Universidad de las Ciencias Informáticas
Vertex, Interactive 3D Environments, Facultad 5

Herramienta para la selección de controles de animación de personajes 3D en Blender

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Oscar Carmenate Rodríguez

Tutor: Ing. Alexis Echemendía González

La Habana, 2016

«El arte pone a prueba la tecnología y la tecnología inspira el arte.»
-- John Lasseter

Dedicatoria

A mi mamá, que siempre ha sido el lugar de mi descanso sentimental luego de cada batalla. A mi tía Miladys, que ha luchado conmigo en cada batalla que me ha impuesto la vida. A mi papá, por hacerme ver la vida más sencilla y darme las fuerzas para afrontar cualquier tipo de situación que se presente. A Raúl M. Feria por sus oportunos consejos, y por ver en mí, un hijo.
Dentro de esta casa de altos estudios, deseo dedicar esta tesis a: Francisca Andrea Mazorra Mestre, por darme las fuerzas y el valor que me faltaron en los momentos difíciles.

Agradecimientos

Le agradezco a las personas que han brindado su apoyo a la elaboración de la presente tesis:

Ing. Raydel Cabrera Isasi:
Por su ayuda en la redacción y revisión de casi la totalidad de su contenido. Él ha sido sinceramente, el mejor amigo que he tenido en esta universidad, sus conocimientos y apoyo, han sido vitales para la culminación de la presente tesis.

Ing. Dailyn García Dominguez:
Por su apoyo emocional, además de contribuir en la redacción de la Introducción, el Capítulo I, y revisión de todo el documento. Dailyn me dio una prioridad muy alta entre todo lo que tenía que hacer como madre y profesional, dándome parte de su tiempo, sé que si hubiese podido me hubiese dado más.

A mi tutor Ing. Alexis Echemendia González:
Por hacerme crecer como profesional, y por su confianza en mi.

Msc. Ernesto de la Cruz Guevara:
Por si guía en la estructura del documento.

Ing. Sailyn Salas Hechavarría:
Por su revisión incluso cuando estaba realizando su trabajo de maestría.

Ing. Yirka Céspedes Boch
Por su influencia sobre mí para el mejoramiento de esta tesis.

Por su oportunas revisiones del documento y retroalimentación agradezco a:
Ing. Enelis Blanca Cuba Rondón

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Oscar Carmenate Rodríguez	Ing. Alexis Echemendía González
Autor	Tutor

Resumen

Las técnicas para representar entornos virtuales se encuentran en constante evolución y cada vez más se muestran en los productos audiovisuales que hoy se comercializan. Los personajes en tres dimensiones (3D) juegan un papel importante en este sentido, lográndose mayor realismo y aceptación a partir del uso de esta tecnología. Una de las técnicas más utilizadas a nivel mundial para la animación de personajes es *Keyframe Animation*, que posee entre sus elementos principales los controles de animación. La selección de estos controles es un proceso complejo cuando se realiza sobre el personaje. La *suite* de diseño 3D: *Blender* constituye una alternativa de código abierto y libre de costo que ha ido evolucionando en los últimos años ganando cada vez más adeptos, aunque aún posee limitaciones en la selección de los controles del personaje. En el presente trabajo se desarrolla una herramienta para la *suite Blender*, que permite la selección de controles de animación de personajes 3D, sin la visualización de los controles sobre el personaje. De este modo, se pretende mejorar el proceso de animación que se realiza en los proyectos del Centro de Entornos Interactivos 3D (VERTEX), logrando reducir el esfuerzo a realizar por los animadores.

Palabras clave: *Blender*, controles de animación, *Keyframe Animation*, personajes 3D.

Índice general

Índice de figuras

Índice de tablas

Lista de códigos fuentes

Introducción

Desde tiempos remotos el hombre ha necesitado hacer uso de la tecnología para mostrar el entorno que le rodea, sus vivencias y comportamiento dentro de la sociedad. Con el desarrollo de la computación uno de los campos más populares ha sido la animación, que se ha empleado con éxito en los videojuegos, la medicina, en la industria aeroespacial, el cine y en las producciones audiovisuales en general.

Impregnar de movimiento a los objetos que son simulados, al igual que sucede en la vida real, ha sido imprescindible para lograr realismo en las producciones visuales. Este movimiento se logra cuando un grupo de dibujos estáticos, los que son ligeramente diferentes unos de otros, son mostrados en un orden secuencial y a una velocidad que da la ilusión de movimiento, también conocida como animación (WILLIAMS, 2001).

La animación es un proceso complejo, que requiere de varias técnicas que se han ido perfeccionando según el avance de las tecnologías. En los últimos años, con las innovaciones en animaciones con computadoras, la animación ha dado un nuevo paso evolutivo. La animación en 3D ha otorgado al artista animador mayor libertad y realismo a sus producciones, logrando mayor aceptación por parte del público que su predecesora dos dimensiones (2D). Los campos como la medicina, arquitectura, derecho, e incluso la medicina forense ahora utilizan la animación 3D (BEANE, 2012).

La industria del entretenimiento es considerada una de las más reconocidas que hacen uso de la animación 3D, e incluye el cine, la televisión y los videojuegos. En ella los personajes virtuales han tomado un rol muy importante. Estos son utilizados en aventuras, en juegos educativos, situaciones peligrosas (ver Figura 1) entre otras.

En el mundo de la animación 3D al igual que sucede en otros campos de la computación existen varias herramientas que permiten el diseño en esta tecnología, conocidas como paquetes integrados de animación o *suites* de animación.

En Cuba como parte de la política del Partido Comunista de Cuba (PCC) y la Revolución se define en el año 2011 el lineamiento 163:

- "Continuar **fomentando la defensa de la identidad, la conservación del patrimonio cultural**, la creación artística y literaria y la capacidad para apreciar el arte..." (PCC, 2011).

En este sentido desde hace varios años se realizan esfuerzos por incorporar la tecnología 3D al desarrollo audiovisual como medio para obtener productos más realistas, amenos y aceptados por el público en general. El Instituto Cubano del Arte e Industria Cinematográficos (ICAIC) en el año 2014 estrenó su primer

(a) Largometraje Meñique. (b) Largometraje UP.

(c) Juego Aventuras en la manigüa. (d) Juego *Call of duty.*

Figura 1. Ejemplos de la utilización de personajes en productos generados en 3D

largometraje en 3D: "Meñique" (CUBADEBATE, 2014) (ver Figura 1(a)), que contó con la colaboración de instituciones extranjeras y nacionales como es el caso de la Universidad de las Ciencias Informáticas (UCI).

La UCI ha jugado un papel importante en esta rama, ya que se han desarrollado varios productos que utilizan la tecnología 3D, en su mayoría en el centro VERTEX, este se enfoca fundamentalmente en el desarrollo de aplicaciones de Diseño Asistido por Ordenador, por sus siglas en inglés (CAD) y videojuegos, como: "Aventuras en la manigüa", "Chivichana" y "Especies Invasoras", laboratorios virtuales, la propia colaboración en el largometraje "Meñique", videos musicales, *spot* publicitarios, entre otros (CUBADEBATE, 2015).

En el centro VERTEX para llevar a cabo el desarrollo de videojuegos se cuenta con un pequeño grupo de diseñadores y especialistas generales en 3D que se encargan de elaborar todo el contenido visual, dígase: modelos 3D, texturas y animaciones. Para esto es necesario la utilización de *software* que permita este propósito, como son las *suites* de animación 3D. Atendiendo a que una de las principales metas de la UCI es la migración a *software* libre (NICADO GARCÍA, 2014), se utiliza como principal *software* de modelado y animación *Blender*.

Entre los componentes que forman parte de la animación de personajes 3D se encuentra el "*rigging*", que es el proceso de crear los "*rigs*". Estos son sistemas que le brindan al animador la flexibilidad y el control necesarios para mover un objeto 3D determinado (BEANE, 2012). Los controles, en el *rig* de un personaje, son los nodos que controlan los movimientos de las articulaciones.

Tanto en el *software Blender*, como en la mayoría de los empleados en la animación 3D, los controles se muestran por encima del personaje, que según la complejidad del *rig*, pueden solaparse y por consiguiente dificultar la selección adecuada de los controles durante el proceso de animación.

El contexto planteado anteriormente trae consigo las siguientes dificultades:

- Se dificulta la visibilidad de otros controles, por lo que la selección puede llegar a ser ambigua al realizarse sobre personajes que pueden estar llenos de controles.
- Se limita la visibilidad de transformación del personaje al animador, trayendo consigo que el animador deba ocultar los controles para poder visualizar la creación activa y mostrarlos nuevamente para seleccionar un nuevo control y transformar la pose.
- Se orbita constantemente alrededor del personaje para realizar la selección del control deseado.

Todo ello provoca demora al proceso de animación 3D que se realiza en el centro, lo que puede provocar atrasos en el proceso de desarrollo.

Ante la problemática expuesta se plantea el siguiente **problema de investigación**: *¿cómo realizar la selección de controles de personajes 3D, de manera que se facilite el proceso de animación en el centro VERTEX?* Para ello se define como **objeto de estudio**: *el proceso de animación de personajes en los software de gráficos 3D*. Como **objetivo general**: *desarrollar una herramienta que permita seleccionar controles de animación de personajes 3D en Blender, sin la visualización de los mismos sobre el personaje*. Todo ello enmarcado en el **campo de acción**: *selectores de controles de animación de personajes 3D*.

Para alcanzar el objetivo general se trazan las siguientes **tareas de investigación**:

1. Caracterizar el proceso de animación de personajes 3D y las *suites* de *software* de animación 3D más utilizados.
2. Caracterizar los selectores de controles de animación existentes, con el objetivo de incluir aquellos que sean útiles para el diseño de la solución.
3. Describir la integración de nuevas funcionalidades a *Blender*.
4. Seleccionar el lenguaje de programación, Entorno de Desarrollo Integrado, por sus siglas en inglés (IDE) y la metodología de desarrollo de *software* a utilizar durante el desarrollo de la investigación.
5. Realizar la descripción de los requisitos funcionales para dar solución al problema planteado.
6. Diseñar la propuesta de solución.
7. Implementar la solución propuesta.
8. Realizar pruebas para validar la solución propuesta.

Para llevar a cabo el desarrollo de la presente investigación se emplearon los siguientes métodos científicos:

Métodos teóricos:

- Histórico - Lógico: para realizar un estudio crítico de los sistemas existentes en el contexto de animación de personajes 3D y utilizarlos como punto de referencia y comparación, además de constatar teóricamente cómo ha evolucionado el tema en el tiempo.

- Análisis y síntesis: se empleará para abstraer todos lo elementos esenciales referente a la animación 3D y su relación en el proceso de animación de personajes en los *software* de animación 3D.
- La modelación: se usará en la realización de los artefactos de acuerdo a la metodología de desarrollo de *software* seleccionada, modelando a su vez, una representación abstracta de la solución que facilite el desarrollo de la herramienta.

Métodos empíricos:

- La observación: a través de este método se realizará una observación sobre la evolución y tendencia del proceso de animación 3D.

Estructura del documento

Para mostrar los resultados de las tareas de investigación. El documento se encuentra estructurado en tres capítulos:

- **Capítulo 1: Marco Teórico**:
  En este capítulo se realiza un análisis sobre el proceso de animación 3D, las principales herramientas empleadas en este campo y otros conceptos relacionados con el estado del arte. Además se seleccionan las herramientas y tecnologías a utilizar en la etapa de desarrollo; y la metodología de desarrollo de *software*.
- **Capítulo 2: Propuesta de Solución**:
  Se describen los requisitos funcionales y no funcionales y generan los artefactos de acuerdo a la metodología seleccionada. Se presenta la propuesta de solución, el modelado y diseño de la herramienta de selección de controles para *Blender*.
- **Capítulo 3: Implementación y Prueba**:
  Se describen los componentes desarrollados en la fase de implementación y realizan pruebas para validar la solución propuesta.

CAPÍTULO 1

Fundamentación Teórica

Introducción

En el presente capítulo se describe el proceso de animación de personajes 3D, así como sus conceptos fundamentales. Se realiza un estudio sobre el proceso de animación de personajes 3D en los *software* de animación más utilizados y las herramientas para la selección de controles que estos utilizan. Se describe la metodología de desarrollo de *software* a emplear, el lenguaje de modelado para la representación gráfica de los productos de trabajo que formarán parte de la solución y se seleccionan las herramientas y tecnologías para la implementación de la propuesta de solución.

1.1. Proceso de animación de personajes 3D

En el proceso de producción 3D existen distintas etapas cada una enfocada en un aspecto determinado, (ver Figura 1.1).

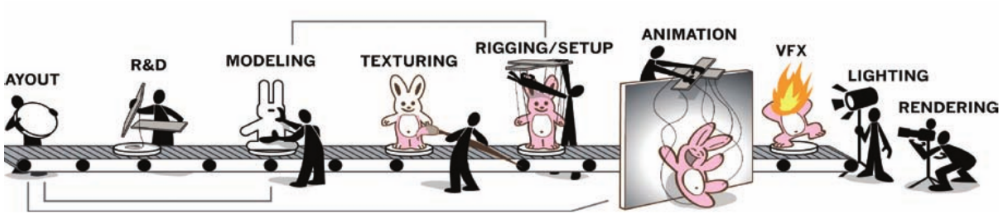


Figura 1.1. Proceso de producción.

La etapa de animación se centra en la creación del movimiento de los objetos y personajes[1], de forma que la audiencia crea que estos son reales. Una de las vías más comunes para la animación de personajes se basa en lograr que cambie su aspecto, basado en el movimiento de otro objeto, por ejemplo su armadura (BEANE, 2012). La animación parte de los elementos desarrollados en etapas anteriores como el *rigging*,

[1]En el contexto de la presente investigación se asume personaje 3D como: Ser virtual diseñado mediante tecnología 3D.

en el que se crea la estructura interna del personaje (esqueleto con las articulaciones), y se configuran los controles del personaje que las manejan, para lograr que el animador las pueda mover.

En los estudios realizados por BEANE, se hace referencia a tres tipos de animación 3D: *motion capture* (en la cual el animador transfiere el movimiento capturado desde un actor a un *rig* de controles y perfecciona el movimiento capturado), *procedural animation* (en la cual un programador crea un grupo de reglas y el personaje se mueve de acuerdo a esas reglas) y *keyframe animation* (en la cual el animador crea cada pose y un conjunto de fotogramas claves para ellas). En VERTEX la técnica utilizada para la animación de personajes es *keyframe animation*, también conocida como animación por fotogramas clave.

1.1.1. Animación por fotogramas claves

Los estudios realizados por BEANE, la señalan como una de las técnicas de animación más populares y utilizadas en la mayoría de las industrias. Esta es similar a la tradicional animación 2D. En ella el animador crea una pose del personaje u objeto y la establece en un punto específico del tiempo. Luego el animador construye una segunda pose y la coloca en otro punto del tiempo. La animación 2D trabaja de la misma manera, exceptuando que las poses deben ser dibujadas.

