Universidad de las Ciencias Informáticas
Facultad 5

Título: “Diseño de un sistema de animación facial muscular para el software de entrenamiento Indicios Aduaneros”

Trabajo de Diploma para optar por el Título de
Ingeniero en Ciencias Informáticas

Autora: Mariela Milagros Bony Fernández

Tutor: Ing. Jaime González Campistruz
Co-Tutora: Ing. Lien Muguercia Torres

Ciudad de La Habana, Junio de 2009
“Año del 50 Aniversario del Triunfo de la Revolución”

Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa.

Mahatma Gandhi

Datos de Contacto

Nombre y Apellidos: Ing. Jaime González Campistruz
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Ciencias Informáticas
e-mail: jgonzalezc@uci.cu

Ingeniero en Ciencias Informáticas, graduado en la Universidad de Ciencias Informáticas (UCI) en el 2007. Profesor de la UCI en adiestramiento, con 2 años de experiencia en su desempeño laboral.

Nombre y Apellidos: Ing. Lien Muguercia Torres
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Ciencias Informáticas
e-mail: lmuguercia@uci.cu

Ingeniero en Ciencias Informáticas, graduada en la Universidad de Ciencias Informáticas (UCI) en el 2007. Profesor de la UCI en adiestramiento, con 2 años de experiencia en su desempeño laboral.

Agradecimientos

Incontables son las personas que han contribuido a la realización de esta investigación, no solo aportando conocimientos, sino brindándome apoyo, cariño y mucho amor.

Las primeras líneas para el dúo más importante en mi vida, mis padres.
A ti mami, por el apoyo incondicional desde el primer momento en que decidí estudiar en esta Universidad, por tus constantes viajes sin importar la distancia, por tu amor sin límites.
A ti papi, por aceptar mi destino y animarme con tus correítos, por educarme desde pequeña con la rectitud que te caracteriza, eso es lo ha hecho de mi quien soy.
A mi querido tío, Jorge, por velar día y noche por mí desde que llegamos juntos a La Habana, por haber sido siempre mi segundo padre, haz cumplido tu promesa.
A mi tía, por su constante preocupación, sus consejitos y las carticas, por su cariño que ha sido igual que el de una madre.
A los Bony en general, la pandilla numerosa que siempre está al tanto de mis estudios y necesidades. A los tíos, tías, primos y primas (te quiero "primi"). En especial a los tres hombres de mi vida, a mis hermanos, por mimarme y cuidarme como siempre les dijo papi.
A mis ángeles de la guarda en la UCI, a la negra (Ene) y a Kike, por ser mi familia más cercana en estos cinco años, por velar mis sueños, mi salud y mis estudios. Los llevo en mi corazón.
A mi amigos especiales, los que están ahora y sé estarán siempre: Edu, Raymond y Ari, por aceptarme tal como soy.
A Carlos Manuel, por darme fuerzas, por estar tan cerca a pesar de encontrarse tan lejos.
A mis pinareños (Aly y Puli) y a Yaimy, por ser tan buenos compañeros.
A Lianet, por las clases de ingeniería. Y a todos los compañeros del proyecto, en especial a Say.

Muchas gracias por el apoyo brindado y el cariño sin límites.

A la memoria de mi adorada madre.

A mis amados padres, los pilares de mi educación y de la persona en la que me he convertido.

A los Bony.

A mi hermano mayor, mi paradigma en todo.

A Ernesto y a Pedro Enrique, para que sigan nuestro camino.

Resumen

El rostro humano es un tema interesante y al mismo tiempo un reto debido a la familiaridad de sus expresiones. La cara es una parte importante del cuerpo que ayuda a distinguir una persona de otra, y sobre todo su estado de ánimo en determinado momento.

Este tema ha constituido el objeto de estudio de muchas investigaciones científicas a lo largo de más de cien años, llevándose a cabo en un principio desde un punto de vista biológico. Los increíbles avances alcanzados en la computación han permitido el desarrollo de numerosas ramas y en años recientes se ha suscitado un creciente interés en el modelado y animación facial tridimensional de caracteres.

El proyecto “Entrenadores Aduaneros” trabaja actualmente con un motor gráfico que brinda técnicas muy avanzadas de animación facial pero con las cuales no se puede aplicar una misma animación a varios personajes, sino que para cada uno hay que definir su propia animación facial. Esta investigación se realiza con la finalidad de desarrollar un sistema de animación facial muscular que permita optimizar el tiempo de trabajo empleado en la definición y aplicación de las animaciones faciales de los personajes del juego “Indicios Aduaneros”.

Palabras Clave: animación facial, motor gráfico, modelado

Tabla de Contenido

Tabla de Contenido

Tabla de Contenido

Introducción

Con el transcurso de los años los videojuegos han adquirido la categoría de una industria muy exitosa comparable con el cine, la música, o las artes plásticas. Dicha industria no es más que el sector económico involucrado en el desarrollo, la mercadotecnia y la venta de videojuegos. Engloba a docenas de disciplinas de trabajo y emplea a miles de personas alrededor del mundo. Los gráficos, el sonido, el lenguaje, la modelación y simulación de efectos especiales, y sobre todo las animaciones faciales (por mencionar algunos aspectos), han mejorado por mucho en comparación con los primeros años.

Sin embargo, en los primeros videojuegos tridimensionales, la poca poligonización que los personajes exhibían en pantalla, acarreaba diversos problemas expresivos. Dichos problemas se refieren a que la capacidad de transmitir emociones, se basaba casi únicamente en la animación corporal. Los personajes gesticulaban y se movían de manera totalmente exagerada, puesto que sus caras no eran capaces de acercarse mínimamente a la expresividad que se espera de un ser humano que siente y padece. La cabeza y la cara son partes del cuerpo muy expresivas y complejas al mismo tiempo. Conocer bien los músculos de la cara y las deformaciones naturales son objetivos muy importantes, sobre todo si se presta especial atención a la boca, dientes, lengua, ojos, párpados, pestañas y cejas.

Los personajes virtuales se están utilizando cada vez más en las interfaces de usuario para mejorar la comunicación persona-máquina, de ahí la importancia de las animaciones faciales de los mismos. Por este motivo, surge la necesidad de definir el comportamiento y la apariencia de estos personajes de una manera intuitiva y visual a través de editores gráficos.

En el Polo Productivo de Realidad Virtual (PRV) de la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI), se desarrolla actualmente el proyecto “Entrenadores Aduaneros”. El mismo surge como una iniciativa de la directiva de la Aduana General de la República (AGR), para mejorar el entrenamiento y la preparación de los inspectores aduaneros que cada año se forjan en la Escuela Nacional de Formación Aduanera (ENFA). Como parte del acuerdo, se desarrolla un juego que permita recrear el flujo de pasajeros en el Aeropuerto Internacional (AI), simulando la búsqueda de indicios sospechosos en personas procedentes del extranjero, pero se torna un tanto difícil el proceso de simulación, debido a la cantidad de personajes que se deben tener en cuenta. Para llevar a cabo dicho proceso, es muy importante tener en cuenta el tema de las expresiones faciales. En el proyecto se trabaja con el motor gráfico *Object-Oriented Graphics Rendering Engine* (OGRE 3D), que entre otras cuenta con

la técnica de animación por vértices en dos vertientes fundamentales; la primera de ellas, animación por *morph,* se emplea principalmente para la animación de objetos; la segunda variante es la animación por poses, única técnica del motor gráfico utilizado para la animación facial, que mediante la mezcla de varios vértices con diferentes niveles de influencia, permite obtener un estado de configuración final de una animación. La animación por poses tiene la desventaja de que para cada personaje que se necesite animar, se deba definir su propia configuración, aunque sea la misma para otro personaje animado anteriormente, conllevando a que se emplee mucho tiempo en la definición de las animaciones cada vez que se hace necesario emplear una de ellas.

Teniendo en cuenta la situación problémica planteada anteriormente, se considera el siguiente **problema científico**: ¿Cómo emplear otras técnicas de animación facial en el juego "Indicios Aduaneros" que permita reutilizar expresiones faciales?

Según el problema científico expuesto, se plantea como **objeto de estudio** técnicas empleadas para la animación facial y como **campo de acción** técnicas de modelos musculares empleadas para la animación facial.

Esta investigación persigue como **objetivo general** diseñar un sistema de animación facial muscular para el juego "Indicios Aduaneros".

Para dar cumplimiento al objetivo propuesto, se concibieron las siguientes **tareas de investigación:**

- ✓ Búsqueda de información sobre las técnicas de animación facial existentes, para tener una visión del estado del arte de las animaciones faciales a nivel mundial y cómo se ha desarrollado esta rama hasta la actualidad.
- ✓ Comparación de las técnicas estudiadas con el objetivo de seleccionar la que se aplicará en el sistema.
- ✓ Estudio de sistemas que implementan animación facial para tener una visión de cómo se puede diseñar el sistema en cuestión.
- ✓ Análisis de las funcionalidades que brinda la herramienta Ogre como motor gráfico para escoger cuáles pueden ser empleadas en el desarrollo de la aplicación.
- ✓ Conceptualización de las herramientas y lenguajes que se utilizarán para el modelado del sistema.

Para el cumplimiento de estas tareas se emplearán varios métodos en la búsqueda y procesamiento de la información tales como:

- ✓ *Analítico - sintético:* para el estudio de los conceptos y técnicas empleados dentro de la Realidad Virtual en la animación facial, analizando todos los documentos para la extracción de los elementos más importantes sobre el tema en cuestión.
- ✓ *Análisis histórico - lógico:* para conocer, con mayor profundidad, los antecedentes y las tendencias actuales referidas a la animación facial, conociendo así la trayectoria histórica real de su desarrollo.
- ✓ *Modelación:* para la elaboración del diseño del sistema mediante los modelos y diagramas necesarios.

El documento consta de 4 capítulos en los que se exponen todo el proceso de desarrollo de esta investigación:

Capítulo 1 "Fundamentación Teórica": En este capítulo se exponen y describen todos los conceptos y técnicas fundamentales relacionados con la animación facial de personajes. También se describen algunas herramientas que implementan sistemas de animación facial basados en modelos musculares, así como las herramientas y lenguajes que se emplearán para el desarrollo de la aplicación.

Capitulo 2 "Propuesta de Solución": Se presenta la propuesta de solución a la problemática planteada y se describe el entorno del sistema representado mediante un modelo de dominio. Además se determinan las funcionalidades que el sistema debe poseer realizando una descripción en detalle de las mismas.

Capítulo 3 "Análisis y Diseño del Sistema": En este capítulo se lleva a cabo el análisis y diseño del sistema, planteado a través de los diagramas de clases del análisis y del diseño, así como de los diagramas de secuencias.

Capítulo 4 "Implementación del Sistema": Se destacan las reglas de codificación a emplear durante la implementación de la aplicación describiéndose cómo debe organizarse la misma y cómo será desplegado el sistema, representado mediante los diagramas de componentes y de despliegue.

Capítulo 1 "Fundamentación Teórica"

1.1 Introducción

La animación es un arte expresivo único ya que provee a la persona que la crea de tener el control sobre la apariencia y el movimiento de los caracteres u objetos que está modelando. En pocas palabras, el creador tiene la autonomía de hacer lo que desee. Esta libertad sin embargo, se convierte en un inconveniente, ya que todos los detalles del proceso de animación como arrugas y arqueado de las cejas deben ser estrictamente controlados, siendo esto una tarea difícil que requiere de mucho trabajo y habilidad.

En los últimos años ha habido un considerable interés en la animación facial de caracteres tridimensionales. La creación de expresiones cada vez más realistas se ha hecho posible en parte gracias al increíble desarrollo de la computación y también de Internet. Este último se ha convertido en parte esencial de la vida de las personas, por lo que ha facilitado el estudio y la búsqueda de información sobre animación facial de una manera fácil y rápida.

La habilidad de modelar el rostro humano y luego animar las sutiles expresiones faciales, se convierte en un desafío importante para los gráficos computacionales. A pesar del uso de muchas técnicas tradicionales como el modelado y renderizado, la animación facial ha sido pobremente definida y continúa siendo un objetivo difícil de alcanzar en la animación por computadora apareciendo poco natural y delicada.

En el presente capítulo se realizará una exposición de los principales conceptos relacionados con la animación facial, entre los cuales se encuentra la definición de animación por computadora y sus características principales, dando paso dentro de esta a la animación facial, los objetivos que persigue y los principales problemas que afronta hoy en día; así como otros conceptos relacionados con los sistemas de tiempo real. Se llevará a cabo un análisis de las distintas técnicas de animación facial que han sido empleadas desde el surgimiento de esta rama, haciendo énfasis en las tendencias que existen actualmente. Además se presentarán algunos sistemas de animación facial basados en modelos musculares, así como los lenguajes y herramientas que se emplearán para el desarrollo de la aplicación.

