto tenemos nominal es:

$$MM_{NOMINAL} = A * (\text{KSLOC})^B$$

Con los datos:

$$MM_{NOMINAL} = 2.45 * 0.1(10.73)^{1.2262}$$

$$MM_{NOMINAL} = 4.496\ personas/mes$$

Duración estimada en meses:

$$D = C * (MM_{NOMINAL})^{0.002*B}\ \ donde\ C = 3.67$$

$$D = 3.67 * (4.496)^{0.002*1.2262}$$

$$D = 3.67 * 1.004$$

$$D = 3.68$$

Es decir que el proyecto tendrá una duración de 4 meses de desarrollo aproximadamente.

Para la cantidad de desarrolladores tenemos:

$$P = \frac{MM_{NOMINAL}}{D}$$

$$P = \frac{4.496}{3.68} = 1.22$$

Por lo tanto como no existe 1.22 cantidad de desarrolladores, se toma la opción de redondeo es decir,$1.22 \cong 1$ una persona será quien desarrolle la aplicación.

### 5.1.2.4. COSTO.

Teniendo en cuenta que el salario mínimo de un profesional informático es de 350 a 750 $us, expresamos este valor en moneda nacional teniendo 2450 a 5250 Bs. Podemos estimar es costo de la aplicación.

$$Costo = SalarionMinimo(\bar{x}) * Duracion * NumPersonas$$

$$Costo = 3500 * 4 * 1$$

Costo total de la aplicación:

$$\boldsymbol{Costo = 14000\ Bs}$$

### 5.1.2.5.BENEFICIO

El beneficio se tratara en términos de Aprendizaje – Estudiante. Y el costo definido anteriormente por costo de la aplicación.

Prácticamente el beneficio del uso de la aplicación en los estudiantes, brindara indirectamente un beneficio al aprendizaje de la materia de Física, en si es el cumplimiento del objetivo que se planteó en el Capítulo uno. La aceptación de la aplicación dependerá de la calidad con la que se desarrolló.

Con el resultado de la tabla 4.2 se obtuvo un 94.17% de Aceptabilidad, relacionando con el costo del proyecto se tiene:

Para costo beneficio se realizara una tasa de descuento del 12% (iva=0.12), puesto que es el interés por prestación de servicio.

$$CostoBeneficio = \frac{aceptavilidad}{CostoNeto} * 100 = \frac{94.17}{12320} * 100$$

$$CostoBeneficio = 0.7644$$

Es decir que por cada boliviano invertido en la aplicación, abra 8 de cada 10 estudiantes tendrán un mejor aprendizaje en aulas.

# CAPÍTULO VI

## CONCLUSIONES Y RECOMENDACIONES

### 6.1. CONCLUSIONES

A través del presente trabajo se efectuó la realización de la aplicación "Simulador 3D de Cinemática", la cual permite realizar simulación en los campos de:

- Movimiento rectilíneo uniforme
- Movimiento rectilíneo uniformemente variado
- Caída libre
- Movimiento parabólico
- Movimiento circular uniforme

Se logró alcanzar las metas propuestas, logrando utilizar con éxito modelos 3D para aumentar el interés de los estudiantes de nivel secundario de la Unidad Educativa Rep. De Italia

Se concluye lo siguiente:

- El uso de modelos tridimensionales mediante la aplicación desarrollada, para el aprendizaje de Cinemática del área de Física, brinda gran afectividad como herramientas didáctica para los alumnos del nivel secundario.
- Se observó una gran aceptación por parte de los estudiante y profesores del are. Incentivando más el proceso de aprendizaje y enseñanza. También haciendo conocer el buen uso que se puede dar a las computadoras.
- La utilización de modelos 3D y sus respectivas animaciones de los componentes que forman el Simulador 3D de Cinemática del área de física, evaluados con criterios pedagógicos: sea más atractiva e interactiva para el aprendizaje.
- El uso de la metodología de ingeniería de software educativo fue ideal en el desarrollo destinado a los estudiantes y profesores por las características que nos brinda al desarrollar en software y así mismo dando soluciones educativas.

A continuación se hace el cierre de los objetivos específicos.

- Se implementar criterios pedagógicos y tecnológicos en el desarrollo de la aplicación.
- Se tomó en cuenta toda la bibliografía necesaria de cinemática.
- Se capacito a los usuarios del uso del simulador de cinemática.
- Se desarrolló los modelos en tres dimensiones para:
  - movimiento rectilíneo uniforme
  - movimiento rectilíneo uniformemente variado
  - caída libre
  - movimiento parabólico
  - movimiento circular uniforme.
- Se realizó las funciones de:
  - movimiento rectilíneo uniforme.
  - movimiento rectilíneo uniformemente variado.
  - caída libre
  - movimiento parabólico.
  - movimiento circular uniforme.

Haciendo cumplir los objetivos específicos, cerramos el objetivo principal

- Se desarrolló una aplicación que permita realizar simulaciones de cinemática basado en la programación grafica 3D para poder captar la atención de los estudiantes de nivel secundario de la Unidad Educativa Rep. De Italia.

## 6.2. RECOMENDACIONES

Buscando siempre el mejoramiento de la aplicación se brinda las siguientes recomendaciones:

- Se recomienda hacer uso de computadoras con capacidad de uso DX9 o superiores para el buen funcionamiento de la aplicación.
- Se recomienda ver la interfaz de ayuda para una mejor comprensión de la aplicación donde se muestra los pasos para su mejor uso.
- Se recomienda estudiantes trabajen conjuntamente con el profesor a cargo para un mejor uso.