Para entender la animación por fotogramas claves es necesario abordar algunos conceptos:

- **poses**: Son valores de una determinada posición, de los controles de las articulaciones de un modelo. (WILLIAMS, 2001)
- **posiciones claves**: Posiciones que cuentan la historia, describen qué sucede en el plano (WHITE, 2009).
- **posición de pase**: Es la posición que se encuentra exactamente a mitad de camino de dos posiciones claves (ibíd.).
- **posiciones intermedias**: Son las posiciones secundarias que se encuentran exactamente entre las posiciones claves y las posiciones de pase (ibíd.).
- **transición**: son calculadas por la computadora a partir de las posiciones intermedias, basándose en una curva de tipo *spline*, que conecta los valores de las posiciones claves (LASSETER, 2001).

Un boceto de cómo sería una caminata básica en 2D, en la que se ejemplifican los conceptos abordados, se muestra en la Figura 1.2.

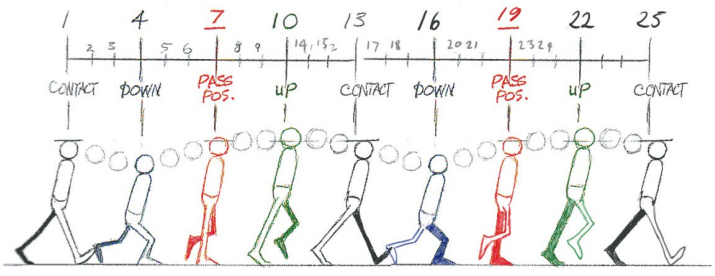


Figura 1.2. Boceto de caminata básica.

Para establecer esas poses a un personaje 3D es necesario algún tipo de sistema que provea al animador del control y la flexibilidad necesaria para mover al personaje de una manera específica. Este sistema de control es nombrado *rig* (MCLAUGHLIN; CUTLER y COLEMAN, 2011); (BEANE, 2012).

1.1.2. *Rigging*

El *rigging* es el proceso de dotar a un personaje de un conjunto de manipuladores y controles (*rig*) para facilitar el trabajo del animador (MA; RHEE y YOSHIYASU, 2015). Este es un proceso fundamental de la animación del personaje, donde varios controles configurados son agregados a cada parte del esqueleto del cuerpo. Este proceso comienza colocando un esqueleto compuesto por huesos o articulaciones dentro de un objeto o personaje. Luego se crean los controladores que permitirán a los animadores mover o rotar estos huesos . Por último, con el uso de deformadores para conectar la geometría al sistema de huesos para que la geometría del personaje siga al *rig* subyacente.

La mayoría de los especialistas ((MCLAUGHLIN; CUTLER y COLEMAN, 2011); (BEANE, 2012)) identifican dentro del *rigging* tres sistemas interconectados: **sistema de movimiento, sistema de controles, sistema de deformación.**

Sistema de movimiento

Se le denomina a la combinación de articulaciones que proporciona el sistema esquelético de los personajes (MCLAUGHLIN; CUTLER y COLEMAN, 2011). En la Figura 1.3 se puede apreciar un ejemplo de este modelo.

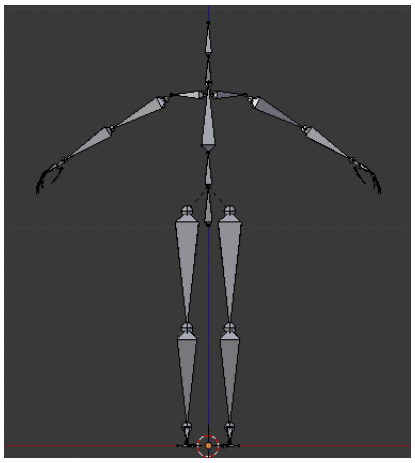


Figura 1.3. Ejemplo de Modelo de esqueleto.

Sistema de controles

Es la conexión de controles a las articulaciones que permite manejar el sistema de movimiento. Este es el punto de acceso del animador para definir la acción del personaje.

MA; RHEE y YOSHIYASU, se refiere a los **controles** como simples articulaciones, manejadores, o incluso ventanas separadas de selección de personajes. De acuerdo con las valoraciones de FERNÁNDEZ BAENA el propósito principal de los controles es interpretar y producir el movimiento pertinente en la forma del personaje atendiendo a la entrada del animador. En la Figura 1.4, se pueden apreciar un conjunto de controles, definidos gráficamente mediante formas. Realizar una definición adecuada de los controles así como la cantidad de estos, facilita y mejora el sistema de movimiento del personaje y por consiguiente el proceso de animación.

Figura 1.4. Controles empleados para modificar la pose del personaje (LOW, 2014).

En este ámbito, resulta importante para lograr un buen diseño del sistema de control, conocer cómo los artistas interactúan con un modelo de personaje, además de la capacidad para realizar un diseño elegante que permitan dar solución a los más complejos movimientos que se puedan presentar (MCLAUGHLIN; CUTLER y COLEMAN, 2011).

El diseño del sistema de control permite definir en toda su dimensión los objetos con los que va a interactuar el animador así como el modo en que estos objetos afectan el modelo del esqueleto, (ver Figura 1.3).

Los sistemas de control están dados a contener mayor cantidad de alteraciones que el sistema de movimiento, esta variedad de cambios implican un reto para el artista animador, que debe trabajar con mayor precisión para lograr el resultado esperado con el menor número de afectaciones.

Según la Real Academia de la Lengua Española, (ESPAÑOLA, s.f.) la selección es la “acción y efecto de elegir a una o varias personas o cosas entre otras, separándolas de ellas y prefiriéndolas”. Por lo que se considera para la presente investigación el siguiente concepto:

DEFINICIÓN **1.1.1.** ***Selección de controles****: Es la acción de elegir una articulación de un personaje 3D con el objetivo principal de lograr su traslación, rotación o escala.*

Esta selección es el paso principal para la creación de las poses de los personajes, juega un papel fundamental dentro del proceso de animación.

Sistema de deformación

En el sistema de deformación se define el desplazamiento de la geometría del modelo con relación a la animación del sistema en movimiento (MCLAUGHLIN; CUTLER y COLEMAN, 2011). De este modo se logra cambiar la forma del objeto (BEANE, 2012). Estas deformaciones permiten que la geometría se mueva más realista con pliegues y piel. Para definir un buen sistema de deformación también es importante que el artista animador sea capaz de comprender de forma adecuada la anatomía del personaje y posea "tacto" para preservar la forma de un modelo y los detalles durante el movimiento. Existen varios métodos que permiten realizar la transformación de un personaje dependiendo de los requisitos del modelo y la complejidad de la línea de producción.

1.2. *Suites* de Software de animación 3D

Para llevar a cabo tanto el proceso de animación como el modelado, simulación, efectos visuales y la representación de los proyectos se han desarrollado distintas *suites* de animación (ibíd.). Cada una de estas *suites* de animación tiene características específicas que la hacen más potentes en un área determinada. Entre las más conocidas se encuentran:

1.2.1. Autodesk Maya

Es una de las *suites* más populares empleadas para la animación 3D, la renderización, el modelado, la simulación de fluidos y de otros elementos, la composición y el rastreo de movimiento (ROMERO SANTILLÁN, 2014). Provee poderosas herramientas necesarias para darle movimiento a los personajes y objetos en la escena. Posee una edición libre para su estudio y aprendizaje; y una comercial cuyo costo asciende hasta los 3495 USD.

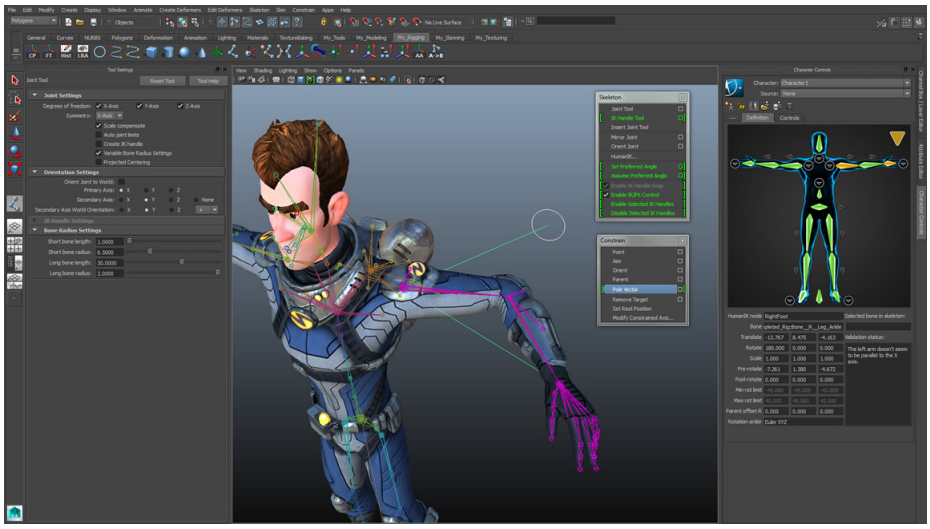


Figura 1.5. Selección de controles de personajes 3D en Maya.

Para la selección de controles en esta herramienta una de las formas definidas es exteriorizar los controles fuera de la malla del personaje para facilitar su visibilidad y selección; otra técnica es la de dar una "vista de rayos-x" al personaje (ver Figura 1.5), esto permite ver a través de la malla los controles que se encuentran dentro de esta.

Las dos técnicas mencionadas anteriormente se consideran poco eficientes. Entre las desventajas que poseen estas técnicas se encuentran las siguientes:

- La percepción del animador sobre la transformación que sufre la malla del personaje se dificulta, debido a la poca visibilidad que se tiene de esta.
- A medida que aumenta la complejidad del *rig* del personaje se hace poco efectiva la selección de los controles, debido al solapamiento de estos a la vista del animador.

Para hacer frente a estas deficiencias se han desarrollado varias herramientas que permiten realizar la selección de los controles desde un panel aparte, entre las que se encuentran: *abxPicker* y *Animation Picker Tool*, ver epígrafe 1.3.

1.2.2. Maxon Cinema 4D Studio

Es una *suite* de código propietario en su versión comercial, aunque también posee una versión libre destinada para uso académico, que presenta algunas limitantes. Esta *suite* facilita la creación de animaciones y gráficos 3D. Su interfaz gráfica (ver Figura 1.6) se puede manejar con cierta facilidad, al tiempo que resulta flexible y personalizable. Es una herramienta modular, que permite agregar componentes independientes tales como (Módulo de Renderizado, Módulo de Dinámicas, Huesos, Pelo, Partículas, Animación Compleja) en función a las necesidades (ROMERO SANTILLÁN, 2014). En esta herramienta, la selección de controles

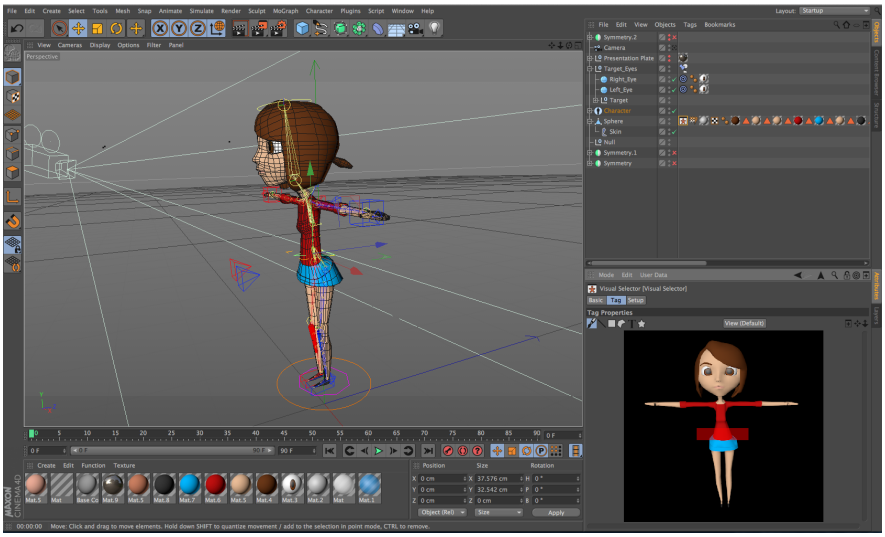


Figura 1.6. Selección de controles de personajes 3D en Maxon Cinema 4D Studio.

de personajes se realiza de manera similar a la analizada anteriormente. Para brindar al animador una alternativa que permita agilizar el proceso existen herramientas como: *Visual Selector*, que se integra al *software* como un *add-on*. Ver epígrafe 1.3.

1.2.3. Blender

Blender es una *suite* libre y de código abierto para la creación de contenido 3D de modelado, mapeo UV, texturización, *rigging*, integración de partículas, renderizado, post-producción, creación de juegos, *skinning*, animación y simulaciones (ibíd.). En la Figura 1.7 se puede observar la interfaz gráfica de *Blender*.

Soporta el ciclo completo de creación 3D. Es multiplataforma y se ejecuta de igual manera en los Sistemas Operativos (SO) Linux, Windows y Macintosh. Posee además otras características entre las que se encuentran:

- Fácil distribución.
- Detección de colisiones, recreaciones dinámicas y lógica, para juegos interactivos.
- Variedad de primitivas geométricas, que incluyen curvas, *B-splines racionales no uniformes (NURBS)*, *metaballs*, mallas poligonales.
- Se adapta con facilidad a formatos gráficos como Iris, SGI, o TIFF, TGA, JPG.

La selección de controles en esta *suite* se realiza sobre los personajes 3D, de manera similar a como se realiza en *Autodesk Maya*, lo que causa las mismas deficiencias. Es por ello que se desarrollan herramientas para solucionar estos problemas como es el caso de *BlenderRig*, que se integra como un *add-on* a *Blender* (ver epígrafe 1.3).

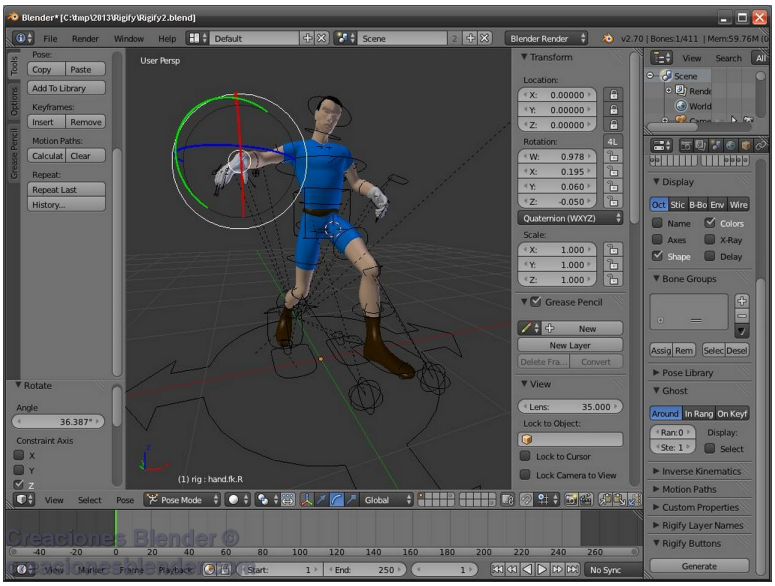


Figura 1.7. Interfaz gráfica de *Blender*

1.3. Herramientas para la selección de controles de animación

En la animación de personajes no hay una "manera exacta" de hacer las cosas, todo depende de la interpretación del animador. Este es un trabajo completamente artístico (BEANE, 2012). Para brindarle al animador una mejor interpretación del movimiento que realiza el personaje, son creadas herramientas para la selección de los controles, que permiten al artista animador una mejor interacción con el personaje, entre ellas se encuentran los *Character Pickers* o Sinópticos.