1.2 Animación por computadora

La *animación* es la simulación de un movimiento, creada por la muestra de una serie de imágenes o cuadros. Un ejemplo sencillo de esto son las caricaturas, que pertenecen a la animación tradicional. [1] Esta no se desarrolló de la noche a la mañana. Se tuvieron que dar muchos pasos pequeños al principio, pero esta forma de expresión ha alcanzado acelerados niveles de innovación de tal forma que cada día se pueden apreciar nuevas técnicas que muchas veces parecen opacar por mucho a sus antecesores. Muchas personas han contribuido a hacer de ella lo que es hoy en día, incluyéndose aquí el fantástico mundo de la animación por computadora.

La *animación por computadora* se puede definir como un formato de presentación de información digital en movimiento a través de una secuencia de imágenes o cuadros creadas o generadas por la computadora; se utiliza principalmente en videojuegos y películas. [1]

Una de sus características fundamentales es que permite crear escenas realmente tridimensionales. La animación 3D se realiza modelando objetos en un ambiente tridimensional, como si se estuviera trabajando con plastilina, pero por computadora. No se elaboran dibujos tal cual, sino que son instrucciones que el operador da a la computadora sobre las características físicas del objeto así como posición, dirección, lugar y dirección de cámara entre otras, para después animarlo (darle movimiento) y renderizarlo (convertirlo a una imagen final).

1.3 Animación facial

Una de las carencias que marca la diferencia entre realidad y ficción es el tema de las expresiones faciales dentro de la animación por computadora. En las primeras animaciones tridimensionales los personajes tenían poca capacidad de transmitir emociones perdiendo así el realismo que necesitan pues apenas daban la sensación de tristeza, felicidad o rabia.

La *animación facial* es principalmente un área de gráficos por ordenador que engloba modelos y técnicas para generar y animar imágenes de la cabeza y la cara. [2] La importancia de la expresión de los rostros humanos ya sea en la comunicación verbal o no verbal, y los avances alcanzados en los gráficos por ordenador, ha causado un considerable interés científico, tecnológico e incluso artístico en cuanto a esta área.

El objetivo principal de la animación facial automatizada, es que permite tomar un modelo 3D del rostro humano como una expresión simple (por lo general neutral) y generar una serie de expresiones y poses faciales sin requerir de la intervención de un animador tradicional. [3] Se ha hecho muy conocida y

popular a través de largometrajes de animación y juegos de ordenador, pero sus aplicaciones abarcan muchas más áreas como la comunicación, la educación, la simulación científica, y los sistemas basados en agentes.

1.4 Principales problemas de la animación facial

El problema de animar el rostro humano no es en absoluto sencillo. La cara contiene 268 músculos voluntarios de tres tipos diferentes. En determinadas partes los músculos pueden contraerse formando arrugas en la piel. Este tipo de deformación superficial no es fácil de capturar utilizando sencillas técnicas de animación y no es intuitivamente fácil de modelar de forma matemática. Entre las regiones de la cara, la boca es la más complicada en términos de su estructura anatómica y su comportamiento de deformación. Su complejidad lleva a considerar su modelado y animación independiente del resto de la cara. Muchas de las ideas básicas y los métodos para el modelado de la boca son versiones optimizadas de los métodos generales empleados para la animación facial. Finalmente debe tenerse en cuenta la estructura del cráneo. Afortunadamente el cráneo sólo contiene un único conjunto (la mandíbula) y puede ser tratado como un cuerpo rígido. Todas las demás expresiones son el resultado de los movimientos de los tejidos blandos. [3]

Es muy conveniente automatizar completamente el proceso de animación ya que esto reduce la necesidad de trabajar con animadores además de la reducción de la enorme cantidad de tiempo requerido para producir personajes virtuales realistas.

1.5 Técnicas empleadas para la animación facial

Muchos esfuerzos de investigación han intentado generar modelos de animación facial realistas. Los más ambiciosos intentos trabajan en el modelado y renderizado en tiempo real, pero debido a la complejidad de la anatomía facial humana no existe ningún sistema en tiempo real que pueda captar las sutiles expresiones y emociones transmitidas a través del rostro.

A pesar de que algunos trabajos recientes producen resultados reales con relativamente rápido rendimiento, el proceso para generar expresiones faciales implica una amplia intervención humana. El objetivo final de los modelos diseñados con este fin es obtener un sistema que cree una animación realista, opere en tiempo real, sea automatizado en la mayor medida posible, y se adapte fácilmente a los distintos rostros.

1.5.1 Interpolaciones

La técnica de interpolación ofrece un enfoque intuitivo para la animación facial. Normalmente, una función de interpolación especifica un movimiento suave entre dos fotogramas claves en posiciones extremas durante un intervalo de tiempo normalizado (Ver Figura 1). Cuando cuatro fotogramas están involucrados en lugar de dos, la interpolación bilineal genera una mayor variedad de expresiones que la interpolación lineal, y cuando se combina de forma simultánea con el morphing de imágenes, crea una amplia gama de cambios en las expresiones faciales. [4]

Figura 1. Interpolación lineal de dos fotogramas

Las imágenes interpoladas son generadas variando los parámetros de las funciones de interpolación. Las interpolaciones geométricas actualizan directamente las posiciones 2D o 3D de la malla de vértices, mientras que la interpolación de parámetros controla las funciones que mueven los vértices indirectamente. [4]

Aunque son rápidas y generan fácilmente expresiones faciales primitivas, la capacidad de esta técnica de crear una amplia gama de configuraciones faciales reales está muy restringida. Las combinaciones de animaciones independientes son difíciles de producir. Se necesita un modelo completo de la cara para cada fotograma clave y la topología de todos los modelos de fotogramas claves debe ser idéntica. Aún así es un buen método para producir una pequeña serie de animaciones de unos pocos fotogramas claves.

1.5.2 Parametrizaciones

Las técnicas de parametrización superan algunas de las limitantes y restricciones de las técnicas de interpolación. Las parametrizaciones ideales especifican cualquier expresión posible a partir de la

combinación de un conjunto de parámetros que manipulan regiones o características locales de la cara. A diferencia de las técnicas de interpolación, las parametrizaciones permiten tener un control explícito de las configuraciones faciales. La combinación de parámetros proporciona una amplia gama de expresiones faciales con relativamente bajos costes computacionales (Ver Figura 2). [4]

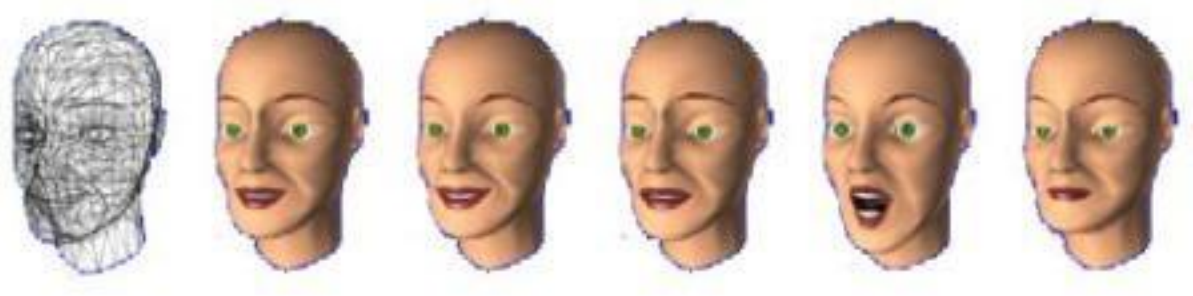

Figura 2. Conjunto de expresiones basadas en animación facial parametrizada.

En el caso de la boca, esta técnica usualmente requiere un número significante de parámetros de entrada para el control de las animaciones. El ancho y altura de la boca son el par de parámetros que determinan el ángulo de apertura de la misma, así como los coeficientes de salida.

Debido a que no existe una forma sistemática para decidir entre dos parámetros que actúan sobre los mismos vértices durante la combinación de expresiones, la parametrización rara vez produce expresiones naturales cuando existe un conflicto entre parámetros. Por esta razón, las parametrizaciones sólo están diseñadas para actuar sobre regiones faciales específicas, constituyendo esto una limitante para esta técnica. Otra limitación de la parametrización es que la elección del conjunto de parámetros depende de la topología de la malla facial y, por tanto, no es posible realizar una parametrización genérica completa. [4]

1.5.3 Morphing 2D y 3D

El morphing se vale de dos fotografías, una de inicio y otra final. Mediante esta técnica se logra una secuencia que une las dos. Si se captura la imagen intermedia de la misma, se vería una unión de ambas imágenes que en el caso de dos personas resulta una tercera irreal con los rasgos de las dos reales. [5]

En el morphing 2D las imágenes, cuidadosamente seleccionadas, pueden proporcionar animaciones faciales bastante realistas. El realismo, sin embargo, exige una amplia labor manual para equilibrar el color y la correspondencia y sintonización de la deformación y disolución de los parámetros.

Para superar sus limitaciones, se combina el morphing 2D con transformaciones 3D de un modelo geométrico. Las expresiones faciales principales se animan con interpolaciones geométricas 3D, mientras que el morphing se realiza entre los correspondientes mapas de texturas. Este enfoque alcanza el punto de vista independiente del realismo, sin embargo, las animaciones están todavía limitadas a interpolaciones entre expresiones claves predefinidas. [4]

Los métodos de morphing 2D y 3D pueden producir expresiones faciales realistas, pero comparten limitaciones similares con los enfoques de la interpolación. La selección de puntos en las imágenes es una tarea manualmente intensa y no son generalizables a los diferentes tipos de cara.

1.5.4 Sistema de Codificación de Acción Facial

Los primeros intentos de definir adecuadamente la animación facial se llevó a cabo con el Sistema de Codificación de Acción Facial (FACS). Inicialmente destinado sólo para la descripción de la acción facial y no para la animación, el FACS es el método más versátil utilizado para medir y describir las expresiones faciales. El objetivo primordial en el desarrollo del FACS era construir un sistema global que podría distinguir visualmente todos los posibles movimientos faciales distinguibles, creando un medio fiable para determinar la categoría o categorías en las que se puede ubicar a cada expresión facial. Por tanto, se trata de una norma común para clasificar sistemáticamente la expresión física de las emociones, y que ha demostrado ser útil en el campo de la animación facial. [4]

El FACS emplea un método de control de parámetros, donde la animación se convierte en un proceso de especificación y control de valores en función del tiempo. El cambio de estos parámetros puede proporcionar un modelo base para crear nuevas animaciones. Este modelo incluye 44 Unidades de Acción (AU) básicas relativas a las expresiones donde intervienen los músculos, y otras 20 unidades para los movimientos de la cabeza y la mirada. [4]

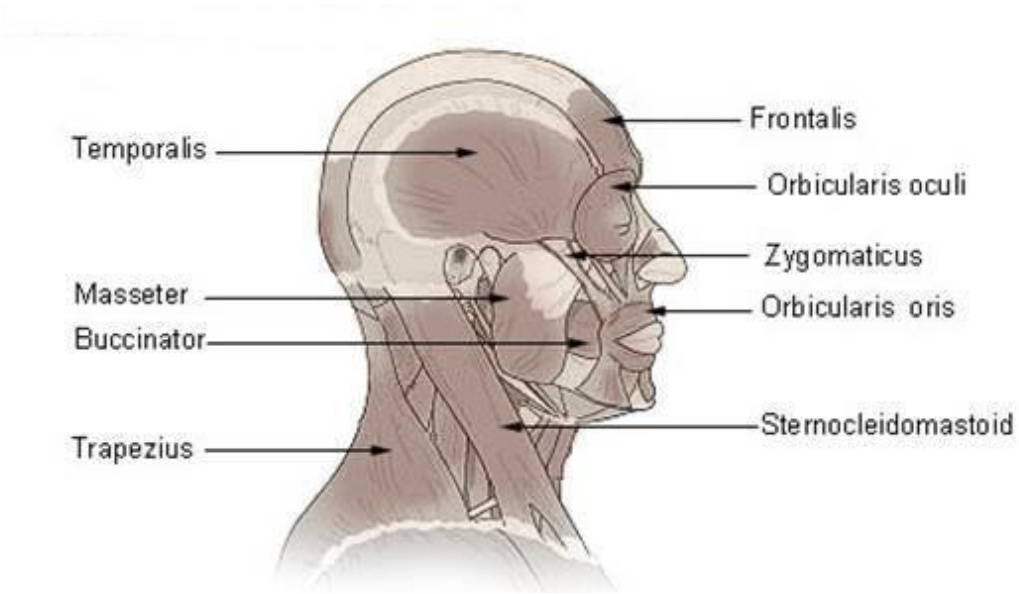

Figura 3. Músculos de la cabeza y el cuello

Las AU representan acciones, no músculos, y esto es debido a dos razones fundamentales. Primero, para unas pocas expresiones, se combinan más de un músculo en una sola AU, ya que el cambio producido aparentemente por un solo músculo, no se puede distinguir. Segundo, los cambios de apariencia producidos por un músculo, en ocasiones son separados en dos o más AU para representar relativamente, acciones independientes de diferentes partes del músculo. [4]

Los métodos de animación que emplean modelos musculares o de simulación de los músculos superan las dificultades de correspondencia e iluminación de las técnicas de interpolación y morphing. El modelado físico de los músculos describe matemáticamente las propiedades y el comportamiento de la piel, los huesos, y sistemas musculares. El FACS constituye la base de muchos modelos de animación facial empleados hoy en día.