Recomienda a los que deseen continuar con el trabajo:

- Complementar con el módulo de Movimiento circular Uniformemente varia.
- Hacer uso de archivos para guardar y abrir escenas de simulación
- Aumentar modelos 3D para abarcar más ejemplos de simulación.

# BIBLIOGRAFÍA


Acosta, Luis Alejandro (2005). *Guía Práctica para la sistematización de proyectos y programa de cooperación regional, Oficina* regional de las FAO para América Latina, Segunda Edición

Fernández G (1993), *Teoría y análisis práctico de la integración,* 1ra edición, La paz Bolivia, Edición Santa Ana.

Real academia Española (2014). *El Diccionario de la lengua española es la obra lexicográfica de referencia de la Academia,* 23.ª edición.

Ian Sommerville (2005), *Ingeniería de Software,* 7ma edición, Madrid España, Person Educación S.A., 2005.

Paul A. Tipler (2006),*Física Preuniversitaria,* 1ra edición Barcelon-Bogota-Buenos Aires-Caracas-México, Editorial Reverte S.A, 2014.

Jara, Oscar (s/f-a). "*orientaciones teórico-práctico para la sistematización de experiencias*" centro de Estudios y Publicaciones Alforja. Disponible en: http://www.cepalforja.org/sistem/documentos/aprenderdepracticas.pdf

Antonieta Abud Figueroa (2005), MECSE, Conjunto de métricas para Evaluar software educativo.

Arteaga, (2008), *Diseño de espacios interactivos tridimensionales a travez de la internet, aplicando realidad virtual inversiva (Tesis de Grado).* La Paz – Bolivia: Universidad Mayor de San Andrés – Informática.

Center for Software Engineering USC (1995 - 2000), COCOMO II Model Definition Manual Version 2.1.

Corinne Wallshein (2009), Software Sizing Lines of Code and Beyond. U.S Air Force – Air Force Cost Analysis Agency. - EcuRed – Conocimiento con todos y para todos (2015), COCOMO II.

Fattocs. Estimación de proyectos de software con COCOMO II: La métrica inmersa en modelo de costos.

Galvis, A. (1992). *Ingeniería de software educativo. Colombia*, Santa fe de Bogotá: Ediciones Uniandes.

Gómez David A. (2009), ISO 9126 sitio web: http://alejandrogomeziso.blogspot.com

Hans. LeRoy. Esquema de Evaluación de Software Educativo. Sitio web http://home.tiscalinet.be/hanslr/evalsed.htm

Ingeniería de software I (2010), Capitulo IV. Estimación y planificación temporal del proyecto de software.

Mendoza Gonzalo M. (2006), ISO 9126-3: Métricas internas de la calidad del producto de software.

Murillo F.J y Fernandez M.J (2002), Software Educativo. Algunos criterios para su evaluación,Infodidac, 18, 8-12.

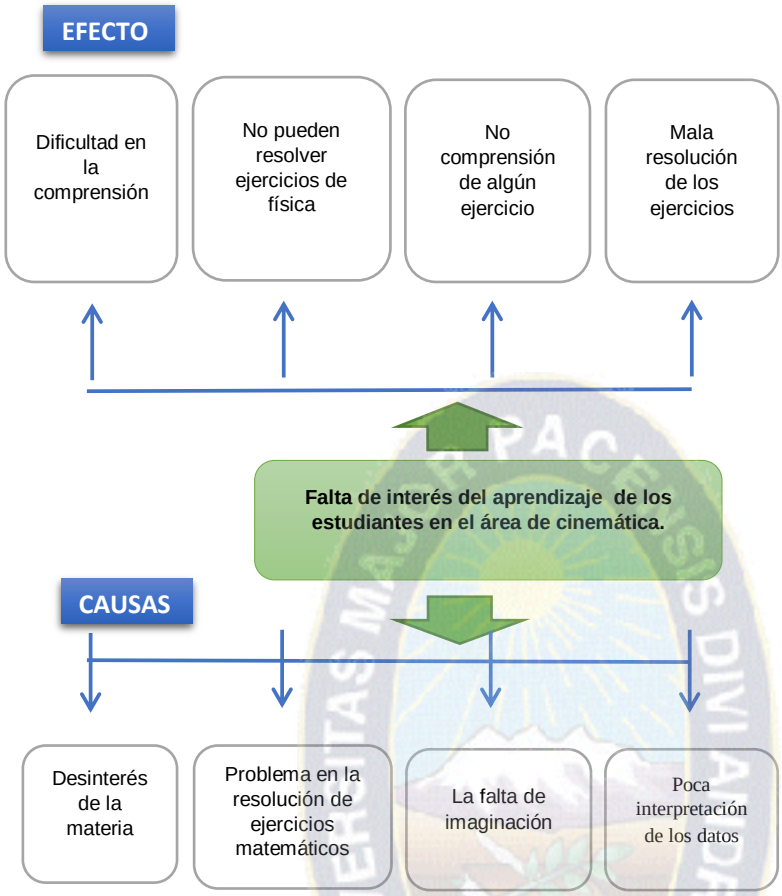
Pressman R (1993), *Ingeniería de Software. Un enfoque práctico*, 5ta edición, México, Mc Graw Hill

**PAGINAS CONSULTADAS**

- Página oficial de OpenGL (sf). Recuperado de: http://www.khronos.org/opengles
- Página oficial de Vuforia (s.f.). Recuperado de https://developer.vuforia.com
- Página oficial de Blender (s.f.). Recuperado de https://www.blender.org
- Página oficial Unity – Game Engine (s.f.). Recuperado de https://unity3d.com/es

# ANEXOS.

## ÁRBOL DE PROBLEMAS




## ÁRBOL DE OBJETIVOS