Los *Character Pickers* son interfaces gráficas que permiten interactuar con un personaje, mediante la selección de elementos denominados puntos de acceso (botones), que se asocian a los controles del personaje. Estas permiten seleccionar los controles, sin tener que mostrarlos durante el proceso de animación, lo que posibilita una vista más natural de la escena y su movimiento. En la Figura 1.8 se muestran los elementos que conforman un *Character Picker* o Sinóptico.

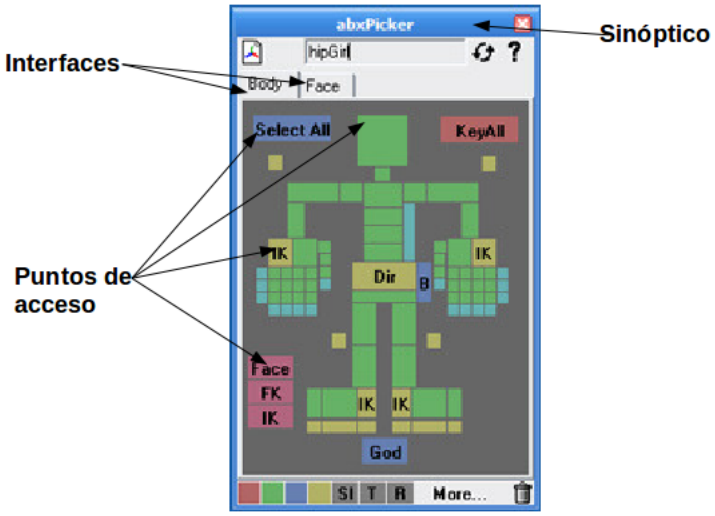


Figura 1.8. Elementos que componen un Sinóptico.

A continuación se identifican las características de herramientas para la selección de controles de animación de personajes 3D.

1.3.1. abxPicker

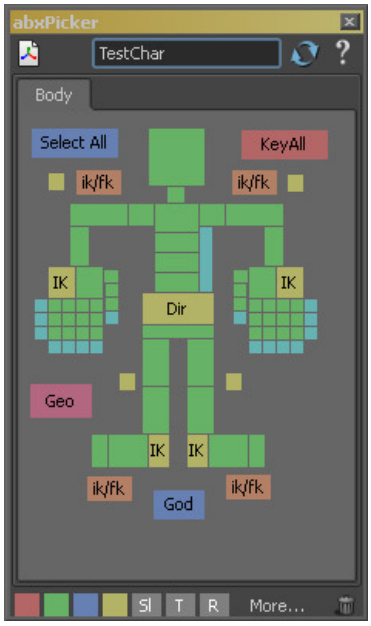


Figura 1.9. Interfaz gráfica de la herramienta abxPicker.

La herramienta *abxPicker* permite la creación de interfaces de forma personalizada para la selección de los controles de un personaje. Se puede integrar a la *suite* de animación Maya. Posee una interfaz que

permite realizar acciones como soltar y arrastrar. Se han identificado entre sus principales características las siguientes:

- Para su utilización el usuario arrastra un punto de acceso hasta el lugar deseado de la interfaz, una vez allí lo suelta y redimenciona en caso de ser necesario.
- Asignación de teclas de forma independiente para cada botón.
- Los datos son almacenados en los nodos de la escena para facilitar su portabilidad.
- Importar / Exportar a otros *rigs*.
- Permite añadir imágenes de fondo a la interfaz.

1.3.2. Visual Selector

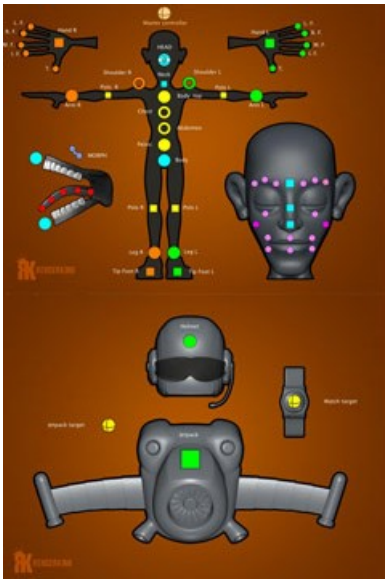
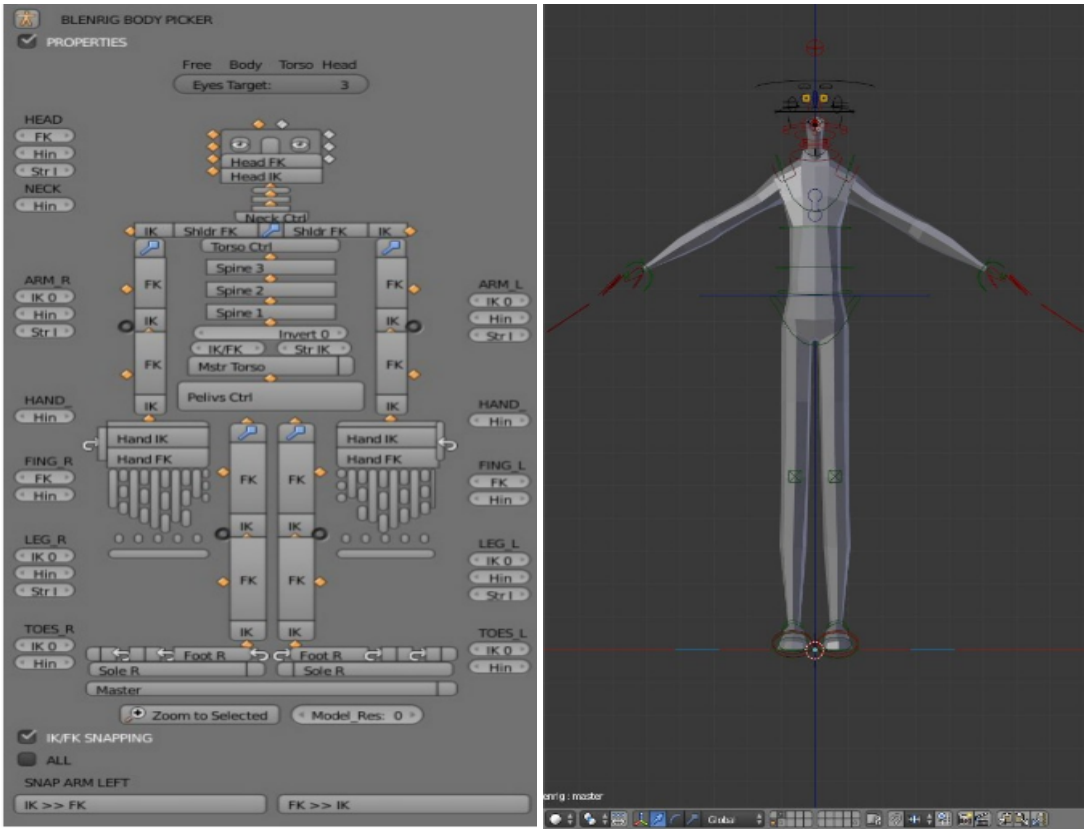


Figura 1.10. Interfaz gráfica de la herramienta Visual Selector.

Es una herramienta desarrollada para Cinema 4D. Cuenta un panel que permite cargar una imagen de fondo y asignar sobre esta objetos que pueden estar relacionados con los controles del personaje. Al escoger un objeto en el selector visual, el control del personaje asociado a este se selecciona.

Este tipo de herramienta facilita la selección de los controles de los personajes mediante el uso de puntos de acceso, lo que permite que los controles no estén visibles sobre los personajes. Posibilita la realización de una selección efectiva cuando se utiliza una imagen de fondo del propio personaje, lo que es especialmente útil para personajes con un *rig* muy cargado de controles.

1.3.3. BlenderRig



(a) Sinóptico asociado al *BlenderRig*.

(b) Rig por defecto de *BlenderRig*.

Figura 1.11. Interfaz gráfica del panel de selección del *BlenderRig* y su *rig* asociado.

Esta herramienta proporciona al animador una plataforma que incluye el tratamiento a sistema facial de forma avanzada. En la Figura 1.11 puede apreciarse un ejemplo de su interfaz gráfica.

Se pueden identificar como principales características las siguientes:

- Incorpora su propio *rig* del personaje, además cuenta con varios paneles, en los que se encuentran ya asociados los selectores a los controles del personaje.
- Define grupos de colores para las distintas partes de la armadura, lo que facilita la visualización de los controles del personaje.

Un inconveniente que presenta esta herramienta es su poca flexibilidad, debido a que viene con un *rig* predefinido, con sus puntos de accesos y selectores de controles ya asociados. Esto limita las posibilidades del animador para gestionar los controles en otros personajes 3D, al no permitir la asignación de otros *rigs*.

1.3.4. Análisis realizado a las herramientas de selección de controles

Los criterios que se definen a continuación, se basan en las necesidades existentes en VERTEX para la animación de personajes 3D. Estos criterios, junto a la revisión bibliográfica realizada a los sistemas anteriores, posibilitan la realización de un análisis del estado actual de las aplicaciones informáticas de animación de personajes 3D enmarcado en el campo de acción de la investigación. Los resultados de este estudio se resumen en la Tabla 1.1.

Tabla 1.1. Estudio realizado a las herramientas de selección de controles

Criterios	**Herramientas**		
	abxPicker	***Visual Selector***	***BlenderRig***
Flexibilidad	Sí	Sí	No
Software Libre	Sí	Sí	Sí
Integración con *Blender*	No	No	Sí

El estudio realizado a las herramientas permitió identificar características esenciales como las que se presentan en la Tabla 1.1. De las herramientas analizadas anteriormente solo *BlenderRig* puede ser utilizada en *Blender*, aunque tiene la limitante de estar configurada únicamente para su propio *rig* en forma de humanoide. Esto no resulta suficiente para lograr proyectos de animación donde la mayoría de las veces interactúan personajes variados que pueden ir desde animales hasta vehículos móviles. El proceso de incorporar un nuevo *rig* al *BlenderRig* resulta extremadamente complejo y significaría la reimplementación casi desde cero de la herramienta.

En ese sentido, herramientas como *abxPicker* tienen como ventaja que no vienen con un *rig* preconfigurado, y permiten al animador asociar los puntos de acceso (botones) a cualquier control del personaje sin importar qué *rig* esté presente. En *BlenderRig* no se puede asignar un botón de la interfaz a un control en tiempo real y no se brinda al animador información en la interfaz del control seleccionado en el *rig*. Debido a esto, el animador de manera gráfica no sabe exactamente cuál o cuáles controles están seleccionados, característica considerada necesaria para los animadores del centro VERTEX ya que se evitan movimientos no deseados y se agiliza el proceso de animación.

Por todo lo anteriormente mencionado se evidencia la necesidad de desarrollar una herramienta que permita la selección de controles de animación, mediante la asociación de cada botón de la interfaz al control deseado del personaje, y brinde información para determinar qué control se selecciona en un momento dado.

1.4. Vías para extender las funcionalidades de Blender

Blender es una *suite* en constante desarrollo, que cuenta todavía con muchas deficiencias (LÓPEZ SANTADER y CHAHIN GARCÍA, 2014). Esta *suite* es desarrollada por contribuidores; por artistas y programa-

dores los cuales van agregando funcionalidades a *Blender*. Estas nuevas funcionalidades pueden ser integradas a futuras versiones de la *suite*. Para integrar funcionalidades a *Blender* existen dos vías, dependiendo de las necesidades:

1. Desarrollando una nueva versión de *Blender*, cuando los cambios se basan en características avanzadas de la herramienta como modificadores de malla y tipos de objetos, esto conlleva adentrarse en el código de C/C++ de *Blender* para la realización de estos (FOUNDATION, s.f.).
2. Para características referidas a la interfaz gráfica, que no requieran modificar las funciones fundamentales del sistema, se utiliza la Interfaz de programación de aplicaciones, por sus siglas en inglés (API)[2] de *Blender* para *Python*. Esta permite la creación de funcionalidades mediante el desarrollo de *script*[3], los que se pueden integrar de dos formas a *Blender*: a través de la ejecución del *script* diseñado para ello o como un *add-on* que puede activarse por el usuario. De esta forma, no es necesaria la modificación del código fuente C/C++ de *Blender*.

1.4.1. Add-ons

Es un componente de *software* que le agrega una funcionalidad específica a un programa ya existente. En *Blender*, los *add-on* son tratados como tipos especiales de *script*, con algunos requerimientos e información básica que se muestra en una lista. Son una manera de encapsular un *script* de *Python*, para hacer más fácil su uso por parte de los usuarios. Pueden ser habilitados o deshabilitados, cuando el usuario lo desee en las "Preferencias de Usuario" (ver Figura 1.12).

[2]Una API es un conjunto de comandos, funciones y protocolos que los programadores pueden utilizar en la construcción de un software. La API permite a los programadores utilizar funciones predefinidas para interactuar con el SO, en lugar de escribir desde cero.

[3]Un guión o *script* no es más que un fichero de texto plano que contiene una lista de comandos de cierto lenguaje. Estos lenguajes se llaman interpretados, ya que en vez de compilar un fichero fuente hasta obtener uno ejecutable son interpretados directamente de la fuente (MARTÍNEZ, 2004).

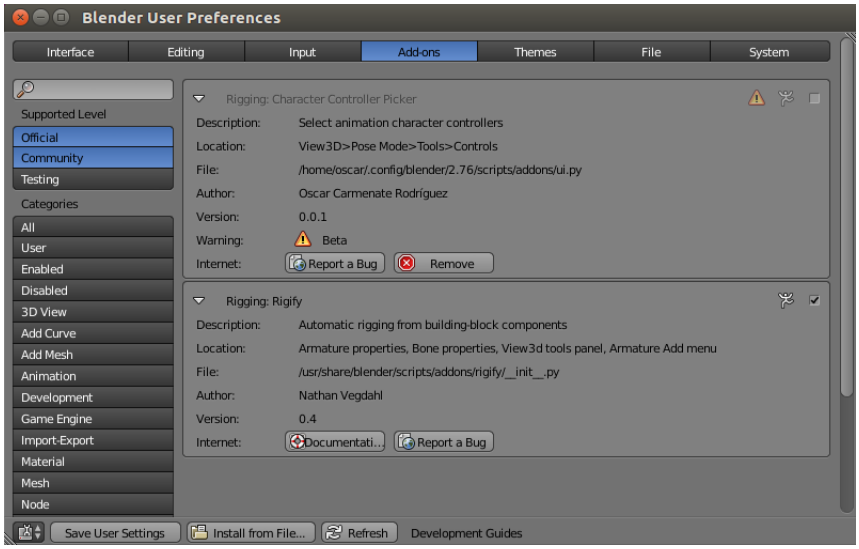


Figura 1.12. *Add-on* en *Blender*.