Los modelos musculares basados en física se dividen en tres categorías: los sistemas de masa-resorte, la representación de vectores y capas de mallas -resorte. Los métodos de masa-resorte propagan la fuerza de los músculos en una malla-resorte elástica que modela la deformación de la piel; el enfoque vectorial deforma una malla facial utilizando movimientos en determinadas regiones y las capas de mallas-resorte extienden una estructura de masa-resorte en tres capas de mallas conectadas para modelar del comportamiento anatómico facial más fielmente.

1.5.4.1 Modelo masa-resorte

Este modelo se centra en el modelado de los músculos y de la estructura de la cara. Las fuerzas que se aplican a las mallas elásticas a través de arcos que representan los músculos permiten generar expresiones faciales realistas. El modelo de masa-resorte fue mejorado convirtiéndose en un modelo que representa los músculos como colecciones de bloques funcionales, establecidos en determinadas regiones de la estructura facial. El mismo consiste de 38 bloques interconectados por una red resorte. En este caso las unidades de acción se crean aplicando las fuerzas musculares para deformar la red resorte. [4]

En el modelado de la boca y la animación de la voz, los sistemas de masa-resorte a menudo modelan la estructura fonética del habla. Debido a que la animación de la boca se genera a partir de relativamente pocos músculos, la representación del realismo es en gran medida independiente del número de elementos representados.

1.5.4.2 Vectores musculares

El modelo de vectores musculares es un modelo muy exitoso en el cual se representa un campo delimitado de deformación que permite modelar la acción de los músculos sobre la piel.

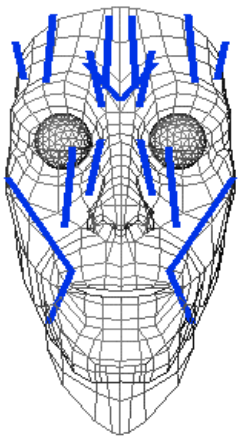

Figura 3. Modelo de vectores lineales

La definición de un músculo incluye la dirección del vector de campo, un punto de unión (al hueso) u origen, y un punto de inserción (en la piel) (Ver Figura 4). Como los músculos se contraen cerca de la piel, los vértices están sugestionados con más fuerza a lo largo del vector de dirección del músculo. [4]

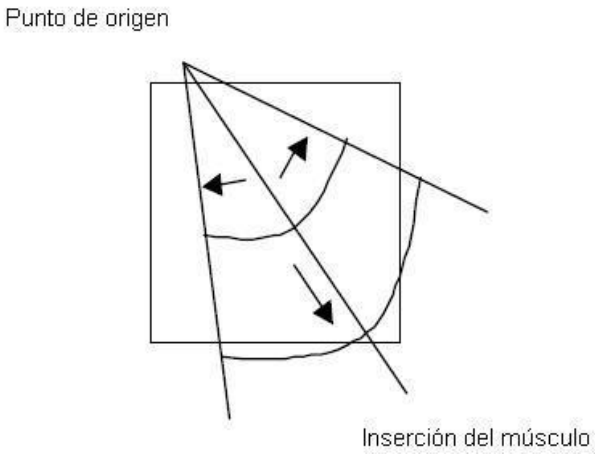

Figura 4. Zona de influencia del modelo de vectores musculares. Las deformaciones se decrementan en la dirección de las flechas.

El modelo también permite analizar los músculos involucrados en el movimiento de la boca, y en este caso es representada como un elipsoide paramétrico simplificado.

La colocación de los vectores musculares en posiciones anatómicamente correctas, puede llegar a ser una tarea desalentadora. El proceso implica realizar pruebas manuales sin ninguna garantía de eficiencia o de una óptima colocación de los vectores, esta última en caso de ser incorrecta, resulta en animaciones poco naturales y no deseadas. Sin embargo, el modelo de vectores musculares se utiliza ampliamente debido a su representación compacta y la independencia de la estructura de la malla facial.

1.5.4.3 Capas de masa-resorte

Este método permite modelar de forma detallada la estructura anatómica y la dinámica de la cara. Haciendo uso del mismo tanto la piel como el tejido graso y los huesos se modelan empleando un modelo de masa-resorte de tres capas. Los resortes elásticos conectan cada malla y cada capa (Ver Figura 5). Las fuerzas musculares se propagan a través del sistema de mallas para crear la animación. [4]

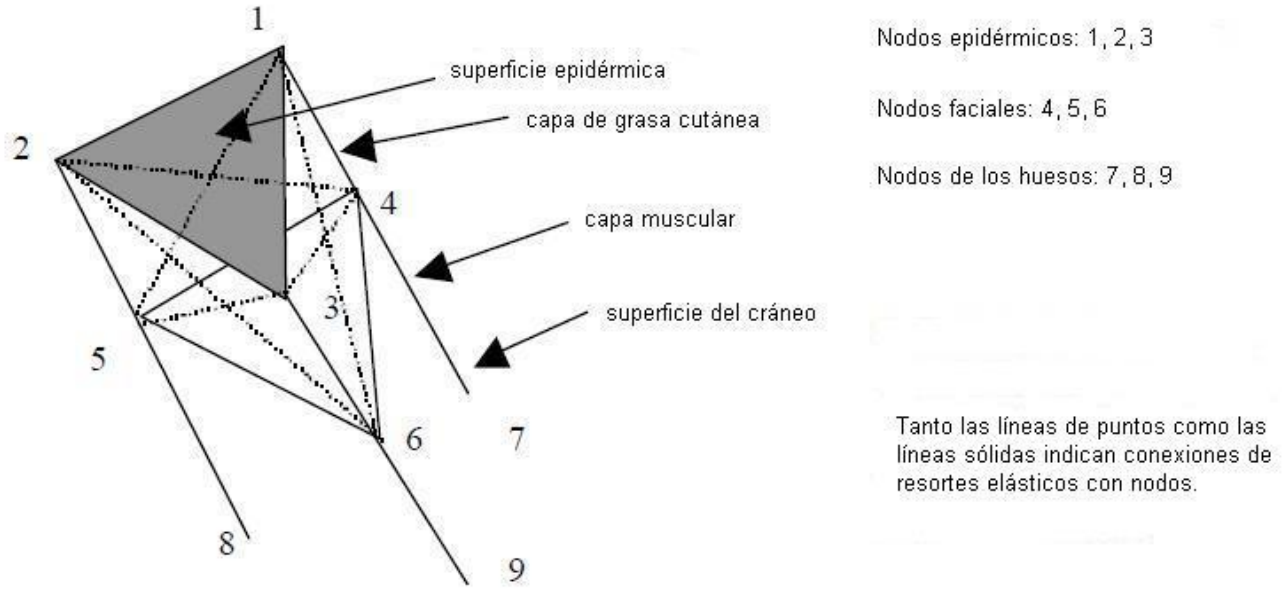

Figura 5. Modelo Capas de masa-resorte

El método es también empleado para la animación de la boca. A la piel de la cara se le añade un mecanismo de control que actúa sobre la forma de la boca como un modelo de tres capas de masa-resorte con coeficientes de elasticidad adecuadamente elegidos. Los valores de contracción muscular para cada fonema están determinados por la comparación de los puntos correspondientes en las fotos y el modelo. Durante la animación del habla, las formas intermedias de la boca se definen mediante una interpolación lineal de los parámetros de la fuerza muscular del resorte.

La técnica incluye además la simplificación de la implementación ya que todo es manejado por un único sistema; el modelo es capaz de generar expresiones en tiempo real y proporciona algunos efectos empleados para crear arrugas. Sin embargo, existe falta de realismo en el modelo de los músculos y huesos y el mismo no puede ser controlado a través de activaciones musculares. [3] A esto se le añade que la simulación de deformaciones volumétricas empleando el sistema de tres capas requiere un amplio cálculo para poder generar las expresiones en tiempo real.

1.5.5 Técnicas para el modelado de arrugas

Las arrugas son también un elemento importante para el modelado y la animación facial. Estas ayudan en el reconocimiento de expresiones tales como la edad de una persona. Existen dos tipos de arrugas, las temporales, que aparecen por un corto período de tiempo en las expresiones, y las que con el tiempo forman rasgos permanentes de la cara.

Las arrugas y pliegues son difíciles de modelar con técnicas como la simulación de los músculos o la parametrización, ya que estos métodos están diseñados para producir deformaciones ligeras. Los modelos basados en la física de los músculos unidos a la plasticidad o viscosidad, y las técnicas de textura como el Bump Mapping, son más adecuadas para la representación de estos elementos.

1.5.5.1 Modelado de arrugas con Bump Mapping

El Bump Mapping produce perturbaciones de las normales de la superficie que alteran el sombreado de la misma. Algunas arrugas arbitrarias pueden aparecer en una superficie geométrica lisa mediante la definición de funciones destinadas a realizar esta tarea. Esta técnica genera arrugas fácilmente variando los parámetros de dicha función (Ver Figura 6). [4]

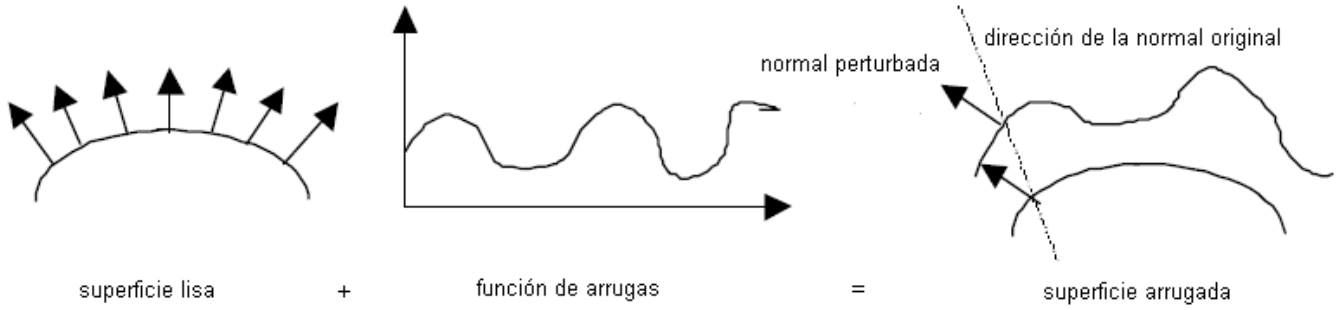

Figura 6. Generación de superficie arrugada usando la técnica de Bump Mapping

La técnica del Bump Mapping es relativamente exigente de forma computacional, ya que requiere aproximadamente el doble del esfuerzo necesario para el color convencional empleado para el mapping de texturas.

1.5.5.2 Arrugas basadas en modelos físicos

Estos modelos utilizan las propiedades plástico-visco-elásticas de la piel. La viscosidad y la plasticidad son dos de las propiedades canónicas inelásticas. La viscosidad es responsable de la

deformación en dependencia del tiempo, mientras que la plasticidad se emplea para la deformación permanente no invertible que se produce cuando una fuerza muscular va más allá de un umbral. [4]

Para generar arrugas expresivas, la superficie de la piel simulada se deforma suavemente por la fuerza de los músculos. En este punto entra en juego la plasticidad, reduciendo algunos efectos causados por la elasticidad, y formando arrugas permanentes. La plasticidad no se produce en todos los puntos simultáneamente, sino que se produce en los puntos que más destacaron por contracciones musculares.

Por la repetición de este proceso en el tiempo, las arrugas se destacan cada vez más en el modelo facial. El modelo es una versión simplificada de la anatomía de la cabeza, sin embargo, los huesos son ignorados, y las capas musculares y de grasa se ubican en función de la superficie de la piel.

1.5.6 Manipulación de texturas

Las imágenes faciales sintéticas derivan su color ya sea del sombreado o del texturizado. El sombreado proporciona valores para dar color a cada pixel teniendo en cuenta las propiedades de la superficie y su iluminación. Debido a la sutileza que conlleva dar color a la piel humana, los sencillos modelos de sombreado generalmente no producen un adecuado realismo en esta parte de la animación. Las texturas en cambio, hacen posible realizar variaciones de las propiedades de la superficie en cada pixel, lo que crea la apariencia de los detalles ausentes en su geometría. En consecuencia, las texturas se utilizan ampliamente para lograr efectos de realismo en la imagen facial. [4]

Cuando la geometría de los objetos 3D o el punto de vista cambian, se producen nuevos mapeos de texturas para la visualización óptima. Este mapeo se realiza en tiempo real debido a la simplicidad y la eficacia del algoritmo propuesto. En dicho algoritmo, un mapeo de la textura es llevado a cabo por una función lineal en cada una de las pequeñas regiones que la forman. Las expresiones faciales realistas y sus animaciones son sintetizadas por interpolación y extrapolación entre múltiples superficies 3D, y el mapeo dinámico de texturas en función de la óptica y los cambios de la geometría.