Por lo antes expuesto, se determina para el desarrollo de la herramienta, como vía de extensión de *Blender* la elaboración de un *script* de tipo *add-on*, para lo cual se utiliza la API de *Blender* para *Python* debido a que lo que se va a desarrollar es una herramienta que le agrega una funcionalidad específica a Blender que no requiere la modificación del código fuente de este.

1.4.2. API de Blender para Python

Para el desarrollo de *script* en *Blender* se utiliza la API de este para Python. Entre sus principales funcionalidades están (FOUNDATION, s.f.):

- Edición de los datos de la interfaz de usuario (Escenas, Mallas, partículas, entre otras).
- Modificación de las preferencias de usuario, temas y mapas de teclado.
- Ejecución de herramientas con su propia configuración.
- Creación de elementos de la interfaz de usuario, como menús, encabezados y paneles.
- Creación de nuevas herramientas.
- Creación de herramientas interactivas.
- Creación de nuevos motores de renderizado que se integran con *Blender*.
- Definición de nuevos ajustes en los datos existentes de *Blender*.
- Dibujo en la vista 3D, mediante los comandos de *OpenGL* desde *Python*.

La API aún no puede:

- Crear nuevos tipos de espacios.
- Asignar propiedades personalizadas para cualquier tipo.
- Definir *callbacks* o *listeners* para notificar cuando se modifican los datos.

- Cargar imágenes en los paneles, lo que limita la información que se le brinda al animador referente a la asociación de los controles con los puntos de acceso.
- Arrastrar y soltar sobre los paneles, de modo que se dificulte su personalización.

1.5. Entornos de desarrollo

Blender posee un editor, que resulta muy útil para hacer pequeños cambios y pruebas al código. Este no es recomendable cuando se desarrollan *scripts* ibíd., ya que la interfaz de Blender puede dificultar el proceso, debido a la cantidad de pasos que se deben realizar como: la recarga manual, la ejecución del *scripts*, entre otros. Es por eso que se hace necesario la utilización de un IDE por las facilidades que brinda al programador: autocompletamiento, resaltado de sintaxis, entre otras. Existe una amplia variedad de ellos que pueden ser utilizados en dependencia del tipo de proyecto, el lenguaje utilizado, la plataforma en la que se trabaje y otros elementos de orden general que forman parte del ambiente de producción. En la comunidad *Python* existe una lista de editores que cubren una amplia gama de plataformas y licencias de *software* (PYTHON, 2016), entre ellos se encuentran:

1.5.1. PyCharm

Es un IDE para *Python*, provee al desarrollador de un completamiento de código inteligente, inspección de código, un resaltado de error sobre la marcha, y arreglos rápidos, además de una refactorización de código automatizada, entre otras funcionalidades. Es multiplataforma, además cuenta con una versión con licencia gratis y otra de pago para entornos profesionales (JETBRAINS, s.f.). Se pueden señalar como desventajas: su gran consumo de recursos y que no soporta el completamiento de código para el uso de la API de *Blender*.

1.5.2. Eclipse

Este IDE puede ser utilizado para varios lenguajes gracias a los *plugins* que se le pueden integrar (ECLIPSE, s.f.). Para el soporte de *Python* utiliza el *plugin PyDev* (PYDEV, s.f.). Tanto *Eclipse* como *PyDev* son multiplataforma y de código abierto.

Esta unión de *Eclipse* con *PyDev* (de ahora en adelante *Eclipse-PyDev*) soporta completamiento de código, resaltado de sintaxis, análisis sintáctico, refactorización, depuración, depuración remota, entre otros (ibíd.). *Eclipse-PyDev* puede ser adaptado a la API de *Blender* (JAWORSKI, 2011). Esta integración, no obstante, es solo parcial, debido a que el autocompletamiento que genera no proviene directamente de los módulos de la API *Blender*.

Una vez analizadas las características de los IDE anteriormente descritos, se selecciona para el desarrollo de la herramienta el IDE *Eclipse-pydev*. Sobre todo, por la facilidad que brinda para la integración con *Blender*, el soporte que se le da a las actualizaciones de *Eclipse* en la universidad; y la experiencia acumulada en los equipos de trabajos de varios proyectos, como los del centro VERTEX.

1.6. Metodologías de desarrollo de software

En la actualidad para emprender exitosamente cualquier proyecto de desarrollo de *software* se hace necesario la utilización de metodologías de desarrollo, las que constituyen marco de trabajo que permiten realizar la planificación, diseño y ejecución del proceso de desarrollo con calidad (GARCÍA DOMINGUEZ y GÓMEZ BARROSO, 2009). Estas se pueden clasificar en:

- Ágiles.
- Tradicionales.

Las **metodologías tradicionales** se componen por una gran variedad de artefactos y un ciclo de desarrollo extenso que se debe cumplir de forma estricta, muy utilizada en proyectos con grandes equipos de trabajos que realizan especial atención al control del proceso (CANÓS; LETELIER y PENADÉS, 2003). Algunas de las más conocidas son: *Rational Unified Process (RUP)* y *Microsoft Solution Framework (MSF)*.

Por otro lado, en los últimos años ha tomado gran auge las **metodologías ágiles**, que a diferencia de las primeras, se centra en las personas y no en el *software* (ibíd.). Como principales características se tienen:

- Existencia de estrecha colaboración del equipo de trabajo con el cliente.
- Desarrollo incremental.
- Iteraciones cortas.

Este tipo de metodología es capaz de adaptarse a las condiciones propias del proyecto de desarrollo (PÉREZ RAMÍREZ; OLIVEROS GUNTÍN; ALVAREZ ALONSO y COELLO MENA, 2008) aún cuando estas sean cambiantes. En equipos de trabajo pequeños, su uso ha dado muy buenos resultados.

En la Tabla 1.2 se describen las principales diferencias existentes entre las metodologías ágiles y las tradicionales. Estos elementos, influyen de manera considerable no solo al proceso de desarrollo como tal, también al contexto que involucre al equipo de desarrollo y su organización.

Tabla 1.2. Diferencias entre metodologías ágiles y tradicionales

Metodologías Ágiles	**Metodologías Tradicionales**
Se basan en heurísticas provenientes de prácticas de producción de código	Basadas en normas de estándares seguidos por el entorno de desarrollo
Preparados particularmente para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
Flexibilidad en el contrato	Existencia de un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Equipos pequeños (menos de 10 integrantes), trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del *software*	La arquitectura del *software* es esencial y se expresa mediante modelos

Una de las metodologías ágiles más reconocidas internacionalmente es Programación Extrema, por sus siglas en inglés (XP), también utilizada en la UCI desde hace varios años en proyectos similares como es el caso de varios proyectos pertenecientes al centro VERTEX. La metodología XP tiene como premisa que las relaciones interpersonales son la clave del éxito en el proceso de desarrollo de *software* (ibíd.). Entre los elementos más importantes que la definen se encuentran (ECHEVERRY TOBÓN y DELGADO CARMONA, 2007):

- Promueve el trabajo en equipo.
- Retroalimentación constante entre el cliente y el equipo de desarrollo.
- Simplicidad en las soluciones.
- Muy adecuada en proyectos con requisitos imprecisos y cambiantes.

El proyecto que enmarca la presente investigación consta de un equipo pequeño de trabajo y está sujeto a cambios, debido a que se cuenta con un tiempo limitado se hace necesario generar sólo la documentación de ingeniería que sea necesaria. Además el cliente forma parte del equipo de trabajo, cuestión que facilita la comunicación. A partir del análisis de los elementos anteriores se selecciona la **metodología** XP para el proceso de desarrollo de la presente investigación.

1.7. Lenguaje de modelado de software

La utilización de los lenguajes de modelado le permite al equipo de desarrollo de *software* obtener modelos para tener una representación conceptual o física a escala de un proceso o sistema, con el fin de analizar su naturaleza, desarrollar o comprobar hipótesis o supuestos y permitir una mejor comprensión del fenómeno real al cual el modelo representa y permitir así perfeccionar los diseños, antes de iniciar la construcción de objetos reales y ayudar a los usuarios a visualizar el producto final (PONS; GIANDINI y PÉREZ, 2010). El lenguaje utilizado para el modelado de la aplicación es Lenguaje de Modelado Universal, por sus siglas en inglés (UML).

1.7.1. Lenguaje Unificado de Modelado

Es un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios y pretende unificar diferentes técnicas de modelado.

UML incluye conceptos semánticos, notación, y principios generales de un sistema además de que permite representar decisiones a tomar durante la implementación. Contiene además construcciones organizativas para agrupar los modelos en paquetes, lo que permite dividir grandes sistemas en piezas de trabajo más simples (RUMBAUGH, 2000).

1.8. Herramienta para el modelado del sistema

La Ingeniería de Software Asistida por Computadoras, por sus siglas en inglés (CASE) es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias del desarrollo de *software*, su objetivo es acelerar el proceso de una o más fases del ciclo de vida del desarrollo de sistemas (K. KENDALL y J. KENDALL, 2005).

1.8.1. Visual Paradigm

Visual Paradigm for UML es una herramienta CASE que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un *software*. Permite integrarse con otras aplicaciones, como herramientas ofimáticas, permitiendo aumentar la productividad. Brinda la posibilidad de generar código de forma automática, reducir los tiempos de desarrollo y evitar errores en la codificación del *software* así como obtener diversos informes a partir de la información introducida en la herramienta (DOMINGO; RIUS y L., 2012).

Esta herramienta se selecciona para el modelado del sistema por la importancia del uso del *software* libre en Cuba, la existencia de una licencia otorgada a la universidad para su utilización y las características que posee.

1.9. Conclusiones parciales

- Como parte del análisis de las *suites* de animación 3D, en *Blender* se ha identificado la necesidad de desarrollar un sinóptico, cumpliendo con algunas de las características que poseen los sinópticos: *abxPicker* y *Visual Selector*. Se determinó que este sinóptico se integrará a *Blender* como un ***add-on***.
- Para desarrollar esta herramienta el lenguaje de programación seleccionado ha sido ***Python*** que es el utilizado para la programación de la API de *Blender*.
- Se selecciona el IDE ***Eclipse-pydev*** ya que constituye una alternativa muy acorde a las necesidades del equipo de desarrollo, sobre todo por la facilidad que brinda la integración con *Blender* y posibilidad de realizar *plugin* para él. En la universidad se le da soporte a las actualizaciones de *Eclipse* y se ha acumulado experiencia en los equipos de trabajos de varios proyectos como los del centro VERTEX debido a que es uno de los IDE más utilizado en este campus universitario.
- El proyecto que enmarca la presente investigación consta de un equipo pequeño de trabajo y está sujeto a cambios, debido a que se cuenta con un tiempo limitado se hace necesario generar sólo la documentación de ingeniería que sea necesaria. Además el cliente forma parte del equipo de trabajo, cuestión que facilita la comunicación. Todos elementos que hacen inclinar la selección de la **metodología** XP como una opción idónea para llevar el proceso de desarrollo.
- A partir del estudio realizado de la API de *Blender* para *Python* se determina que esta no cuenta con las funcionalidades encontradas en las herramientas estudiadas en 1.3, cuestión que limita la información que se le brinda al animador referente a la asociación de los controles con los puntos de acceso. En el presente trabajo, la herramienta que se propone estará basada en ***template***, en su versión inicial las interfaces estarán organizadas acorde a la armadura ***rigify***, que es una de las predefinidas en *Blender*. La arquitectura estará preparada para adicionar nuevos *template* que representan nuevas armaduras, posibilitando generalizar la propuesta de solución.

CAPÍTULO 2

Propuesta de solución

Introducción

El presente capítulo contiene la información relacionada con el análisis y el diseño de la propuesta de solución. Se describen las acciones llevadas a cabo para la construcción de la herramienta, guiadas por la metodología de desarrollo de *software* XP. Se describen las clases del dominio en cuestión y se especifican los requisitos funcionales y no funcionales, identificados como resultado de la aplicación de las técnicas de obtención de requisitos propuestas por la metodología.

2.1. Descripción del sistema

La propuesta de solución de esta investigación define una herramienta para la selección de controles de animación de personajes 3D en *Blender* con el nombre de: **CATB** (**C**ontrol **A**nimation **T**ool in **B**lender).

A continuación se tratan varios conceptos que son manejados en la solución.

Los *rigs* se asignan a un personaje para poder manejar el movimiento de este a través de los controles. El **animador** interactúa con un **sinóptico** para acceder a los selectores o puntos de acceso que permiten el manejo de los controles del personaje. Cada sinóptico está conformado por una o varias **interfaces**, las que se componen por **puntos de acceso**. A su vez, los puntos de acceso se asocian a un control en específico de un ***rig***, para facilitar la selección de este. Para una mejor aproximación a los conceptos antes expuestos se describe a continuación el modelo de dominio.

2.1.1. Modelo de dominio

El modelo de dominio es una representación visual de los objetos y conceptos (o partes esenciales), que se requieren en la modelación de un problema determinado, y las relaciones que subsisten entre ellos. Es una representación de las clases conceptuales del mundo real, no de componentes de *software* (GUIBERT

ESTRADA, 2011). En la Figura 2.1 se muestra el modelo de dominio que representa los conceptos del contexto referente a la selección de controles de animación de personajes 3D.

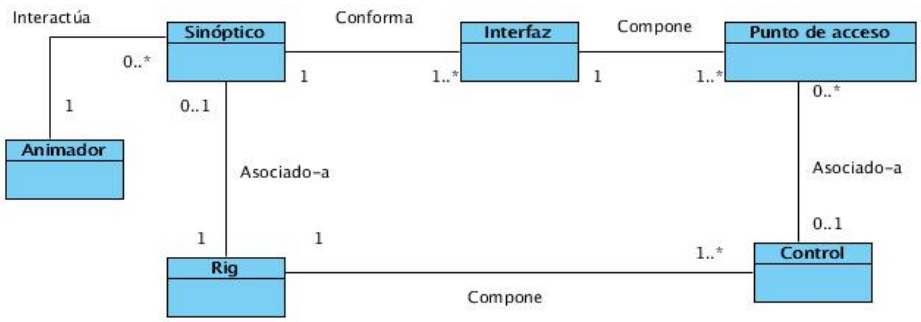


Figura 2.1. Modelo de dominio.

Animador:

Encargado de realizar la animación del personaje.

Punto de acceso:

Elemento visual al cual se le asocia un control del personaje.

Control:

Nodo que controla los movimientos de las articulaciones del personaje.

Interfaz:

Área donde se encuentran contenidos los puntos de acceso.

Sinóptico:

Interfaz gráfica mediante la cual los animadores interactúan con los *rigs*.

***Rig*:**

Sistema que provee al animador, del control y la flexibilidad necesaria para mover al personaje de una manera específica.

Para la definición de los tipos de interfaces iniciales que se usarán en la aplicación, se crea un plantilla basada en los controles que utiliza el *Rigify*, se organizan de forma gráfica los puntos de acceso que se asocian a esos controles en forma humanoide, debido a que este es uno de los *rigs* más utilizados por los animadores en el centro, para el diseño de los personajes. Una de sus principales ventajas es que automatiza la mayor parte del proceso de *rigging*.