1.5.7 Método de Elemento Finito

El método de elemento finito (FEM, de sus siglas en inglés Finite Element Method) es un enfoque numérico que permite hacer una aproximación de la física de un objeto arbitrario complejo. Implícitamente define funciones de interpolación entre nodos para las propiedades físicas del material. Un objeto se descompone en elementos de área o volumen, cada uno dotado de parámetros físicos. [4]

El FEM es muy utilizado para animar los labios. Los parámetros del modelo son determinados formando un conjunto de propuestas de la medida del labio, para reducir al mínimo la tensión transmitida en todo la estructura elástica lineal del elemento finito. Los modelos en 2D sufren complicaciones causadas por los cambios realizados en las formas de las proyecciones del labio. Al modelar la verdadera estructura tridimensional de los labios, las variaciones complejas y no lineales en las proyecciones 2D se convierten en simples cambios de los parámetros lineales. El difícil problema del control de los músculos asociados es minimizado por la etapa de formación, así como los problemas de exactitud resultantes del uso de fotogramas para la animación de la boca.

El modelo de elementos finitos proporciona resultados más exactos y estables que los diferentes modelos de masa-resorte, sin embargo, son mucho más costosos computacionalmente. El tiempo de cálculo, incluso para animaciones simples, es demasiado amplio para considerar su uso donde puede ser necesario obtener resultados interactivos. [3]

1.5.8 Parches

Los parches (o conjunto de splines) definen indirectamente una ligera curva de superficie a partir de un conjunto de puntos de control. Un conjunto pequeño de puntos de control pueden definir una superficie compleja. Un tipo de spline son las llamadas NURBS (Non-Uniform Rational B-Splines). Las NURBS son representaciones matemáticas de geometría en 3D capaces de describir cualquier forma con precisión, desde simples líneas en 2D, círculos, arcos o curvas, hasta los más complejos sólidos o superficies orgánicas de forma libre en 3D. Este grupo además permite a cada punto de control tener su propio peso. Gracias a su flexibilidad y precisión, se pueden utilizar modelos NURBS en cualquier proceso por lo que son muy adecuados para el modelado facial. [6]

A pesar de las facilidades que brindan tienen varias cuestiones en contra. Para crear un modelo facial utilizando NURBS se deben utilizar puntos de control ubicados uniformemente en toda la cara, pero esta tiende a tener zonas de alto detalle como los ojos, la boca y las orejas, y áreas de bajo detalle como la frente y las mejillas. Las NURBS trabajan esta parte de dos formas: realizando un largo ajuste de alta densidad en todas partes ya que hace falta en las áreas de mayor detalle, o bien haciendo un ajuste de baja densidad para las áreas de bajo detalle y uno de alta densidad para zonas de alto detalle. Esto por supuesto implica una pérdida de tiempo. Otra cuestión es que el rostro tiene muchas arrugas y representa un gran desafío a la hora de hacer los pliegues. [6]

1.5.9 Captura de movimiento

La captura de movimiento es el proceso de convertir los movimientos de la cara en una base de datos digital haciendo uso de cámaras o de escáneres laser. Esta base de datos puede ser empleada luego para producir animaciones digitales para películas, juegos y otras áreas. Como la propuesta de los caracteres se deriva de los movimientos de una persona real, resulta más real y matizada la animación de los caracteres digitales, que si la animación hubiera sido creada manualmente.

Como se muestra en la figura 7, varios marcadores se colocan en puntos específicos en el rostro de un actor durante la captura óptica del movimiento facial.

Figura 7. Captura del movimiento de la cara

La captura de movimiento es el método de ejecución más utilizado, principalmente en la producción de películas, y se realiza normalmente a través de sistemas de seguimiento de una o varias cámaras, como pequeños puntos reflectantes que se colocan en posiciones estratégicas en la cara de los artistas.

1.6 Principales tendencias dentro de la animación facial

Desde que se inició la “revolución 3D” en el ámbito de los juegos de computadora, la tendencia de la tecnología aplicada a este rubro ha sido trasladar el trabajo del procesamiento de gráficos tridimensionales, desde la CPU hacia la tarjeta de video.

En este sentido ocurrieron muchos cambios, pero sin dudas el mayor de ellos ocurrió a partir de la incorporación de los píxel shaders y vertex shaders. Esto permite a los programadores una mayor libertad a la hora de diseñar gráficos en tres dimensiones, ya que puede tratarse a cada píxel y cada vértice por

separado. De esta manera, los efectos especiales y de iluminación pueden crearse mucho más detallados, sucediendo lo mismo con la geometría de los objetos.

1.6.1 Vertex shaders y pixel shaders.

A grandes rasgos, los *shaders* son simples programas que transforman un vértice (denominado vertex) o bien un pixel. Actualmente no sólo sirven para iluminación y sombreado, como indica su nombre, sino que además sirven para crear gráficos más ricos, como animaciones, efectos de partículas, etc. [7]

Un *vertex shader* es una función de procesado gráfico que manipula los valores de un vértice en un plano 3D mediante operaciones matemáticas sobre un objeto. Estas variaciones pueden ser diferencias en el color, en las coordenadas de la textura, en la orientación en el espacio o en el tamaño del punto. El *shader* no opera sobre una primitiva (un triángulo, por ejemplo), sino sobre un solo vértice cada vez, tampoco puede crearlos ni destruirlos sino solamente manipularlos. [7]

Con esto se puede lograr ciertos efectos específicos, como los que tienen que ver con la deformación en tiempo real de un elemento; por ejemplo, el movimiento de una ola. De esta forma toma una gran importancia en el tratamiento de las superficies curvas, y su avance se ve reflejado en los videojuegos más avanzados de la actualidad, particularmente, en el diseño de los personajes y sus expresiones corporales.

En cambio, un píxel shader no interviene en el proceso de la definición del “esqueleto” de la escena, sino que forma parte de la etapa de rendering. Allí es donde se aplican las texturas y se tratan los píxeles que forman parte de ella.

Básicamente, un *píxel shader* especifica el color de un píxel. En el caso más simple, el shader hace esto multiplicando el color de la iluminación producida por el vertex shader con el color de la textura obtenida para ese píxel en particular. [8] Este tratamiento individual de los píxeles permite que se realicen cálculos principalmente relacionados con la iluminación del elemento del cual forman parte en la escena, y en tiempo real. La particularidad de los píxel shaders es que, a diferencia de los vertex shaders, requieren de un soporte de hardware compatible.

Un *vertex program* es el código que funciona en cada vértice y que no puede añadir o restar vértices. Una manera simple de analizarlo es que todos los vértices pueden ser procesados en paralelo. Los *fragment programs* funcionan en cada fragmento y manejan tanto la textura como las operaciones de búsqueda. [9]

Al utilizar *vertex programs* y *fragment programs* la mayor parte de las funcionalidades previamente codificadas en el hardware de gráficos, están apagadas. Por ejemplo, cuando se emplea un *vertex program* se apagan las transformaciones 3D estándares, la iluminación y la generación textura completamente. Del mismo modo, al usar un *fragment program* se reemplaza cualquier textura que haya sido definida con anterioridad. Escribir *vertex programs* o *fragment programs* requiere un conocimiento profundo de transformaciones 3D, iluminación y coordinación de espacios.

La principal ventaja es que, como su naturaleza lo indica, pueden ser programados por el desarrollador, otorgando una flexibilidad que hasta antes de la aparición de los *shaders* era poco más que impensada. Recursos como las operaciones condicionales o los saltos se utilizan de forma similar que en los lenguajes más conocidos. Sin los *shaders*, muchos de los efectos eran realizados en conjunto con la unidad de procesamiento central, disminuyendo en gran medida el rendimiento y limitando el avance a nivel gráfico de los mismos.

Actualmente, los *shaders* no sólo sirven para iluminación y sombreado, como indica su nombre, sino que además sirven para crear efectos gráficos más ricos, como animaciones, efectos de partículas, etc. Los vertex *shaders* programables permiten una ilimitada gama de efectos visuales que pueden ser en tiempo real. Ahora, los desarrolladores pueden utilizar vertex *shaders* para dar vida y personalidad a los personajes y ambientes, como la niebla que se sumerge en un valle y los rizos sobre una colina, o las animaciones faciales reales tales como los hoyuelos o arrugas que aparecen cuando una persona sonríe.

1.6.2 Animación esqueletal

La animación esqueletal es una técnica empleada para crear modelos de un caracter. [10] Cada uno constituye una jerarquía de huesos y estos controlan la deformación de la malla que lo representa. [11]

La técnica de animación esqueletal se puede implementar utilizando *shaders*, específicamente *vertex programs*, siendo especialmente importante para modelos grandes y detallados. Como el nombre lo indica, este tipo de animación se emplea mayormente para la representación de los huesos del cuerpo, en el caso de un personaje, por lo que no hace énfasis específicamente en el rostro, no siendo recomendable para la animación facial.

1.6.3 Animación de vértices

La animación de vértices trata el uso de información sobre el movimiento de los vértices utilizados directamente para animar la malla que representa una superficie. Actualmente existen dos subtipos de animación de vértices aunque ambas técnicas pueden ser implementadas utilizando solamente la animación por poses. [12]

1.6.3.1 Animación por morph

La animación por morph trabaja almacenando instantáneas de las posiciones absolutas de los vértices en cada fotograma e interpolándolos entre ellos. La animación por morph es principalmente útil para la animación de los objetos que no pueden ser adecuadamente tratados con animación esqueletal, esto es mayormente para los objetos que tienen que cambiar radicalmente la estructura y la forma como parte de la animación, de tal manera que una estructura ósea no es adecuada. [12]

Para la animación en un vertex shader, la animación por morph es bastante simple y sólo requiere de 2 buffers de vértices, con los datos de la posición absoluta y un factor de interpolación.

Debido a que se utilizan posiciones absolutas, este enfoque simplista no es compatible con la mezcla de múltiples animaciones por morph. Si se activa más de una animación para el mismo vértice de datos, sólo el último surtirá efecto realmente. Esto también significa que el 'peso' no se utiliza cuando se emplea la técnica de morph. [12]

1.6.3.2 Animación por poses

La animación por poses es una técnica más compleja que permite mezclar múltiples imágenes discretas, expresadas como offsets a la base de datos de vértices, no siendo necesario que cada vértice tenga un offset, y con diferentes pesos para proporcionar un resultado final. Un uso común de esta técnica es la animación facial, donde cada expresión facial se coloca en una imagen, y es empleada ya sea en la transición de una expresión a otra, o para combinar las expresiones por completo. [12]

El inconveniente de utilizar animación por poses es que puede ser más difícil de establecer, lo que requiere definir las imágenes por separado y luego referenciarlas como fotogramas. También, ya que utiliza más buffers, si se está empleando vertex shaders, se debe tener cuidado con el número de imágenes que se mezclan a la vez, por lo que es necesario definir un número máximo de apoyo en definición del vertex program.

La animación por morph no puede ser mezclada con otros tipos de animación de vértices; sin embargo la animación por poses puede mezclarse con otras, y ambos tipos pueden ser combinados con animación esqueletal. Además la animación por morph se puede expresar como animación por poses, pero no viceversa.

1.7 Sistemas de animación facial basados en modelos musculares.

1.7.1 Langwidere

Langwidere es la base para un sistema flexible capaz de imitar una amplia gama de características y acciones, como el habla o expresar emociones. Integra un sistema jerárquico de modelado spline con la simulación de músculos, basado en la deformación del área local de la superficie. La representación multinivel de la forma permite controlar el grado de deformaciones, y al mismo tiempo reducir el número de vértices de control necesarios para definir la superficie. [13]

El modelo de la cara se construye desde una única superficie cerrada que permite modelar estructuras internas tales como la lengua y los dientes. Los músculos simulados se adjuntan a diversos niveles de la superficie sustituyendo algunos huesos como el cráneo y la mandíbula. La combinación de un modelo jerárquico con la simulación de músculos proporciona precisión, un flexible control de la superficie y apoya la fácil generación de nuevos personajes con un mínimo de re codificación. [14]

Anatómicamente, la fiel colocación de los músculos permite generar expresiones naturales, tal como se define en el Sistema de Codificación de Acción Facial, el sistema de notación de animación facial más utilizado.

1.7.2 Interface

Interface es un sistema que utiliza un conjunto sencillo de expresiones faciales pre-modeladas para crear una amplia gama de emociones, diversas posiciones de la boca y movimientos completos de la cabeza en general. La animación se realiza a través de capas agrupadas de acciones, que combinadas le dan al actor virtual total libertad para llevar a cabo diferentes acciones de forma independiente y simultáneamente, sin la intervención del animador. El sistema fue implementado con éxito utilizando los lenguajes Java y VRML, y puede ser ejecutado a través de Internet en PCs o estaciones de trabajo con un navegador web Java. Las aplicaciones que pueden beneficiarse de este sistema son los agentes de interfaz, sistemas de realidad virtual y software de animación, entre otros. [14]

1.7.3 Expression Toolkit

Expresion Toolkit es un sistema de animación basado en el modelo anatómico de la cara, que hace uso de simulaciones básicas musculares. Esta herramienta simplifica la creación de personajes vivos, lo que permite crear un rostro en cuestión de horas en lugar de días. Escrito en C++ y OpenGL, *Expresion Toolkit* es un sistema de uso general para animaciones faciales en tiempo real, ya sea en juegos o aplicaciones web. [15]

1.7.3.1 Características de Expression Toolkit

- ✓ Herramientas para la construcción:
  - Herramientas de creación y exportadores de 3D Studio Max.