2.1.2. Requisitos no funcionales

Los requisitos no funcionales son restricciones de las funciones o servicios ofrecidos por el sistema. No se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este. Definen restricciones sobre el sistema y el proceso de desarrollo de *software* (SOMMERVILLE, 2011). Los requisitos no funcionales definidos para el desarrollo de la herramienta propuesta se listan a continuación:

RnF. 1 **Usabilidad**: La interfaz debe ser intuitiva, fácil de usar e integrarse adecuadamente a la interfaz de *Blender*.

RnF. 2 **Soporte**: La codificación del sistema será guiada por los estándares de la API de *Blender*, lo que permite que pueda ser modificado en un futuro, para agregar o modificar sus funcionalidades.

RnF. 3 **Portabilidad**: El sistema deberá poder ser activado o desactivado por parte del usuario sin afectar las otras funcionalidades de *Blender*.

RnF. 4 ***Software*:**

- RnF. 4.1 Sistemas operativos *Linux*, *Windows* y *Mac*.
- RnF. 4.2 *Blender* en su versión 2.76.

RnF. 5 ***Hardware***: Los requisitos mínimos de *hardware* son los requeridos por *Blender* en su versión 2.76.

La metodología XP define un conjunto de reglas y prácticas que serán utilizadas para el desarrollo de la solución. La etapa inicial de todo proyecto en XP, es la planeación (ECHEVERRY TOBÓN y DELGADO CARMONA, 2007).

2.2. Planeación

En esta etapa se comienza a interactuar con el cliente y el resto del grupo de desarrollo para descubrir los requerimientos del sistema. Se identifican el número y tamaño de las iteraciones, al igual que se plantean ajustes necesarios a la metodología según las características del proyecto (ibíd.). Esta comienza con la toma de requisitos del sistema descritos en historias de usuario (HU).

2.2.1. Historias de usuario

Las siguientes historias de usuario contienen los requisitos funcionales del sistema, así como una breve descripción de lo que deben realizar. Son el resultado del análisis de las herramientas homólogas, el estudio del modelo de dominio y las reuniones con el cliente. Estas no profundizan mucho en el nivel de detalle, pues se cuenta con la participación del cliente. El cliente es el encargado de aclarar cualquier duda que se presente durante el desarrollo de la herramienta. La HU están ordenadas por prioridad según las necesidades del cliente.

Tabla 2.1. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Seleccionar control a través de un punto de acceso.
Usuario: Animador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3.0	**Iteración asignada:** 1
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: El usuario tendrá la posibilidad de seleccionar y deseleccionar un control del personaje a través de un punto de acceso, sin tener que recurrir a este directamente.	
Observaciones: En caso de existir un control seleccionado, la selección de otro no debe implicar que se deshabiliten los demás.	

Tabla 2.2. Historia de usuario # 2

Historia de usuario	
Número: 2	**Nombre:** Asociar punto de acceso a un control de animación.
Usuario: Animador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3.0	**Iteración asignada:** 2
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: Permite la asociación y desvinculación de puntos de acceso de un sinóptico, a un control de animación del personaje.	
Observaciones: Si el usuario selecciona varios puntos de acceso, en el momento de asociación, estos serán asignados a un solo control. Si el usuario elige más de un control de animación del personaje, el control que será asignado a los puntos de acceso seleccionados, será el control de animación activo.	

Tabla 2.3. Historia de usuario # 3

Historia de usuario	
Número: 3	**Nombre:** Gestionar interfaces del sinóptico.
Usuario: Animador	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1.0	**Iteración asignada:** 3
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: El usuario debe tener la posibilidad de añadir o eliminar una nueva interfaz al panel del sinóptico del personaje.	
Observaciones:	

Tabla 2.4. Historia de usuario # 4

Historia de usuario	
Número: 4	**Nombre:** Listar interfaces del sinóptico.
Usuario: Animador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2.0	**Iteración asignada:** 3
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: El usuario podrá observar en una lista, todas las interfaces que tiene asignadas actualmente el sinóptico del personaje.	
Observaciones:	

Las HU elaboradas anteriormente dan una idea al programador del alcance del *software* y permiten estimar el esfuerzo requerido para el desarrollo de la solución.

2.2.2. Estimación de esfuerzo por Historia de usuario

Luego de obtenidas las historias de usuario se le asignó un valor, el cual determina, según la experiencia del programador, cuanto esfuerzo requiere cada historia de usuario. Estos valores son mostrados en la Tabla 2.5.

Tabla 2.5. Estimación de esfuerzo por Historia de Usuario.

	Historias de Usuario	**Puntos estimados (semanas)**
1	Seleccionar control a través de un punto de acceso.	3.0
2	Asociar punto de acceso a un control de animación.	3.0
4	Gestionar interfaces del sinóptico.	1
5	Listar interfaces del sinóptico.	2

Después de haber estimado el esfuerzo necesario para el desarrollo de cada historia de usuario, el cliente selecciona qué construir, de acuerdo a sus prioridades y a las restricciones de tiempo.

2.2.3. Plan de iteraciones

En la metodología XP, el proceso de desarrollo del sistema se divide en iteraciones. Sobre este tema en particular, la mayoría de los investigadores (KNIBERG, 2007; JOSKOWICZ, 2008) consideran que la duración de cada una de las iteraciones no debe superar las tres semanas. Se define la duración de las iteraciones de desarrollo en la Tabla 2.6.

Tabla 2.6. Plan de duración de las iteraciones.

Iteración	Historias de Usuario		Duración (semanas)
1	1	Seleccionar control a través de un punto de acceso.	3.0
2	1	Asociar punto de acceso a un control de animación.	3.0
3	1	Gestionar interfaces del sinóptico.	3.0
	2	Listar interfaces del sinóptico.	
Total			**9.0**

Ya determinadas las HU, y con una aproximación de cuánto debe durar cada iteración, se diseña un plan de entregas, que establece las fechas de entrega al usuario de versiones funcionales de la aplicación.

2.2.4. Plan de entregas

El plan de entregas establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas (ibíd.).

Tabla 2.7. Plan de entrega de versiones.

Iteración	Fecha	Versión del Producto
1ra	7 de marzo	0.3
2da	1 de abril	7.6
3ra	2 de mayo	1.0

En la metodología XP, debido a los cambios que se presentan durante el desarrollo del proyecto, el diseño del sistema es constantemente revisado y modificado. En el siguiente epígrafe se describen las características fundamentales del producto a desarrollar, las que, como se prevé en XP, pueden estar sujetas a modificaciones.

2.3. Diseño

La metodología XP hace especial énfasis en los diseños simples y claros. Ofrece una guía de implementación para las HU con el propósito de reducir el riesgo al comenzar la verdadera implementación (PRESSMAN, 2010).

2.3.1. Descripción de la arquitectura

La solución se desarrollará e integrará a la API de *Blender* para *Python*, que presenta una arquitectura basada en componentes[1] o módulos, entre sus características se encuentran (TORRE LLORENTE; UNAI ZORRILLA CASTRO; MIGUEL ANGEL RAMOS BARROS; JAVIER CALVARRO NELSON, 2010):

- La descomposición del sistema en componentes con interfaces muy bien definidas.
- Aproximación al diseño a través de componentes que se comunican mediante interfaces que componen métodos, eventos y propiedades.
- Diseñar aplicaciones a partir de componentes individuales.
- Los componentes son diseñados de forma que puedan ser reutilizados en distintos escenarios en distintas aplicaciones aunque algunos componentes son diseñados para una tarea específica y pueden ser extendidos a partir de otros componentes para ofrecer nuevos comportamientos.

La API de *Blender* para *Python* está conformada por diversos módulos, entre los más importantes se encuentran:

***bpy.data*:**
Provee el acceso a los datos del archivo actual de *Blender*. Cada uno de sus campos es una colección de un tipo de objeto (*scenes*, *objects*, *meshes*, entre otras).

***bpy.context*:**
Provee el acceso al estado actual de *Blender*: el objeto activo, la escena, la selección actual.

***bpy.ops*:**
Contiene todos los comandos (operadores de *Blender*).

***bpy.types*:**
Contiene las definiciones de todas las clases que son usadas en las estructuras **bpy.data**, ***bpy.context*** y ***bpy.ops***.

***bpy.props*:**
Permite la creación de nuevas propiedades de clases, que *Blender* puede mostrar en los paneles cuando esto sea necesario, para su distinción de las propiedades comunes de las clases (campos), son denominadas propiedades personalizadas de *Blender*.
Para el desarrollo de la solución se utiliza como arquitectura la basada en componentes de la API de *Blender* para *Python*, pues el *add-on* que se implementa se integrará como un módulo que engloba funcionalidades

[1] Un componente es un elemento de *software* que encapsula en unidades, un conjunto de funcionalidades específicas que pueden ser distribuidas e instaladas separadamente de otras (MICROSOFT, 2009).

específicas, a partir de un conjunto de clases, que se diseñan de manera que se garantice la portabilidad y extensibilidad. A continuación se muestra el diseño de las clases que componen la solución.

2.3.2. Diagrama de clases

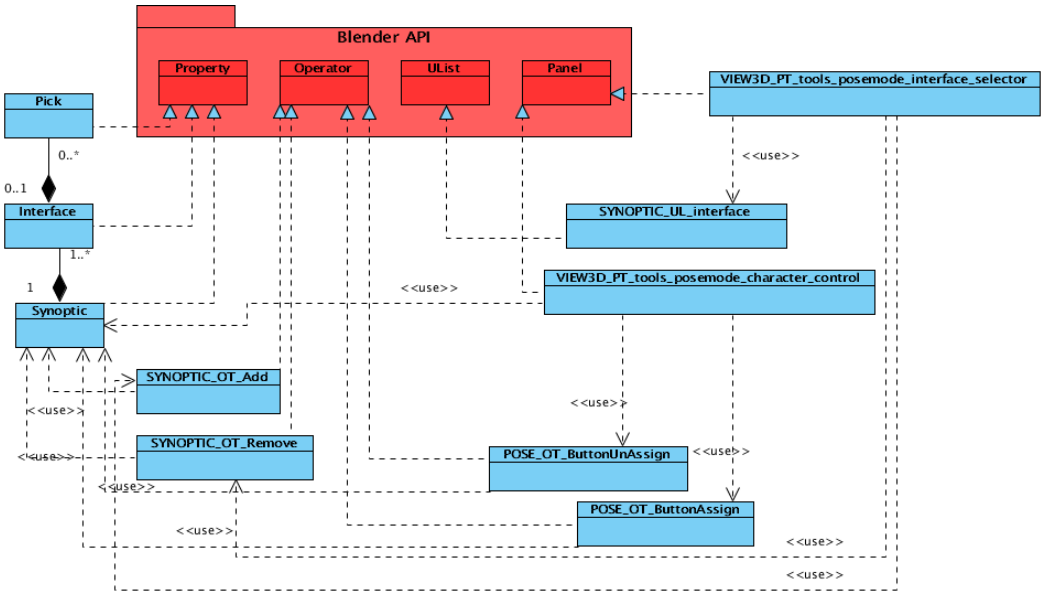


Figura 2.2. Diagrama de clases de la solución.

Las clases que se muestran dentro del paquete "*Blender API*", son clases de la API de *Blender* para *Python* de las cuales heredan el resto de las clases de la solución. Para guiar la creación de la solución propuesta se aplicaron principios generales y estilos, también conocidos como patrones de diseño.

2.3.3. Patrones de diseño

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de *software*, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular (BUSCHMANN; MEUNIER; ROHNERT; SOMMERLAD y & STAL, 1996). Para el diseño de la clases de la aplicación se utilizaron los Patrones de Principios Generales para Asignar Responsabilidades, por sus siglas en inglés (GRASP), que se encargan de la asignación de responsabilidades.

Patrones de diseño GRASP:

Los patrones GRASP describen los principios fundamentales del diseño de objetos y asignación de responsabilidades.

- **Experto:**
  Este se encarga de asignar la responsabilidad de ejecutar una acción, a la clase que contiene la información necesaria para ello. Un ejemplo en la solución, es la clase *"Interface"*, que contiene la información de todos los "*Pick*" que hay dentro de ella, en la que se implementa el método "*selected_picks*', que determina los "*Picks*" seleccionados.
- **Bajo Acoplamiento:**
  Este se encarga de asignar las responsabilidades de manera que el acoplamiento permanezca bajo. Un elemento tiene bajo acoplamiento, cuando no depende de muchos otros. Para el desarrollo de la solución, se relacionan las clases solo de manera imprescindible, con lo que se reduce la dependencia entre estas. Así, al hacer una modificación en alguna de ellas, se reduce el impacto de este en las otras clases. Ejemplo de su uso es la clase es *SYNIPTIC_UL_interface* la cual se relaciona con la clase *UList* a través de la herencia.
- **Alta Cohesión:**
  Enfatiza que las responsabilidades que tiene un elemento, deben estar fuertemente relacionadas a este. Se garantiza con este patrón, que las clases no realicen funcionalidades que no les pertenezcan, lo que facilita la lectura, reutilización y el mantenimiento de las clases. Ejemplo de este, es la clase *Synoptic* en la cual las funcionalidades esta posee están relacionadas con el dibujo y la modificación del objeto *Interface* que esta contiene.

Junto a los patrones, las tarjetas Clase, Responsabilidad, Colaboración (CRC) en XP, son utilizadas también, para la asignación de las responsabilidades y colaboraciones de las clases o elementos del diseño. En el siguiente epígrafe, se hará referencia al uso de estas en el diseño de la solución.