- ✓ Anatomía:
  - 6 tipos básicos de músculos.
  - Párpados parametrizados.
  - Mandíbula ponderada.

- ✓ Sistema de animación:
  - Rendimiento en tiempo real.
  - Sistema de animación basado en eventos.
  - Soporte para poses o imágenes básicas.
  - Lenguaje de scripting para la generación de expresiones compuestas.

1.7.3.2 Panorama de la API de Expression Toolkit

El sistema de animación es un asunto algo sencillo ya que es un sistema orientado a eventos. Dicho sistema está encapsulado por la clase *expHead* y en la función miembro *expHead::event* es donde se procesan todos los eventos. [15]

Las animaciones por su parte consisten en un vector de cadenas llamado *Expression*. El número de canales de una expresión se crea en el tiempo de inicio mediante el cálculo del número de músculos y

el número de canales adicionales. Cada expresión es una pose facial, que puede ser la transición a/composición con otra pose facial. El sistema de animación en efecto es una máquina de estados, con cada nuevo caso, la animación es la transición de la anterior. [15]

La segunda parte de la caja de herramientas de este sistema, es un compositor de vértice generalizado, responsable de la deformación de la malla en un orden que se define dentro de la clase *expHead*. Esta funcionalidad se implementa a través de la interfaz *ExpMeshAdapter*. El adaptador es un resumen de interfaz que permite a *Expression* comunicarse con una malla sin saber la forma en que está estructurada en memoria. Los gastos generales de la interfaz no son totalmente despreciables, una función virtual debe ser llamada para obtener y configurar cada método que trata los vértices, y la clase que define la malla debe heredar de la interfaz. [15]

La esencia del sistema y lo que hace de este un sistema de animación facial, es el modelo anatómico. Una clase base virtual llamada *Músculo* actúa como interfaz para todos los tipos de músculos definidos en el sistema. Esta clase define algunas propiedades y operaciones como la malla donde se actúa, un nombre, un método de activación y una función para dibujar. [15]

1.7.3.3 Licencia

Expression Toolkit se publica bajo la Licencia Pública Q (licencia QPL), un acuerdo de licencia Open Source, y un acuerdo más tradicional de licencia comercial. El software puede ser utilizado en un paquete de software, siempre y cuando las condiciones de la QPL se cumplan. Una de las condiciones más notables de la QPL es que cuando las modificaciones de los software se distribuye bajo esta licencia, un derecho no exclusivo y libre de regalías se concede a los primeros desarrolladores del software para distribuir las modificaciones realizadas por otras personas, en versiones futuras del software siempre y cuando tales versiones queden disponibles bajo estos términos. [15]

1.8 Sistemas de tiempo real

La generación y animación en tiempo real de complejas escenas tridimensionales todavía se considera una difícil tarea, pero sin embargo se ha convertido en un requisito para simuladores, juegos y aplicaciones de realidad virtual. En particular, el modelado realista de determinados fenómenos naturales requiere de un gran número de polígonos, por lo que requieren también de un gran número de cálculos de renderizado. En tiempo real, la animación facial es aún más difícil, debido al equilibrio entre el rendimiento necesario para la animación en tiempo real y el nivel de detalle requerido para la renderización realista.

El tiempo se conoce por lo general como una representación matemática de qué punto del día es. Del latín *tempus*, la palabra *tiempo* es la magnitud física que permite medir la duración o separación de las cosas sujetas a cambio, o sea, el período que transcurre desde que el sistema aparenta un estado hasta el instante en que dicho estado registra una variación perceptible para el observador. [16]

La palabra *real* se refiere sencillamente a que la reacción de un sistema ante determinados eventos debe ocurrir durante su evolución.

El *tiempo real* se define como la simultaneidad entre el registro de un evento y el momento en que ocurre, también se le llama procesamiento sincrónico. [17] Se dice que un ordenador trabaja en *tiempo real* cuando realiza una transacción que le ha sido ordenada desde un terminal en ese mismo momento, sin espera alguna. [18]

Otra definición que se puede dar para tiempo real es la respuesta inmediata de forma que el tiempo que transcurre en el mundo virtual se corresponde con el tiempo real. [19]

Por el contrario, el tiempo no real es un término usado para describir un proceso o evento que no ocurre de forma inmediata. Un ejemplo de ello pueden ser los fórums, ya que las respuestas con frecuencia no se dan inmediatamente y pueden tomar varias horas e incluso días.

Es bien sabido que los sistemas de tiempo real pueden llevar el control de eventos que ocurren en el mundo real, por lo tanto son sistemas que interactúan activamente con un entorno con dinámica conocida dando respuesta a un estímulo dentro de un tiempo especificado. Es decir, los sistemas de tiempo real interactúan con el entorno que se le presente y pueden ejecutar acciones de respuesta para determinados estímulos de dicho entorno.

Todos los sistemas de tiempo real tienen la facultad de ejecutar actividades o tareas en intervalos de tiempo bien definidos. Todas las tareas son ejecutadas inmediatamente en una forma concurrente, para sincronizar el funcionamiento del sistema con la simultaneidad de acciones que se presentan en el mundo físico.

La eficiencia de estos sistemas no solo depende de la exactitud de los resultados de cómputo, sino también del momento en que los entrega, siendo la predictibilidad su característica principal. Además, un sistema de tiempo real debe necesariamente tener la característica de un tiempo de respuesta crítico. Por ejemplo, el software encargado de controlar un respirador artificial debe ser de tiempo real, ya que un retraso en su tiempo de respuesta no es aceptable.

Los sistemas de tiempo real pueden tener muchísimas aplicaciones y con el paso del tiempo y el desarrollo de nuevas tecnologías surgen, nuevos campos de utilización para estos sistemas.

1.9 Lenguajes y herramientas a utilizar

Durante el desarrollo de una aplicación la selección de las tecnologías y herramientas a utilizar es un elemento importante ya que constituyen la base para cualquier en el desarrollo del sistema. Este epígrafe tiene por objetivo realizar una definición completa de la metodología, lenguajes y herramientas que se emplearán en el desarrollo del sistema.

1.9.1 Metodología Utilizada

Las metodologías y estándares utilizados en un desarrollo de software proporcionan las guías para poder conocer todo el camino a recorrer desde antes de empezar la implementación, con lo cual se asegura la calidad del producto final al captar las mejores prácticas que el estado actual de la tecnología permite.

Rational Unified Process

El Proceso Unificado del Software (RUP), proporciona disciplinas o fases en las cuales se encuentran determinados artefactos que constituyen una guía para documentar e implementar de una manera fácil y eficiente el sistema. [20] También tiene como ventajas que proporciona un lenguaje común practicable en el desarrollo de cualquier software; como proceso es muy completo y en la actualidad constituye la metodología estándar más utilizada en el desarrollo informático a escala mundial.

1.9.2 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML) cuenta con una notación estándar y semánticas esenciales para el modelado de un sistema orientado a objetos. Es un lenguaje de propósito general cuyos artefactos además se pueden especificar y documentar para una mayor comprensión de los mismos y del sistema en general. También permite descubrir fallas en el diseño del sistema incluso antes de entrar en su implementación y realizar modificaciones de una forma sencilla. [21]

Dada la expresividad del mismo ya que cubre todas las vistas necesarias para desarrollar y luego desplegar un sistema, se escoge el mismo para llevar a cabo el modelado de la aplicación.

1.9.3 Visual Paradigm

El Visual Paradigm es una herramienta CASE profesional que emplea el UML como lenguaje de modelado y soporta el ciclo completo de vida del desarrollo de un software. Además es una herramienta

multiplataforma teniendo disponible una versión libre para la comunidad. También ayuda a lograr una construcción más rápida de aplicaciones de calidad, mejores y a un menor coste. Permite recrear todo tipo de diagramas, generar código desde diagramas y documentación. [22]

1.9.4 C++

El C++ es un lenguaje de programación de propósito general que proporciona capacidades de Programación Orientada a Objetos que ayudan mucho en el incremento de la productividad, calidad y reutilización del software. La extensibilidad es una de sus características más valiosas y en la actualidad es un lenguaje muy versátil y potente en el desarrollo de aplicaciones. [23]

1.9.5 Herramienta de desarrollo

Visual Studio 2008 es un Entorno de Desarrollo Integrado (IDE) que comprende herramientas, procesos, y guías para ayudar a los miembros de un equipo a mejorar sus conocimientos y trabajar juntos de forma más efectiva. También permite crear aplicaciones rápidamente, conectadas con la más alta calidad y con atractivas experiencias de usuario, así como la creación de soluciones multiplataforma adaptadas para funcionar con las diferentes versiones de .Net Framework. [24]

1.10 Conclusiones

A raíz del estudio realizado en el presente capítulo, se puede concluir que con el transcurso de los años y de acuerdo a las necesidades observadas en cada momento, se han desarrollado una amplia gama de técnicas empleadas para la animación facial, que han contribuido a alcanzar grandes logros no solo en esta rama sino en la computación en general. De ellas las más empleadas son los sistemas musculares, que superan muchas de las dificultades encontradas en las técnicas de morphing e interpolación.

Capítulo 2 "Propuesta de Solución"

2.1 Introducción

El capítulo presenta una visión del sistema, para ello se describe la propuesta de solución, representándose a través de un diagrama de clases el modelo de dominio en donde se enmarcará la aplicación propuesta. Se realiza un levantamiento de los requisitos funcionales y no funcionales con los que el sistema debe cumplir, agrupándose en Casos de Uso, además de realizar la descripción de los procesos que responden a las funcionalidades definidas.

2.2 Propuesta de solución

La solución que se propone para satisfacer la problemática planteada al inicio de esta investigación, se centra fundamentalmente en el diseño de una aplicación que sea capaz de simular los distintos estados de ánimo que se pueden expresar a través del rostro, brindando una mayor facilidad para personalizar el comportamiento de los pasajeros representados en el juego "Indicios Aduaneros", y tan importante en la detección de indicios sospechosos llevada a cabo en el mismo. También debe permitir reutilizar dichas animaciones, lográndose aplicar una misma animación a varios personajes. Los estados de ánimo pueden ser: expresar tristeza, alegría, rabia, miedo, sonreír, entre otros. En la aplicación se tendrán en cuenta además, algunos elementos específicos tales como el movimiento de las cejas, ojos, párpados y de la cabeza, con la finalidad de darle un mayor realismo a las animaciones.

Para la simulación de las expresiones el sistema hará uso del "*Expression Toolkit*", permitiendo la creación de estados de ánimo de una manera sencilla, conjuntamente con el motor gráfico Ogre, que permitirá definir entre otros elementos el adaptador de malla y el esqueleto del personaje. El control de los estados de ánimo estará dado por un controlador de estados, cuyo principal objetivo será el de activar y desactivar el estado deseado por el actor del sistema.

Tras el análisis realizado en el capítulo anterior, se escoge el FACS como técnica de animación facial a utilizar en la aplicación, dada las facilidades que brinda al permitir clasificar las expresiones faciales de acuerdo a los músculos que intervienen en cada acción, y también por el prestigio alcanzado siendo la base de muchos sistemas de animación facial.

La arquitectura del sistema estará basada en el patrón Modelo Vista Controlador (MVC) permitiendo separar los datos, la interfaz de usuario y la lógica del control en tres elementos diferentes. La

solución técnica estará regida por la metodología RUP haciendo uso de la herramienta CASE Visual Paradigm y UML como lenguaje de modelado. La construcción de la aplicación deberá realizarse empleando el lenguaje de programación C++ sobre la base del Entorno Integrado de Desarrollo (IDE) Visual Studio 2008.

2.3 Modelo de Dominio

Debido a que no se cuenta con una estructura con fronteras bien definidas de los procesos de negocio, se plantea elaborar un Modelo de Dominio en el que se describen las distintas entidades que participan así como las relaciones entre ellas.