2.3.4. Tarjetas CRC

Las tarjetas CRC son elementos de diseño de XP, que permiten la asignación de las responsabilidades y relaciones, a las clases que serán implementadas durante la etapa de desarrollo. A continuación se muestran las tarjetas CRC generadas para el desarrollo de la solución:

Tabla 2.8. Tarjeta CRC # 1

Tarjeta CRC	
Clase: *VIEW3D_PT_tools_posemode_interface_selector*	
Responsabilidad	**Colaboración**
• Muestra la lista de interfaces. • Muestra el botón de añadir una interfaz a la lista. • Muestra el botón de eliminar un sinóptico de la lista.	*SYNOPTIC_UL_interfaces* *SYNOPTIC_OT_Add* *SYNOPTIC_OT_Remove* Panel

Tabla 2.9. Tarjeta CRC # 2

Tarjeta CRC	
Clase: *SYNOPTIC_UL_interfaces*	
Responsabilidad	**Colaboración**
• Crear la lista de interfaces a mostrar.	*UIList* *VIEW3D_PT_tools_posemode_interface_selector*

Tabla 2.10. Tarjeta CRC # 3

Tarjeta CRC	
Clase: *VIEW3D_PT_tools_posemode_character_control*	
Responsabilidad	**Colaboración**
• Mostrar la interfaz relacionada con el *rig* activo. • Mostrar el botón de asociar un punto de acceso a un control de animación. • Mostrar el botón de eliminar la asociación de punto de acceso a un control de animación.	*POSE_OT_ButtonUnAssign* *POSE_OT_ButtonAssign* Panel

POSE_OT_ButtonUnAssign

Tabla 2.11. Tarjeta CRC # 4

Tarjeta CRC	
Clase:	
Responsabilidad	**Colaboración**
• Eliminar la asociación de punto de acceso a un control de animación.	*Synoptic* *VIEW3D_PT_tools_posemode_character_control* *Operator*

Tabla 2.12. Tarjeta CRC # 5

Tarjeta CRC	
Clase: *POSE_OT_ButtonAssign*	
Responsabilidad	**Colaboración**

Continúa en la próxima página

Tabla 2.12. Continuación de la página anterior

• Asociar un punto de acceso a un control de animación.	*Synoptic* *VIEW3D_PT_tools_posemode_character_control* *Operator*

Tabla 2.13. Tarjeta CRC # 6

Tarjeta CRC	
Clase: *SYNOPTIC_OT_Remov*e	
Responsabilidad	**Colaboración**
• Eliminar la interfaz activa de la lista de interfaces.	Synoptic VIEW3D_PT_tools_posemode_interface_selector Operator

Tabla 2.14. Tarjeta CRC # 7

Tarjeta CRC	
Clase: *SYNOPTIC_OT_Add*	
Responsabilidad	**Colaboración**
• Adicionar una interfaz al sinóptico.	*Synoptic* *VIEW3D_PT_tools_posemode_interface_selector* *Operator*

Tabla 2.15. Tarjeta CRC # 8

Tarjeta CRC	
Clase: *Synoptic*	
Responsabilidad	**Colaboración**
• Dibujar la interfaz. • Transformar la interfaz.	*Property* *VIEW3D_PT_tools_posemode_interface_control* *POSE_OT_ButtonUnAssign* *SYNOPTIC_OT_Add* *SYNOPTIC_OT_Remove* *POSE_OT_ButtonAssign* *POSE_OT_ButtonUnAssign* *Operator*

Tabla 2.16. Tarjeta CRC # 9

Tarjeta CRC	
Clase: *Interface*	
Responsabilidad	**Colaboración**
• Adicionar una interfaz al sinóptico.	*Synoptic* *Pick* *Property*

Tabla 2.17. Tarjeta CRC # 10

Tarjeta CRC	
Clase: *Pick*	
Responsabilidad	**Colaboración**
• Almacenar nombre del control asociado.	*Interface* *Property*

2.4. Conclusiones parciales

- Se formaliza como propuesta de solución la herramienta **CATB** para la selección de controles de animación de personajes 3D, se define un plantilla similar a la armadura *rigify*, su arquitectura se basa en el uso de plantilla como vía para generalizar la propuesta.
- El uso de las reglas y prácticas de la metodología XP, permitió definir los requisitos funcionales y no funcionales del sistema, a partir de las HU escritas por el cliente. Se confeccionaron además, los artefactos que propone XP para la fase de planeación, entre los que se encuentran: el plan de iteraciones, el plan de entregas y las historias de usuarios, haciendo posible la organización del proceso en tres iteraciones y la estimación del tiempo de desarrollo en ocho semanas aproximadamente.
- El estudio de la arquitectura de la API de *Blender* para *Python*, posibilitó la definición de la arquitectura de la solución como basada en **componentes**, para asegurar su integración correcta a *Blender*.
- El empleo de los patrones de diseño **GRASP** (Experto, Bajo acoplamiento y Alta Cohesión), permitió determinar de forma clara y precisa los principios fundamentales para la definición de los objetos y asignación de responsabilidades.
- El diagrama de clases obtenido durante el diseño proporciona la definición de las clases y sus responsabilidades, mediante el uso de las tarjetas CRC, con lo que se crean las bases necesarias para la etapa de implementación.

CAPÍTULO 3

Implementación y Prueba del sistema

Introducción

En el presente capítulo se define el comportamiento del sistema y se definen los estándares de codificación a seguir durante la implementación de sus funcionalidades. Se describe el proceso de verificación y validación desarrollado para evaluar la propuesta de solución. Se verifica la calidad del resultado de la implementación, mediante el diseño y ejecución de las pruebas, exponiendo los resultados obtenidos.

3.1. Implementación

En XP, esta etapa es donde se codifican las HU. La implementación de estas, debe hacerse ateniendo a estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente, facilita su comprensión y escalabilidad (DAYANA BUSTAMANTE, 2014).

3.1.1. Estándares de codificación

La metodología XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible. Un estándar de codificación "es un conjunto de reglas, normas o patrones destinados a establecer uniformidad en el proceso de generación de un código. Son pautas de programación enfocadas a la estructura y apariencia física del código para facilitar su lectura, comprensión y mantenimiento" (OSORIA, 2013).

En la implementación de la herramienta se utilizaron los estándares definidos por la API de *Blender* para *Python* basados en el estándar PEP8 de *Python*, con el objetivo de evitar la mezcla de diferente estilos de codificación y hacer más fácil su reutilización en otras herramientas. A continuación se detallan los mismos:

- Los nombres de los módulos se escriben en minúsculas y se subrayan separado.
- Se utiliza una indentación de cuatro espacios con lo que se sustituyen las tabulaciones.
- Se utiliza un espacio alrededor de los operadores.

- Se utiliza únicamente las importaciones explícitas.
- Se usa comillas simples para las numeraciones y comillas dobles para las cadenas.

En el Código Fuente 3.1 se presenta un fragmento de código de la clase *Synoptic*. En ella se ejemplifica el uso de los estándares de codificación utilizados en la implementación de la solución.

Código fuente 3.1. selected picks

```
def selected_picks(self, context):
    """The state of prop change
    """
    for pick in self.picks:
        if pick.bone_name not in {""}:
            print(pick.bone_name)
            if context.object.pose.bones[pick.bone_name].bone.select:
                yield pick
        else:
            if pick.select:
                yield pick
```

Los estándares de codificación utilizados en la investigación permiten tener un estilo único de codificación para facilitar el estudio, entendimiento del código de la aplicación y garantizar que este sea más fácil de mantener.

3.1.2. Iteraciones

En XP, esta es la fase fundamental del ciclo de desarrollo, donde son desarrolladas las funcionalidades que implementan las HU. Para su implementación, estas se desagregan en tareas de ingeniería a desarrollar por el equipo de desarrollo (JOSKOWICZ, 2008).

Tareas de ingeniería

Las tareas de ingeniería permiten la asignación de las funcionalidades a los miembros del equipo de desarrollo, para implementar las HU definidas por el cliente. Estas provienen de las HU y especifican el tiempo, el tipo, sus responsables y una descripción acorde al lenguaje usado por el equipo de desarrollo. Con el objetivo de implementar exitosamente los requisitos definidos en las HU, se definen las siguientes tareas (CANÓS; LETELIER y PENADÉS, 2003) en cada iteración.

Tareas de ingeniería asignadas a la iteración I

Para la primera iteración, se asignaron tres tareas de ingeniería con las funcionalidades de mayor prioridad para el cliente. Estas establecen el funcionamiento básico de la herramienta a desarrollar:

Tabla 3.1. Tarea de ingeniería # 1

Tarea	
Número de tarea: 1	**Número de Historia de usuario:** 1
Nombre de la tarea: Seleccionar control a través de un punto de acceso.	

Continúa en la próxima página

Tabla 3.1. Continuación de la página anterior

Tipo de tarea: Implementación	**Puntos estimados:** 1.9
Fecha de inicio: 11 de febrero de 2016	**Fecha de fin:** 15 de febrero de 2016
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: Se debe permitir la selección por parte del usuario del control deseado, desde una interfaz del sinóptico, a través de un punto de acceso asociado a este.	

Tabla 3.2. Tarea de ingeniería # 2

Tarea	
Número de tarea: 2	**Número de Historia de usuario:** 2
Nombre de la tarea: Asociar punto de acceso a un control de animación.	
Tipo de tarea: Implementación	**Puntos estimados:** 0.9
Fecha de inicio: 1 de febrero de 2016	**Fecha de fin:** 5 de febrero de 2016
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: Se debe crear una relación entre un punto de acceso y un control, que puede ser a través del nombre del control, lo que permite que el usuario establezca que puntos de acceso se asocian a un determinado control.	

Tabla 3.3. Tarea de ingeniería # 3

Tarea	
Número de tarea: 3	**Número de Historia de usuario:** 3
Nombre de la tarea: Eliminar asociación de los puntos de acceso a un control de animación.	
Tipo de tarea: Implementación	**Puntos estimados:** 0.2
Fecha de inicio: 1 de febrero de 2016	**Fecha de fin:** 5 de febrero de 2016
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: Se debe eliminar la relación existente entre un punto de acceso y un control, que puede ser a través de la modificación del atributo del punto de acceso que hace referencia al control.	

Tareas de ingeniería asignadas a la iteración II

En la segunda iteración se implementaron las tareas de ingeniería que permiten al usuario mayor flexibilidad en el uso de la herramienta:

Tabla 3.4. Tarea de ingeniería # 4

Tarea	
Número de tarea: 4	**Número de Historia de usuario:** 4
Nombre de la tarea: Asignar sinóptico al *rig* personaje.	
Tipo de tarea: Implementación	**Puntos estimados:** 0.2
Fecha de inicio: 1 de febrero de 2016	**Fecha de fin:** 5 de febrero de 2016
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: Se implementa de forma que al cargarse un *rig* del personaje, se crea una clase sinóptico, la que es agregada como propiedad al *rig*.	

Tabla 3.5. Tarea de ingeniería # 5

Tarea	
Número de tarea: 5	**Número de Historia de usuario:** 4
Nombre de la tarea: Añadir interfaz a un sinóptico.	
Tipo de tarea: Implementación	**Puntos estimados:** 0.7
Fecha de inicio: 4 de febrero de 2016	**Fecha de fin:** 10 de febrero de 2016
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: Se selecciona por el usuario, una de los tipos de interfaces predefinidas por la herramienta, la cual es añadida al listado de interfaces que posee el sinóptico del personaje.	

Tabla 3.6. Tarea de ingeniería # 6

Tarea	
Número de tarea: 6	**Número de Historia de usuario:** 6
Nombre de la tarea: Eliminar interfaz de sinóptico.	
Tipo de tarea: Implementación	**Puntos estimados:** 0.3
Fecha de inicio: 11 de febrero de 2016	**Fecha de fin:** 15 de febrero de 2016
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: Se debe quitar la interfaz del panel de las interfaces asociadas al sinóptico.	

Tareas de ingeniería asignadas a la iteración III

En una tercera iteración se implementaron las tareas de ingeniería que complementan las funcionalidades desarrolladas en la segunda iteración:

Tabla 3.7. Tarea de ingeniería # 7

Tarea	
Número de tarea: 7	**Número de Historia de usuario:** 5
Nombre de la tarea: Listar interfaces del sinóptico.	
Tipo de tarea: Implementación	**Puntos estimados:** 1.3
Fecha de inicio: 11 de febrero de 2016	**Fecha de fin:** 15 de febrero de 2016
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: Se debe mostrar en un panel las diferentes interfaces que posee el sinóptico.	

Tabla 3.8. Tarea de ingeniería # 8

Tarea	
Número de tarea: 8	**Número de Historia de usuario:** 2
Nombre de la tarea: Modificar el comportamiento del movimiento de los huesos.	
Tipo de tarea: Implementación	**Puntos estimados:** 1.9
Fecha de inicio: 11 de febrero de 2016	**Fecha de fin:** 15 de febrero de 2016
Programador responsable: Oscar Carmenate Rodríguez	
Descripción: .	

3.2. Integración con Blender

La herramienta se integra a *Blender* como un *add-on*, se puede instalar al ir a:

1. *File, User Preferences, Add-ons, Install from file.*
2. Se busca donde se encuentra el archivo de la aplicación y se instala.
3. Se busca el *add-on* en la categoría *"Rigging"*.

Para activarlo, se selecciona el *checkbox*, como se muestra en la Figura 3.1.

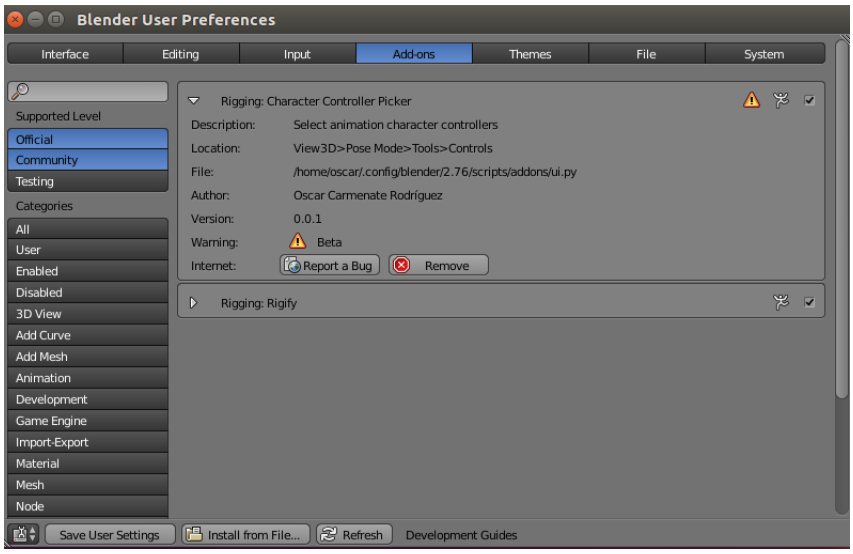


Figura 3.1. Panel principal de la herramienta

Una vez instalado el *add-on* este puede ser utilizado cuando se encuentra un *rig* en el visor 3D, en modo "Pose".

3.2.1. Interfaz gráfica de la herramienta

Después de haber instalado el *add-on*, este es accesible desde el panel de herramientas, que se encuentra en el panel lateral. A continuación se muestra una imagen con una breve descripción:

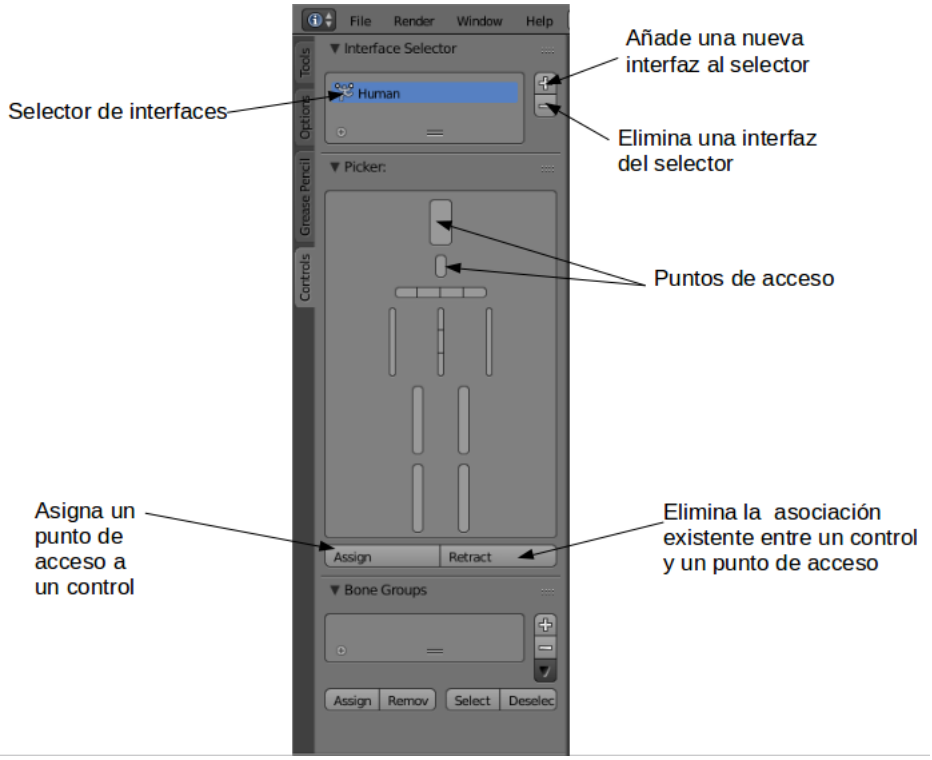


Figura 3.2. Vista principal de la herramienta

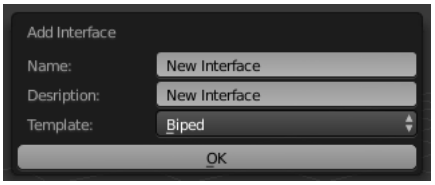


Figura 3.3. Adicionar Interfaz

En la Figura 3.3 se observa el cuadro de diálogo que se muestra al seleccionar la opción Agregar nueva interfaz, para ello se define su nombre, descripción y tipo.