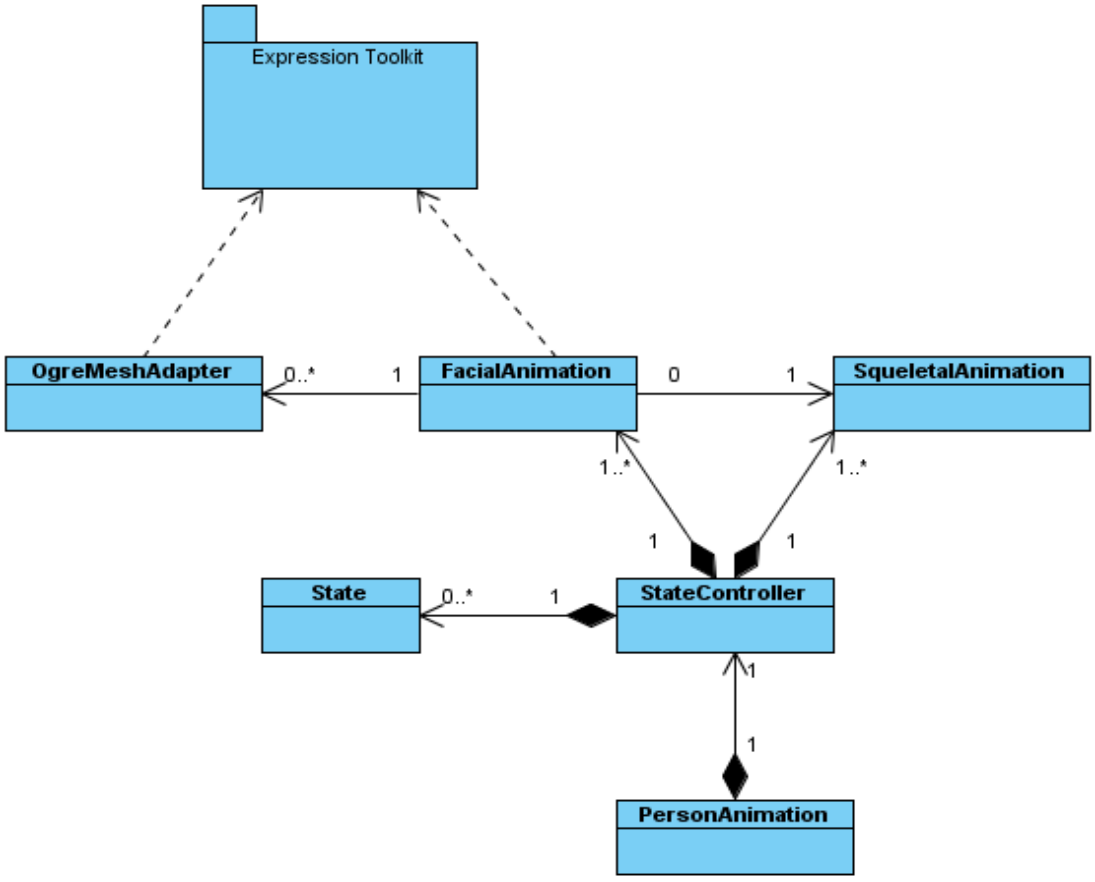

Figura 10. Diagrama de clases del dominio

2.1.1 Descripción de las clases del dominio

A continuación se explican los conceptos tratados en el modelo de dominio:

ExpEvent: contiene los tipos de eventos como tiempo de inicio y de culminación de un evento.
ExpHead: estructura que contiene todos los datos para el sistema de animación facial definido por *Expression Toolkit*.
ExpMeshAdapter: interfaz abstracta que permite la comunicación con una malla sin saber la forma en que se establece en memoria.
Expression: contiene los datos para los fotogramas claves que definen cada expresión.
FacialAnimation: encargada de controlar el adaptador de malla.
Morpher: almacena los datos necesarios para una animación específica.
OgreBone: define los huesos del esqueleto del personaje.
OgreMeshExpression: implementa todas las funcionalidades del adaptador de malla.
PersonAnimation: encargada de controlar las funcionalidades del sistema en general, asigna a cada componente la responsabilidad que le corresponde en dependencia de lo que se quiera lograr.
SkeletalAnimation: define el esqueleto del personaje a animar.
State: clase abstracta que define y activa un estado de ánimo.
StateController: encargada de controlar todos los estados de ánimo que existen.

2.4 Captura de requisitos

Una vez definido el modelo del dominio, se hace necesario definir lo que el sistema debe hacer, iniciándose así la captura de los requisitos que este debe cumplir. Esta acción incluye un conjunto de Casos de Uso que describen todas las interacciones que se prevé tenga lugar entre los usuarios y el software y contiene además requisitos no funcionales.

2.1.2 Requisitos funcionales

Seguidamente se enumeran las principales funcionalidades o capacidades que el sistema debe cumplir:

RF1. Cargar actor
- RF1.1 Cargar malla
- RF1.2 Cargar esqueleto
- RF1.3 Inicializar estados

RF2. Gestionar músculo
- RF2.1 Inicializar músculo

RF2.2 Contraer un músculo

RF2.3 Relajar músculo

RF2.4 Activar músculo

RF3. Activar estado de ánimo

RF4. Gestionar evento

RF4.1 Activar evento

RF4.2 Terminar evento

RF5. Gestionar parpadeo

RF5.1 Activar parpadeo

RF5.2 Desactivar parpadeo

RF6. Actualizar sistema

RF7. Terminar animación

RF8. Mirar a una posición

2.1.3 Requisitos no funcionales

A continuación se enumeran las cualidades que debe tener la aplicación para dar cumplimiento a los requisitos funcionales del sistema:

- *Requerimientos de Software:* compatible con los sistemas operativos Windows y Linux.
- *Requerimientos de Hardware:* requiere como mínimo un procesador Pentium 3 y 128Mb de RAM.
- *Requerimientos de Usabilidad:* el sistema debe proporcionar una interfaz sencilla y fácil de entender.

2.5 Descripción del Sistema

Actores del Sistema

Tabla 1: Actores del sistema

Actores	Justificación
Motor de Juego	Sistema con la responsabilidad de interactuar con las configuraciones de animaciones faciales existentes, mediante la activación de estas, la gestión de parpadeo y de músculos, además de la administración de los

	cambios que ocurran en el sistema.

Casos de uso del Sistema

Las acciones que el sistema debe permitir llevar a cabo son:

CU1. Inicializar sistema de animación

CU2. Actualizar sistema

CU3. Terminar animación

CU4. Activar estado de ánimo

CU5. Gestionar parpadeo

CU6. Mirar a una posición

CU7. Mirar a una posición random

CU8. Activar estado de ánimo automático

Diagrama de Casos de Uso del Sistema

El diagrama de Casos de Uso constituye una visión general que se ha identificado para satisfacer los requerimientos funcionales del sistema.

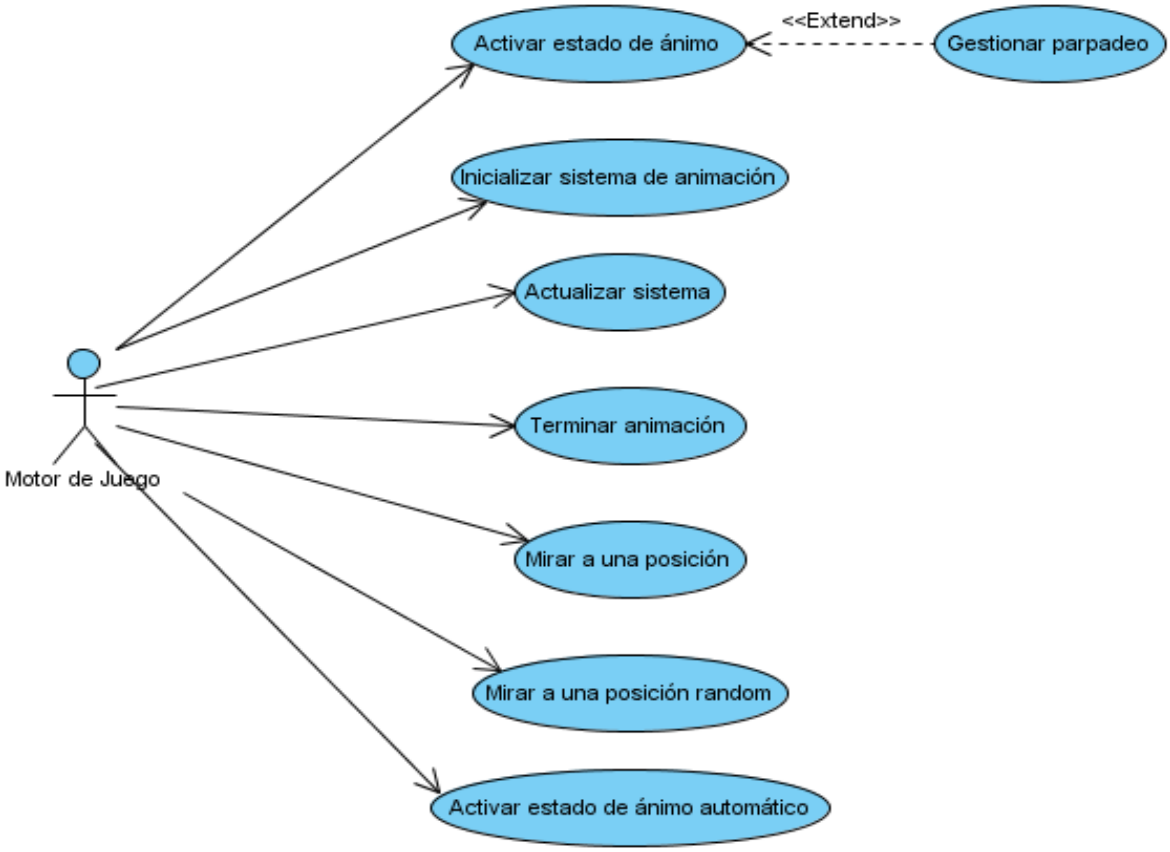

Figura 11. Diagrama de Casos de Uso del sistema

2.6 Especificación de los Casos de Uso

Para entender bien las funcionalidades asociadas a cada caso de uso, no es suficiente con la representación gráfica del diagrama de casos de uso. La descripción de estos es fundamental para lograr un mayor entendimiento entre analistas y desarrolladores. A continuación se muestran cada una de las descripciones de los casos de uso definidos en el sistema:

Tabla 2: Descripción del Caso de Uso "Inicializar sistema de animación"

Caso de Uso	Inicializar sistema de animación
Actor(es)	Motor de Juego
Propósito	Preparar el sistema para iniciar el proceso de animación

Resumen	El caso de uso comienza cuando el Motor de Juego solicita que se inicie el sistema de animación. El sistema inicializa al controlador de estados que a su vez manda a inicializar los estados de ánimo, las mallas y el esqueleto.
Referencia	RF1
Precondiciones	El fichero que guarda las direcciones de cada uno de los elementos a inicializarse debe existir.
Pos-condiciones	El sistema fue inicializado y la malla, el esqueleto y los estados fueron cargados.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El actor solicita inicializar el sistema de animación facial.	2. Busca el fichero de configuración que posee las direcciones físicas de los elementos que intervienen en la animación. 3. Inicializa los datos referentes a la malla, el esqueleto y los estados.
Prioridad	Crítico

Tabla 3: Descripción del Caso de Uso "Actualizar sistema"

Caso de Uso	Actualizar sistema
Actor(es)	Motor de Juego
Propósito	Actualizar todas las variables que intervienen en el proceso de animación.
Resumen	El caso de uso inicia cuando el Motor de Juego solicita que se actualicen los elementos que intervienen en el proceso de animación: mallas, esqueleto y estados.
Referencia	RF6
Precondiciones	Debe haberse inicializado el sistema de animación.
Pos-condiciones	Se actualizan todos los elementos que intervienen en la animación.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El actor solicita la actualización de los	2. Actualiza los elementos de animación: la malla,

elementos que intervienen en el proceso de animación.	el esqueleto y los estados. 3. Muestra las animaciones con la nueva actualización realizada por el usuario.
Prioridad	Crítico

Tabla 4: Descripción del Caso de Uso "Terminar animación"

Caso de Uso	Terminar animación
Actor(es)	Motor de Juego
Propósito	Terminar el proceso de animación.
Resumen	El caso de uso comienza cuando el Motor de Juego solicita terminar la animación. El sistema finaliza el controlador de estados y este a su vez destruye las mallas, el esqueleto y los estados.
Referencia	RF7
Precondiciones	Debe haberse inicializado el sistema de animación.
Pos-condiciones	Concluye el proceso de animación.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El actor selecciona terminar animación.	2. Destruye las mallas, el esqueleto y los estados. 3. Termina la animación.
Prioridad	Crítico

Tabla 5: Descripción del Caso de Uso "Activar estado de ánimo"

Caso de Uso	Activar estado de ánimo
Actor(es)	Motor de Juego
Propósito	Activar un estado de ánimo
Resumen	El caso de uso inicia cuando el actor desea activar un estado de ánimo existente en el sistema para un personaje correspondiente.
Referencia	RF4, CU5 (extend)
Precondiciones	Debe haberse inicializado el sistema de animación.
Pos-condiciones	Se activó un estado de ánimo.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El actor solicita activar un estado de ánimo.	2. Muestra los estados de ánimos existentes.
3. Selecciona el estado que desea activar para un personaje determinado.	4. Activa el estado de ánimo seleccionado para el personaje en cuestión. 5. Muestra la animación seleccionada en el personaje correspondiente.
Prioridad	Crítico

Tabla 6: Descripción del Caso de Uso "Gestionar parpadeo"

Caso de Uso	Gestionar parpadeo
Actor(es)	Motor de Juego
Propósito	Permitir las operaciones específicas del parpadeo
Resumen	El caso de uso se inicia cuando el actor solicita que el personaje realice o no la acción de parpadear. En caso de que el actor seleccione la acción de parpadear el sistema activa esta acción. En caso contrario desactiva el parpadeo.
Referencia	RF5
Precondiciones	Debe haberse inicializado el sistema de animación.
Pos condiciones	Se activó la acción de parpadear para un personaje determinado.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El actor decide incorporar la opción de parpadeo a la animación facial del personaje determinado.	2. Brinda la posibilidad de realizar las acciones: - Activar parpadeo - Desactivar parpadeo en caso de que esté activado.
3. Selecciona la opción de activar parpadeo para el personaje en cuestión.	4. Muestra la activación de la acción de parpadear. 5. Muestra la animación del parpadeo para el personaje correspondiente.

Flujo Alterno “Desactivar parpadeo”	
3.1 Selecciona la opción de desactivar parpadeo para el personaje en cuestión.	4.1 Muestra la desactivación de la acción de parpadear. 5.1 Muestra el personaje correspondiente sin la animación de parpadeo.
Prioridad	Secundario

Tabla 7: Descripción del Caso de Uso “Mirar a una posición”

Caso de Uso	Mirar a una posición
Actor(es)	Motor de Juego
Propósito	Permitir que el personaje mire hacia una posición específica.
Resumen	El caso de uso se inicia cuando el actor selecciona la opción de que el personaje mire hacia una posición específica. El personaje dirige la mirada en dirección a la posición del mouse.
Referencia	RF8
Precondiciones	Debe haberse inicializado el sistema de animación.
Pos-condiciones	El personaje dirigió la mirada hacia la posición determinada.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El actor decide incorporar la opción de mirar a una posición específica en la animación facial del personaje determinado.	2. Brinda la posibilidad de realizar las acciones: - Activar mirar a una posición específica. - Desactivar mirar a una posición específica en caso de que esté activada.
3. El actor selecciona activar la opción de mirar hacia una posición específica para animar en el personaje determinado.	4. Muestra la opción activada. 5. El personaje mira en dirección a la posición determinada.
Flujo Alterno “Desactivar parpadeo”	
3.1 El actor selecciona desactivar la opción de mirar hacia una posición específica para aplicar en el personaje determinado.	4.1 El sistema muestra la opción desactivada. 5.1 El personaje en cuestión permanece con la vista inmóvil.

Prioridad	Secundario

Tabla 8: Descripción del Caso de Uso "Mirar a una posición random"

Caso de Uso	Mirar a una posición random	
Actor(es)	Motor de Juego	
Propósito	Permitir que el personaje mire hacia una posición aleatoria.	
Resumen	El caso de uso se inicia cuando el actor selecciona la opción de que el personaje mire hacia una posición aleatoria. El personaje dirige la mirada a una dirección seleccionada por el sistema.	
Referencia	RF8	
Precondiciones	Debe haberse inicializado el sistema de animación.	
Pos-condiciones	El personaje dirigió la mirada hacia la posición determinada por el sistema.	
Curso Normal de los Eventos		
Acciones del Actor		**Respuesta del Sistema**
1. El actor decide incorporar la opción de mirar a una posición aleatoria en la animación facial del personaje determinado.		2. Brinda la posibilidad de realizar las acciones: - Activar mirar a una posición aleatoria. - Desactivar mirar a una posición aleatoria en caso de que esté activada.
3. El actor selecciona activar la opción de mirar hacia una posición aleatoria para animar en el personaje determinado.		4. Muestra la opción activada. 5. El personaje mira en dirección a una posición aleatoria.
Prioridad	Secundario	

Tabla 9: Descripción del Caso de Uso "Activar estado de ánimo automático"

Caso de Uso	Activar estado de ánimo automático
Actor(es)	Motor de Juego
Propósito	Activar los estados de ánimo de forma automática.
Resumen	El caso de uso inicia cuando el Motor de Juego solicita que se activen los estados de ánimo de forma automática.
Referencia	RF3

Precondiciones	Debe haberse inicializado el sistema de animación.
Pos-condiciones	Los estados de ánimo se activaron de forma automática.
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El actor decide incorporar la opción de activar estados de ánimo automáticos en la animación facial del personaje determinado.	2. Brinda la posibilidad de realizar las acciones: - Activar mirar a una posición aleatoria. - Desactivar mirar a una posición aleatoria en caso de que esté activada.
3. El actor selecciona activar la opción de mirar hacia una posición aleatoria para animar en el personaje determinado.	4. Muestra la opción activada. 5. El personaje mira en dirección a una posición aleatoria.
Prioridad	Crítico

2.7 Conclusiones

En este capítulo se expuso en detalle la solución que se propone para dar respuesta a la problemática planteada. Se obtuvo una primera visión de la aplicación que se desea desarrollar a través del Modelo de Dominio, definiéndose además los requisitos funcionales y no funcionales de la misma, se confeccionó el diagrama de casos de uso del sistema y se realizó una descripción detallada de cada caso de uso, facilitando así la comprensión del sistema.

Capítulo 3 "Análisis y Diseño del Sistema"

3.1 Introducción

El análisis y el diseño constituyen partes fundamentales dentro del proceso de desarrollo de software. Es muy importante realizar un esbozo preliminar del comportamiento que se desea obtener a partir de los elementos de modelación del sistema y concretarlo en el modelo que será implementado posteriormente.

En el presente capítulo se llevará a cabo el análisis y el diseño del sistema, representado mediante los diagramas de clases del análisis y del diseño, y de una forma más profunda en los diagramas de secuencia, logrando así modelar la aplicación que se desea desarrollar.

3.2 Modelo de Análisis

El Modelo de Análisis se emplea fundamentalmente para representar la estructura global del sistema y constituye un primer intento por definir los conceptos claves que describen el sistema.

3.2.1 Diagramas de clases del análisis

Debido a que algunos de los Casos de Uso del sistema tienen un comportamiento similar en lo que se refiere a diagramas de clases del análisis, se modela un mismo diagrama para dichos Casos de Uso, especificando en cada diagrama o figura cuáles son los que corresponden al mismo.

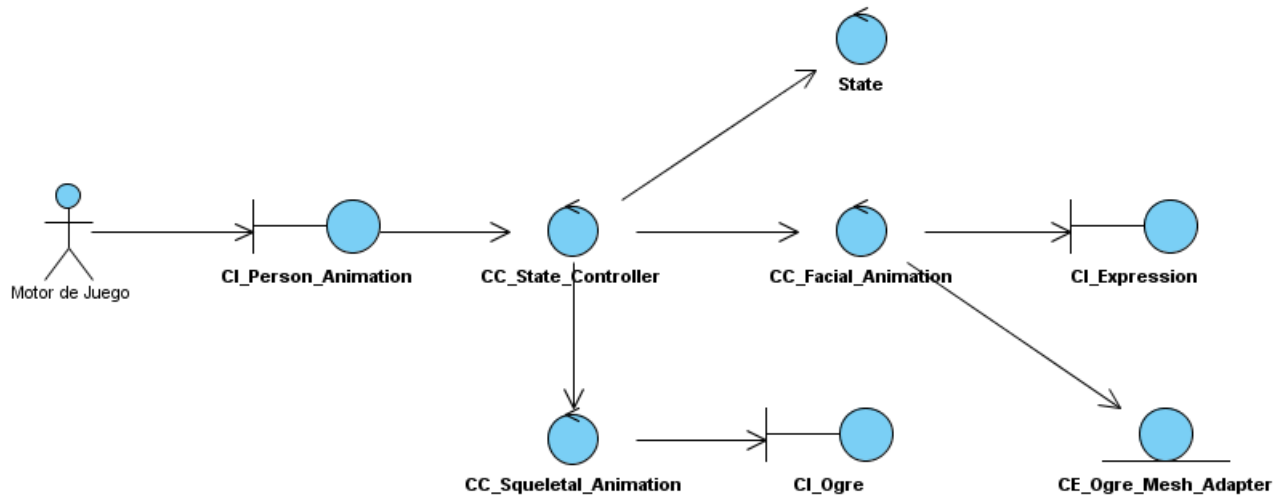

Figura 12.

Diagrama de clases del análisis *"CU Inicializar sistema"*

Diagrama de clases del análisis *"CU Actualizar sistema"*

Diagrama de clases del análisis *"CU Terminar animación"*

Diagrama de clases del análisis *"CU Mirar a una posición"*

Diagrama de clases del análisis *"CU Mirar a una posición random"*

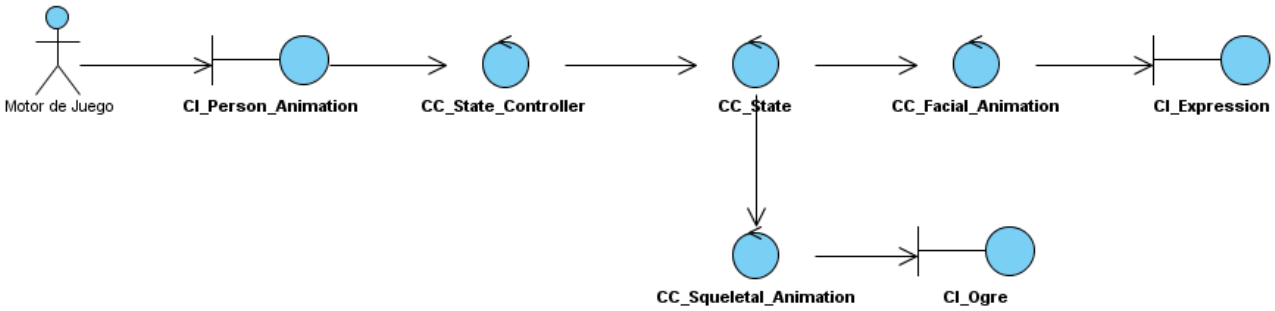

Figura 13.

Diagrama de clases del análisis *"CU Activar estado de ánimo"*

Diagrama de clases del análisis *"CU Activar estado de ánimo automático"*

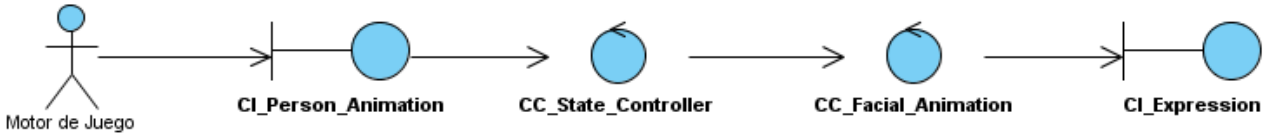

Figura 14. Diagrama de clases del análisis *"CU Gestionar parpadeo"*

3.2.2 Diagramas de interacción

Los diagramas de interacción hacen hincapié en las interacciones entre objetos. Tienen diferentes formas, basadas todas en una misma información subyacente pero resaltando cada punto de vista de la misma. En este caso dicha información será modelada a través de diagramas de secuencias.