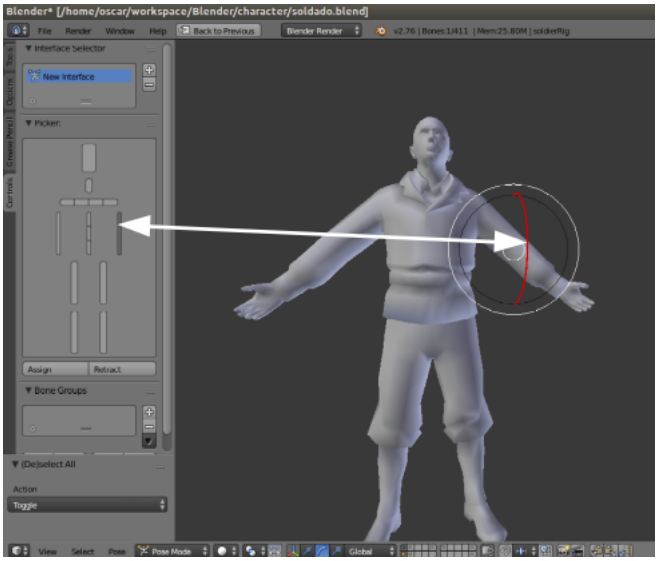


Figura 3.4. Asociación de controles del personaje a puntos de acceso.

En la Figura 3.4 el punto de acceso asociado al control se muestra en color gris más intenso, para facilitar al animador la identificación de este y por consiguiente del control asociado.

3.3. Pruebas

El proceso de pruebas de *software* consiste en comprobar que lo que se ha especificado he implementado es lo que el usuario realmente desea. Las pruebas añaden valor al producto obtenido, por lo que permiten verificar y revelar la calidad de un producto de *software* (PRESSMAN, 2010). En el desarrollo de la propuesta de solución se ejecutaron Pruebas de aceptación.

3.3.1. Pruebas de aceptación

Las pruebas de aceptación son pruebas funcionales que se realizan con el cliente basándose en los requerimientos tomados de las HU. En todas las iteraciones, cada una de las HU seleccionadas por el cliente deberá tener una o más pruebas de aceptación, se deberán determinar los casos de prueba e identificar los errores que serán corregidos ECHEVERRY TOBÓN y DELGADO CARMONA, 2007. A continuación se muestran los casos de pruebas de aceptación ejecutados en cada iteración:

Pruebas de aceptación para la iteración I

Tabla 3.9. Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Seleccionar control a través de un punto de acceso.	
Descripción: El usuario tendrá la posibilidad de seleccionar un control del personaje, a través de un punto de acceso, para su manejo sin tener que recurrir a este directamente.	

Continúa en la próxima página

Tabla 3.9. Continuación de la página anterior

Condiciones de ejecución: • El add-on debe estar instalado. • El add-on debe estar activo. • Debe existir un *rig* en la escena. • El *rig* debe estar en modo "Pose". • El punto de acceso debe estar asociado a un control. • El control asociado al punto de acceso debe estar deseleccionado.
Pasos de ejecución: 1. El usuario selecciona el punto de acceso, el cual cambia de color, al ser seleccionado. 2. En el visor 3D el control muestra los manipuladores.
Resultados esperados: El usuario se selecciona el control correctamente.

Tabla 3.10. Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU1_P2	**Historia de usuario:** 1
Nombre: Deseleccionar control a través de un punto de acceso.	
Descripción: El usuario tendrá la posibilidad de deseleccionar un control del personaje, a través de un punto de acceso, para su manejo sin tener que recurrir a este directamente.	
Condiciones de ejecución: • El add-on debe estar instalado. • El add-on debe estar activo. • Debe existir un *rig* en la escena. • El *rig* debe estar en modo "Pose". • El punto de acceso debe estar asociado a un control. • El control asociado al punto de acceso debe estar seleccionado.	
Pasos de ejecución: 1. El usuario selecciona el punto de acceso, el cual cambia de color, al ser deseleccionado. 2. En el visor 3D se ocultan los manipuladores.	
Resultados esperados: El usuario se selecciona el control correctamente.	

Pruebas de aceptación para la iteración II

Tabla 3.11. Prueba de aceptación # 3

Caso de prueba de aceptación	
Código: HU2_P1	**Historia de usuario:** 2
Nombre: Deseleccionar control a través de un punto de acceso.	
Descripción: Asocia los puntos de acceso de un sinóptico, a un control de animación del personaje.	

Continúa en la próxima página

Tabla 3.11. Continuación de la página anterior

Condiciones de ejecución: • El add-on debe estar instalado. • El add-on debe estar activo. • Debe existir un *rig* en la escena. • El *rig* debe estar en modo "Pose". • Debe haber un control de animación activo. • Debe haber al menos un punto de acceso seleccionado.
Pasos de ejecución: 1. El usuario hace *click* sobre el botón *"Assign"*. 2. Se muestra un mensaje en la parte superior de la ventana de *Blender*, que indica que la operación fue realizada satisfactoriamente.
Resultados esperados: Satisfactorio.

Tabla 3.12. Prueba de aceptación # 4

Caso de prueba de aceptación	
Código: HU2_P2	**Historia de usuario:** 2
Nombre: Deseleccionar control a través de un punto de acceso.	
Descripción: Desvincular un punto de acceso de un sinóptico, de un control de animación del personaje.	
Condiciones de ejecución: • El add-on debe estar instalado. • El add-on debe estar activo. • Debe existir un *rig* en la escena. • El *rig* debe estar en modo "Pose". • Debe haber un control de animación activo. • Debe haber al menos un punto de acceso seleccionado.	
Pasos de ejecución: 1. El usuario hace *click* sobre el botón *"Retract"*. 2. Se muestra un mensaje en la parte superior de la ventana de *Blender*, que indica que la operación fue realizada satisfactoriamente.	
Resultados esperados: Satisfactorio.	

Pruebas de aceptación para la iteración III

Tabla 3.13. Prueba de aceptación # 5

Caso de prueba de aceptación	
Código: HU3_P1	**Historia de usuario:** 3
Nombre: The test case name	
Descripción: Añadir interfaces del sinóptico.	
Condiciones de ejecución: • El add-on debe estar instalado. • El add-on debe estar activo. • Debe existir un *rig* en la escena. • El *rig* debe estar en modo "Pose".	

Continúa en la próxima página

Tabla 3.13. Continuación de la página anterior

Pasos de ejecución: 1. El usuario hace *click* sobre el botón añadir (+), el cual se encuentra en la esquina superior derecha del selector de interfaces. 2. Se muestra un cuadro de diálogo. 3. El usuario teclea el nombre, la descripción, y selecciona un tipo de interfaz. 4. El usuario hace *click* en el botón "OK". 5. Se muestra en el selector de la interfaces, el nombre de la nueva interfaz creada. 6. Se muestra un mensaje en la parte superior de ventana de *Blender* que indica que la operación se realizó de manera exitosa.
Resultados esperados: Satisfactorio.

Tabla 3.14. Prueba de aceptación # 6

Caso de prueba de aceptación	
Código: HU3_P2	**Historia de usuario:** 3
Nombre: The test case name	
Descripción: Muestra en una lista las interfaces asociada al sinóptico.	
Condiciones de ejecución: • El add-on debe estar instalado. • El add-on debe estar activo. • Debe existir un *rig* en la escena. • El *rig* debe estar en modo "Pose". • Debe haber una interfaz seleccionada en el selector de interfaces.	
Pasos de ejecución: 1. El usuario hace *click* sobre el botón eliminar (-), el cual se encuentra en la esquina superior derecha del selector de interfaces. 2. Se muestra un cuadro de diálogo preguntando al usuario si se desea eliminar la interfaz. 3. El usuario hace *click* en el botón "OK". 4. Se elimina del selector de interfaces la interfaz seleccionada. 5. Se muestra un mensaje en la parte superior de ventana de *Blender* que indica que la operación se realizó de manera exitosa.	
Resultados esperados: Satisfactorio.	

Tabla 3.15. Prueba de aceptación # 7

Caso de prueba de aceptación	
Código: HU4_P1	**Historia de usuario:** 4
Nombre: The test case name.	
Descripción: Muestra en una lista las interfaces asociada al sinóptico.	
Condiciones de ejecución: • El add-on debe estar instalado. • El add-on debe estar activo. • Debe existir un *rig* en la escena. • El *rig* debe estar en modo "Pose". • Debe existir al menos una interfaz en el selector de interfaces.	

Continúa en la próxima página

Tabla 3.15. Continuación de la página anterior

Pasos de ejecución: Listar interfaces del sinóptico.
Resultados esperados: Satisfactorio.

Análisis de pruebas de aceptación

Las pruebas fueron realizadas por los animadores del centro VERTEX. Se realizaron tres iteraciones, en la primera se encontraron siete no conformidades de funcionalidad, en la segunda iteración se encontraron cuatro no conformidades de funcionalidad y en la tercera iteración no se encontraron no conformidades. Todas estas fueron resueltas oportunamente. En la Figura 3.5 se presenta un gráfico con los resultados de las pruebas realizadas.

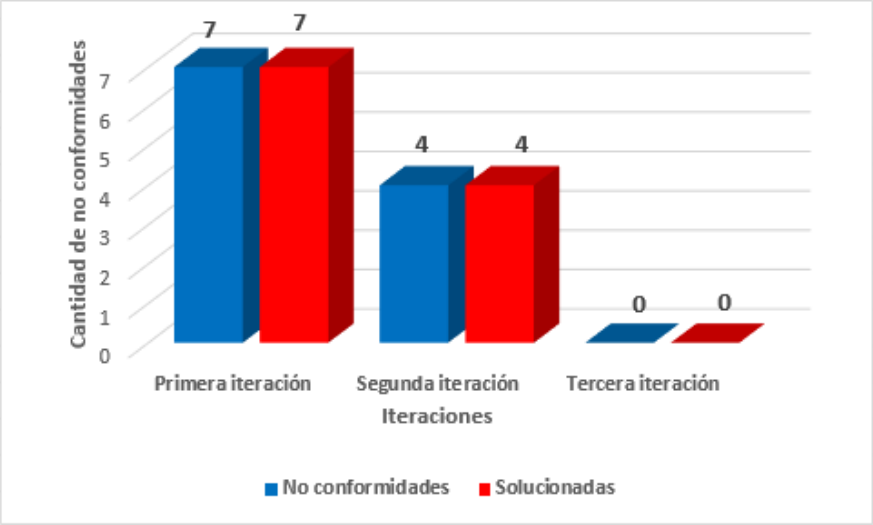


Figura 3.5. Descripción general del caso de prueba para el requisito funcional Adicionar sinóptico.

Una vez probadas todas las funcionalidades de la herramienta se procede a la definición de un escenario de animación, para establecer una comparación del proceso de selección de controles en *Blender* con la herramienta CATB y sin ella.

3.4. Análisis de resultados

Primeramente se tiene en cuenta que la selección de los controles, según los animadores, es compleja, debido a que cuando se trata de seleccionar un determinado control, puede seleccionarse otro no deseado, lo que consume tiempo de trabajo de los animadores, al tener que intentar sucesivamente escoger un control. Para un escenario donde se quiera animar a un personaje haciendo movimiento oscilatorio alternando sus brazos, que requiere la selección y uso de los controles de los brazos, de la forma tradicional, pudiera realizarse de esta forma:

1. Seleccionar el control asociado al brazo izquierdo.
2. Entrar a la vista *Only Render* para poder ver claramente la transformación de la malla del personaje.
3. Rotar el brazo izquierdo hacia delante.

4. Salir de la vista *Only Render*.
5. Seleccionar el control asociado al brazo derecho.
6. Entrar a la vista *Only Render* para poder ver claramente la transformación de la malla del personaje.
7. Rotar el brazo derecho hacia delante.
8. Salir de la vista *Only Render*.
9. Seleccionar el control asociado al brazo izquierdo.

Así, sucesivamente, hasta lograr la animación. En la Figura 3.6 se ilustra cómo se realiza la selección de controles sobre el personaje, donde se puede apreciar la complejidad que esto conlleva.

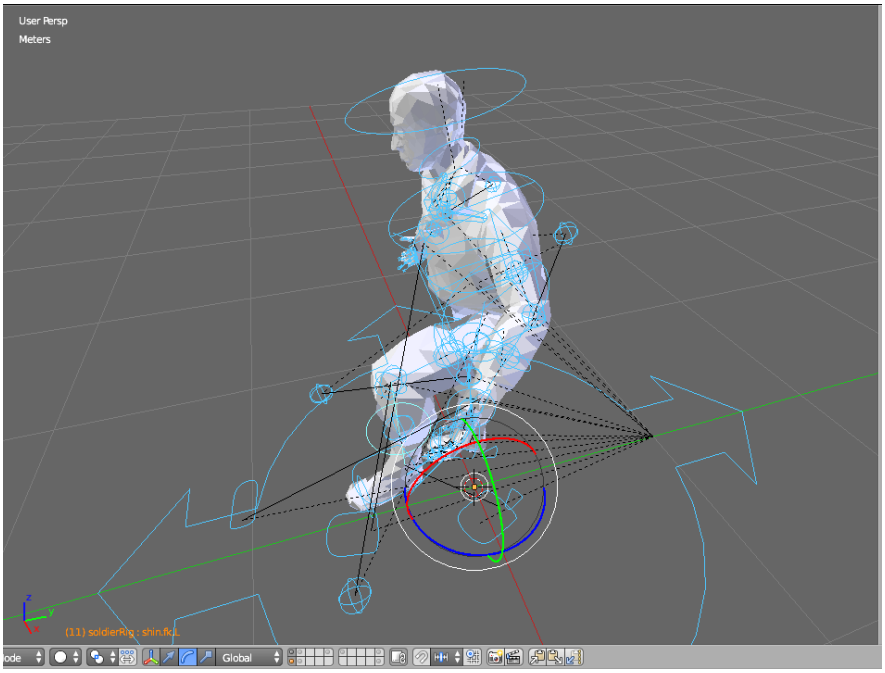


Figura 3.6. Selección de controles de animación de personaje de forma tradicional.

Con el uso de la herramienta CATB, esta animación pudiera realizarse así:

1. Seleccionar el control asociado al brazo izquierdo.
2. Seleccionar el punto de acceso a asociar.
3. Asociar el punto de acceso al control.
4. Seleccionar el control asociado al brazo derecho.
5. Seleccionar el punto de acceso a asociar.
6. Asociar el punto de acceso al control.
7. Entrar a la vista *Only Render* para poder ver claramente la transformación de la malla del personaje.
8. Seleccionar el punto de acceso asociado al brazo derecho.
9. Rotar el brazo derecho hacia delante.
10. Seleccionar el punto de acceso asociado al brazo izquierdo.
11. Rotar el brazo izquierdo hacia delante.
12. Seleccionar el punto de acceso asociado al brazo derecho.
13. Rotar el brazo derecho hacia delante.
14. Seleccionar el punto de acceso asociado al brazo izquierdo.

15. Rotar el brazo izquierdo hacia delante.

Como se puede apreciar en el escenario antes expuesto, mediante el uso de la forma tradicional de selección de los controles, se deben escoger los controles varias veces sobre el personaje, lo que conlleva la dificultad, dependiendo de la complejidad del personaje, de no siempre seleccionar el control deseado. Esta tarea requiere salir constantemente de la vista *"Only Render"*, lo que a su vez hace consumir mucho tiempo a los animadores.

Mientras que, con el uso de la herramienta CATB, la selección de los controles se realiza una sola vez sobre el personaje, para asociarlos a los puntos de acceso de la interfaz del sinóptico. Posteriormente no se requiere salir de la vista *"Only Render"* para la selección de otros controles. Esto permite el manejo de los controles, a partir estos puntos de acceso, sin ser mostrados directamente sobre el personaje, con lo que se evitan los problemas que conllevaba la forma tradicional.

Se puede afirmar que la herramienta desarrollada facilita la selección de los controles de los personajes, al realizar la selección sobre el personaje una sola vez, y permitir que se accedan estos mediante puntos de acceso, sin que se muestren directamente sobre el personaje.

3.5. Conclusiones parciales

- El uso de las tareas de ingeniería que define la metodología XP permitió organizar la implementación de los requisitos definidos en las HU y el desarrollo de la herramienta.
- Se logra la integración con *Blender* a partir del uso de las clases del API de *Blender* para *Python* en la implementación de la solución.
- Las pruebas de aceptación ejecutadas permitieron verificar el cumplimiento de los requerimientos del cliente, así como la detección de no conformidades, que fueron corregidas oportunamente.
- A partir de un escenario de animación se muestran las ventajas que ofrece la herramienta para la selección de controles respecto a la forma tradicional utilizada, demostrando como esta resuelve la problemática planteada al desarrollarse una herramienta que permite la selección de controles de animación de personaje 3D, sin la visualización de los mismos sobre el personaje, esta puede ser asociada a cualquier *rig* de un personaje 3D y posibilita el almacenamiento, en el proyecto de *Blender*, de la asociación existente entre los puntos de acceso y los controles de un personaje, para un posterior uso.
- En las plantillas creadas para esta primera versión se encuentran limitados los puntos de acceso. No se muestran todos los posibles puntos de acceso que se pueden asociar a los controles de un personaje, pues estos se basaron en los que posee la armadura *rigify*.

Conclusiones

1. La herramienta CATB brinda al animador una nueva forma de selección de los controles de animación de personajes 3D. Esta permite realizar la selección de controles sin la necesidad de mostrarlos sobre el personaje, lo que facilita el proceso de animación.
2. Se define la arquitectura basada en *template*, como vía para generalizar la propuesta. Se emplea un *template* similar a la armadura *rigify*.
3. La metodología de desarrollo de software XP, permitió especificar los elementos necesarios para la implementación de la herramienta.
4. Se realizaron pruebas de aceptación, como resultado luego de tres iteraciones no se encontraron no conformidades respecto a las funcionalidades implementadas en la solución.
5. Se compara el método tradicional empleado en *Blender* para la selección de los controles de los personajes con la solución propuesta, evidenciando las facilidades que incorpora CATB al proceso de animación.

Para el perfeccionamiento de la herramienta se recomienda:

- Hacer un seguimiento de la estabilidad del módulo *bgl* de la API de *Blender* para *Python*, y valorar su uso en la creación de interfaces en el sinóptico.
- Implementar otros tipos de interfaces de acuerdo a las necesidades que presenten los animadores del centro VERTEX.
- Desarrollar un editor de interfaces que permita crear interfaces personalizadas.

Glosario

add-on Componente de software que le agrega una funcionalidad específica a un programa ya existente.. 10, 16, 17

huesos o articulaciones Los huesos o articulaciones en la animación 3D no lucen como los huesos humanos reales. En ves de eso, estos tienen formas alambres las cuales representan el sistema esquelético. 2

metaballs s el nombre de una técnica de gráficos realizada por ordenador para simular interacción orgánica entre diferentes objetos n-dimensionales.. 10

Acrónimos

2D dos dimensiones. 1
3D tres dimensiones. 1–6, 8–10, 12, 14, 15, 17, 21, 22, 33, 45

API Interfaz de programación de aplicaciones, por sus siglas en inglés. 16–18, 21, 28, 33, 34, 44, 46

CAD Diseño Asistido por Ordenador, por sus siglas en inglés. 2
CASE Ingeniería de Software Asistida por Computadoras, por sus siglas en inglés. 20
CRC Clase, Responsabilidad, Colaboración. 30, 33

GRASP Patrones de Principios Generales para Asignar Responsabilidades, por sus siglas en inglés. 29

HU historias de usuario. 24, 26, 27, 33

ICAIC Instituto Cubano del Arte e Industria Cinematográficos. 1
IDE Entorno de Desarrollo Integrado, por sus siglas en inglés. 3, 17, 18, 21

MSF Microsoft Solution Framework. 18

NURBS B-splines racionales no uniformes. 10

PCC Partido Comunista de Cuba. 1

RUP Rational Unified Process. 18

SO Sistemas Operativos. 10

UCI Universidad de las Ciencias Informáticas. 2, 19
UML Lenguaje de Modelado Universal, por sus siglas en inglés. 20

VERTEX Centro de Entornos Interactivos 3D. 2, 3, 5, 15, 18, 19, 21, 46

XP Programación Extrema, por sus siglas en inglés. 19–21, 24, 26, 27, 33, 35, 45

Referencias bibliográficas

BEANE, Andy. 2012. *3D Animation Essentials*. 2012.

BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P. y & STAL, M. 1996. *Pattern - Oriented Software Architecture. A System of Patterns.* Inglaterra : John Wiley & Sons, 1996.

CANÓS, José H; LETELIER, Patricio y PENADÉS, Mª Carmen. 2003. Metodologías ágiles en el desarrollo de software. *Universidad Politécnica de Valencia, Valencia*. 2003.

CUBADEBATE. 2014. *Cuba estrenó este domingo Meñique, su primera película de animados en 3D (+ Video)*. 2014. Dirección: ⟨http://www.cubadebate.cu/noticias/2014/07/20/cuba-estrena-este-domingo-a-menique-su-primera-pelicula-de-animados-en-3d/⟩.

CUBADEBATE. 2015. *Estudios de animación del ICAIC presentarán tres videojuegos este verano*. 2015. Dirección: ⟨http://www.cubadebate.cu/noticias/2015/06/17/estudios-de-animacion-del-icaic-presentaran-tres-videojuegos-este-verano/⟩.

DAYANA BUSTAMANTE, Jean C. Rodríguez. 2014. *Metodología Actual: Metodología XP*. 2014.

DOMINGO, I.; RIUS, G. y L., Cuenca. 2012. Una revisión sobre el estado del arte en herramientas de modelado basado en UML. *6th International Conference on Industrial Engineering and Industrial Management. VI Congreso de Ingeniería de Organización, Vigo*. 2012.

ECHEVERRY TOBÓN, LUIS MIGUEL y DELGADO CARMONA, LUZ ELENA. 2007. *CASO PRÁCTICO DE LA METODOLOGÍA ÁGIL XP AL DESARROLLO DE SOFTWARE*. 2007.

ECLIPSE. *Eclipse IDE*. Dirección: ⟨https://eclipse.org/ide/⟩.

ESPAÑOLA, Real Academia. *Dicionario de la lengua española*. Dirección: ⟨http://dle.rae.es⟩.

FERNÁNDEZ BAENA, Adso. 2015. *Animation and Interaction of Responsive, Expressive, and Tangible 3D Virtual Characters*. 2015.

FOUNDATION, Blender. *Blender/Python API*. Dirección: ⟨https://www.blender.org/api/blender_python_api_2_76b_release/⟩.

GARCÍA DOMINGUEZ, Dailyn y GÓMEZ BARROSO, Clariannis. 2009. *Incorporación de comportamientos a elementos que se mueven sobre grafos de camino*. 2009.

GUIBERT ESTRADA L.and Altuna Castillo, Enrique José. 2011. Ingeniería de requisitos del software educativo "Mis Mejores Cuentos". *Centro de Tecnologías para la Formación. Universidad de las Ciencias Informáticas*. 2011. Url: ⟨http://gte2.uib.es/edutec/sites/default/files/congresos/edutec11/Ponencias/Mesa%205/Guibert%20-%20Altuna-%20Edutec%202011.F.pdf.⟩.

JAWORSKI, Witold. 2011. *Programming Add-Ons for Blender 2.5: Writing Python Scripts with Eclipse IDE*. 2011.

JETBRAINS. *PyCharm*. Dirección: ⟨http://www.jetbrains.com/pycharm/⟩.

JOSKOWICZ, José. 2008. *Reglas y Prácticas en eXtreme Programming*. 2008.

KENDALL, K. y KENDALL, Julie. 2005. *Análisis y Diseño de Sistemas*. 6th. México : Pearson, 2005.

KNIBERG, Henrik. 2007. *Scrum y XP desde las trincheras: Como hacemos Scrum*. 2007. ISBN: 978-1-4303-2264-1. Url: ⟨http://infoq.com/minibooks/scrum-xp-fromthetrenches⟩.

LASSETER, John. 2001. Tricks to animating characters with a computer. *ACM Siggraph Computer Graphics*. 2001, vol. 35, n.º 2, págs. 45-47. Url: ⟨http://dl.acm.org/citation.cfm?id=563706⟩.

LÓPEZ SANTADER, Daniel y CHAHIN GARCÍA, Etzedine. 2014. *Sistema de aprendizaje de Blender*. 2014. Universidad Veracruzana. Región Poza Rica-Tuxpan.⟨http://cdigital.uv.mx/handle/123456789/40330⟩.

LOW, Ser En. 2014. *Sketch-Based Animation Tool for Character Animation Intergrating into a Production Pipeline*. 2014. Url: ⟨http://oaktrust.library.tamu.edu/handle/1969.1/152733⟩.

MA, Wan-Chun Alex; RHEE, Taehyun y YOSHIYASU, Yusuke. 2015. Making Digital Characters: Creation, Deformation, and Animation. En. *SIGGRAPH Asia 2015 Courses*. New York, NY, USA : ACM, 2015, págs. 11:1-11:79. SA '15. Url: ⟨http://doi.acm.org/10.1145/2818143.2818172⟩.

MARTÍNEZ, Jon Latorre. 2004. 2004. Dirección: ⟨http://etxea.net/docu/taller_bash/bash.html⟩.

MCLAUGHLIN, Tim; CUTLER, Larry y COLEMAN, David. 2011. Character rigging, deformations, and simulations in film and game production. En. *ACM SIGGRAPH 2011 Courses*. N.º 5, 2011. Url: ⟨http://dl.acm.org/citation.cfm?id=2037641⟩.

MICROSOFT. 2009. *Microsoft Application Architecture Guide*. 2nd. 2009. 9780735627109. Url: ⟨https://www.fizyka.umk.pl/~jacek/docs/net/Application_Architecture_Guide_v2.pdf⟩.

NICADO GARCÍA, Miriam. 2014. *Resolución Migración SWL*. 2014. Url: ⟨http://intranet2.uci.cu/servicios/bases-legales/vista⟩.

OSORIA, Dayamí Palma. 2013. *Módulo para la extracción y representación de los metadatos en el Sistema de Gestión Documental de audio y video digitales TeVeo Plus V1*. 2013. Url: ⟨http://repositorio_institucional.uci.cu/jspui/bitstream/ident/8302/1/TD_06469_13.pdf⟩.

PCC. 2011. *Lineamientos de la política económica y social del partido y la revolución*. 2011.

PÉREZ RAMÍREZ, Danay; OLIVEROS GUNTÍN, Yoanna; ALVAREZ ALONSO, Yanniel y COELLO MENA, Jorge. 2008. METODOLOGÍAS ÁGILES .CÓMO DESARROLLO UTILIZANDO XP? *Convención Científica de Ingeniería y Arquitectura*. 2008.

PONS, C.; GIANDINI, Roxana. y PÉREZ, Gabriela. 2010. *Desarrollo de software dirigido por modelos. Conceptos teóricos y su aplicación práctica*. 1ra. 2010.

PRESSMAN, R.S. 2010. *Software engineering: a practitioner's approach*. 7th. McGraw-Hil. New York, EUA : McGrawHill, 2010. ISBN:978-0-07-337597.

PYDEV. *Python IDE for Eclipse*. Dirección: ⟨http://www.pydev.org/⟩.

PYTHON. 2016. 2016. Dirección: ⟨https://wiki.python.org/moin/PythonEditors⟩.

ROMERO SANTILLÁN, Paco Andrés. 2014. *GUÍA METODOLÓGICA DE MODELADO Y ANIMACIÓN 3D PARA MUNDOS VIRTUALES INTERACTIVOS*. 2014.

RUMBAUGH, James. 2000. *El lenguaje unificado de modelado : manual de referencia*. Madrid, España : Addison Wesley, 2000.

SOMMERVILLE, I. 2011. *Ingeniería de software*. España : Pearson, 2011. ISBN: 9786073206044.

TORRE LLORENTE; UNAI ZORRILLA CASTRO; MIGUEL ANGEL RAMOS BARROS; JAVIER CALVARRO NELSON, César de la. 2010. *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0*. 2010.

WHITE, Tony. 2009. *How to make animated films : Tony White' s complete masterclass on the traditional principles of animation*. 2009.

WILLIAMS, Richard. 2001. *The Animator's Survival Kit: A Manual of Methods, Principles, and Formulas for Classical, Computer, Games, Stop Motion, and Internet Animators*. 2001. ISBN 0-5712-0228-4.

Generado con LaTeX: 6 de julio de 2016: 10:54am