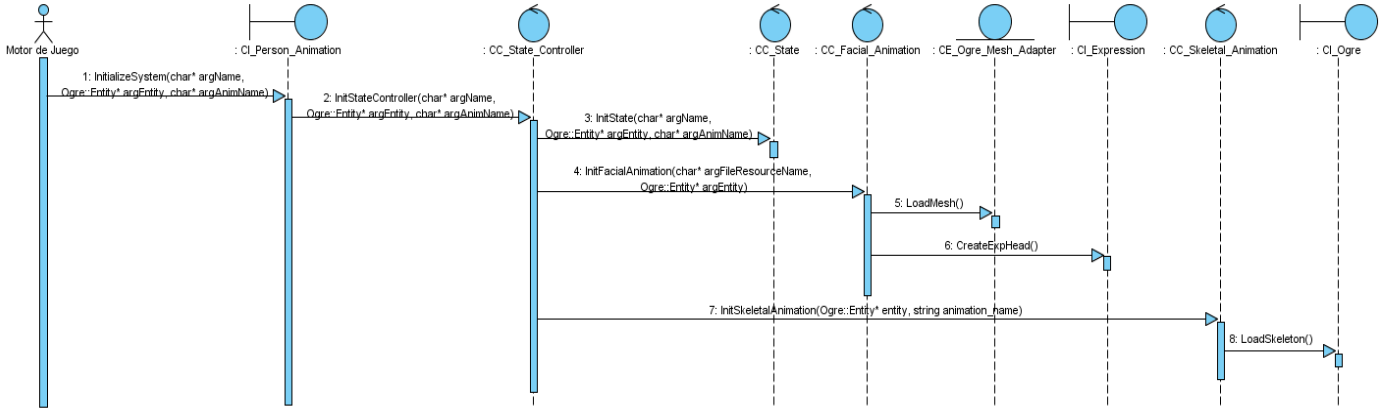

Figura 15. Diagrama de secuencia *"CU Inicializar sistema"*

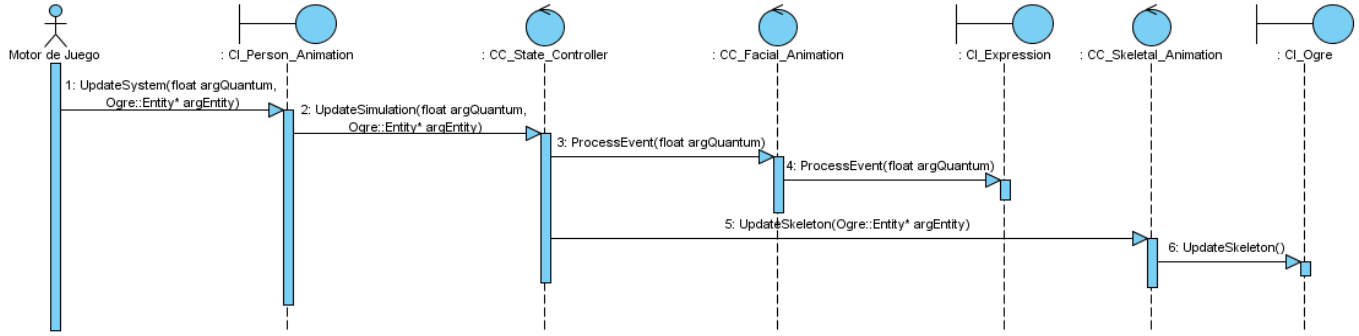

Figura 16. Diagrama de secuencia *"CU Actualizar sistema"*

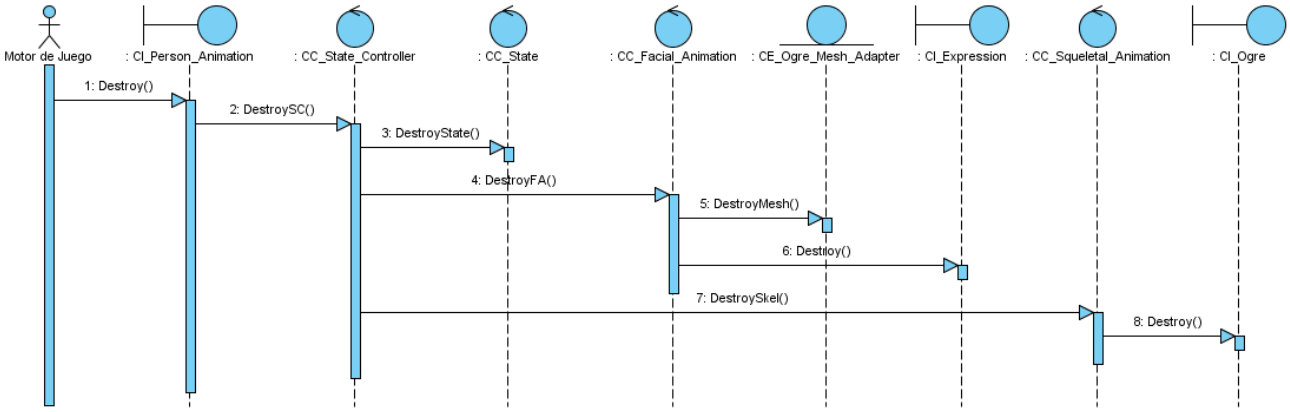

Figura 17. Diagrama de secuencia *"CU Terminar animación"*

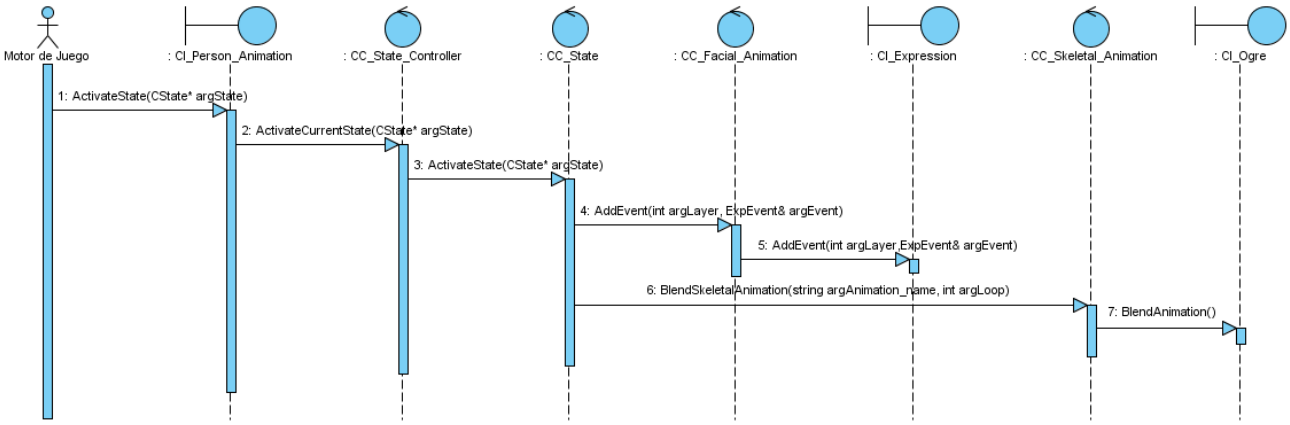

Figura 18. Diagrama de secuencia *"CU Activar estado de ánimo"*

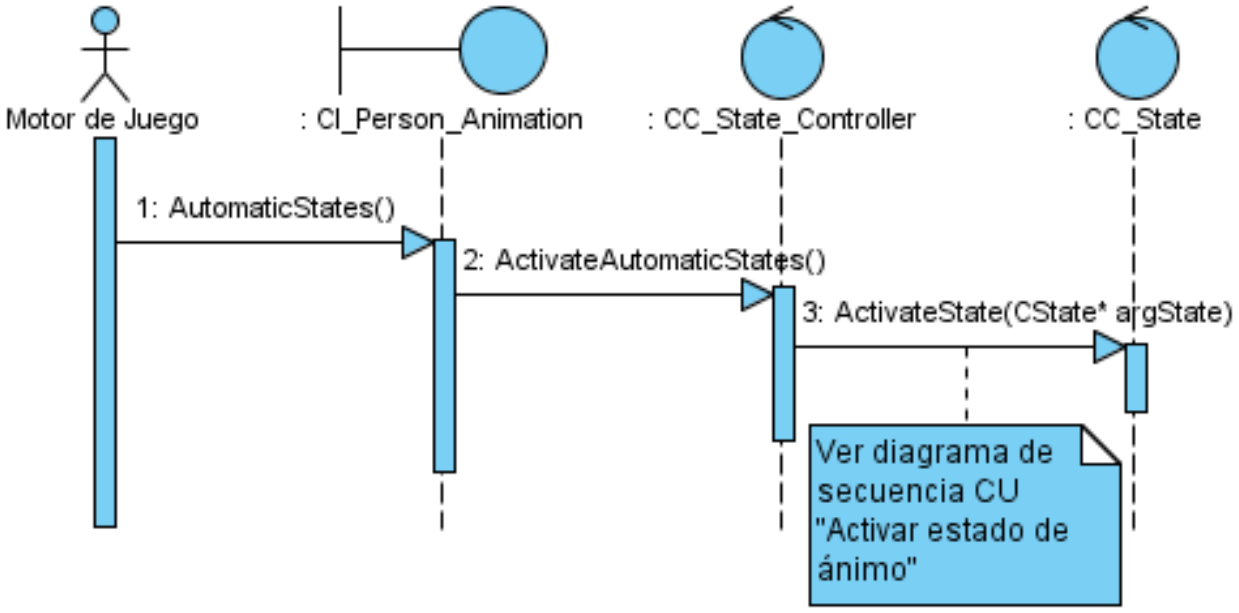

Figura 19. Diagrama de secuencia *"CU Activar estado de ánimo automático"*

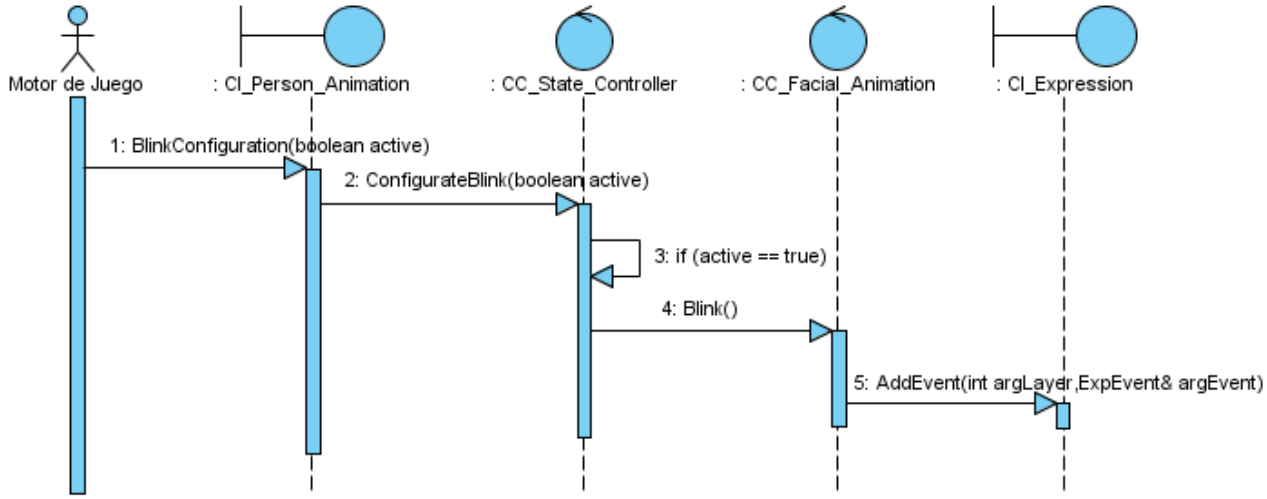

Figura 20. Diagrama de secuencia *"CU Gestionar parpadeo"*

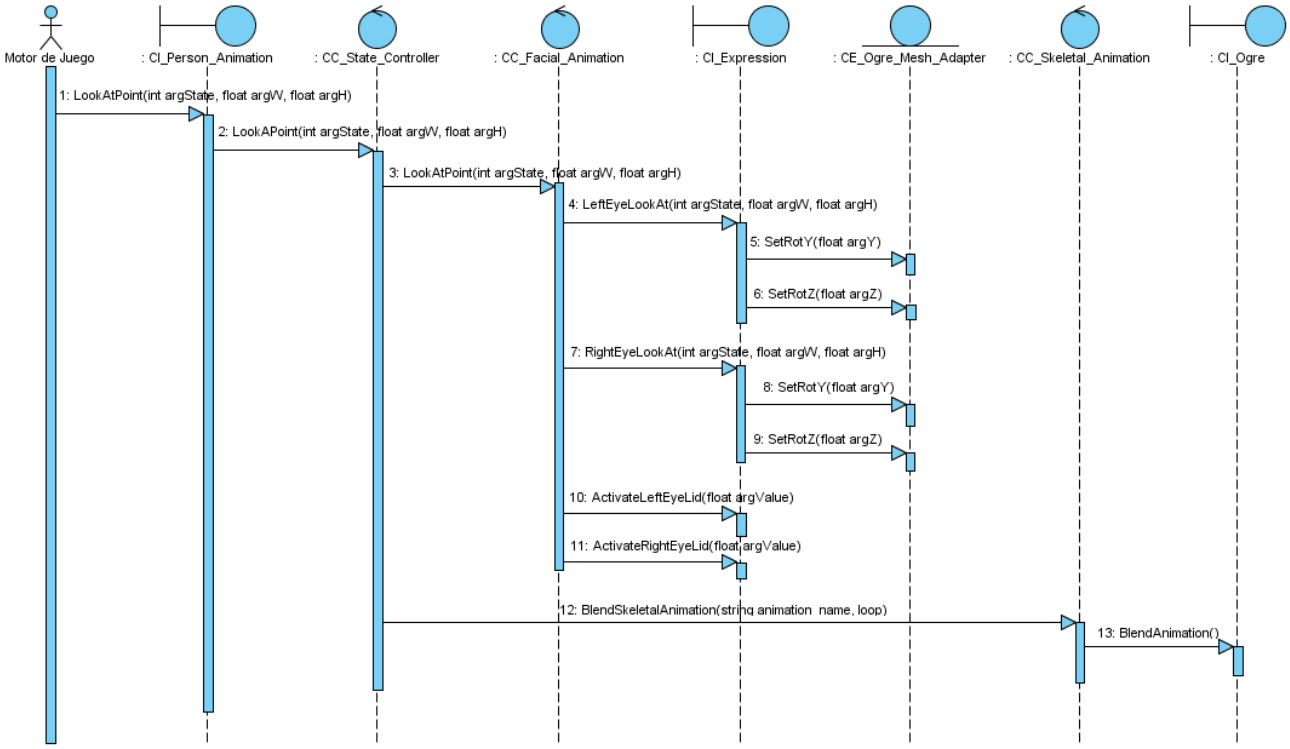

Figura 21. Diagrama de secuencia *"CU Mirar a una posición"*

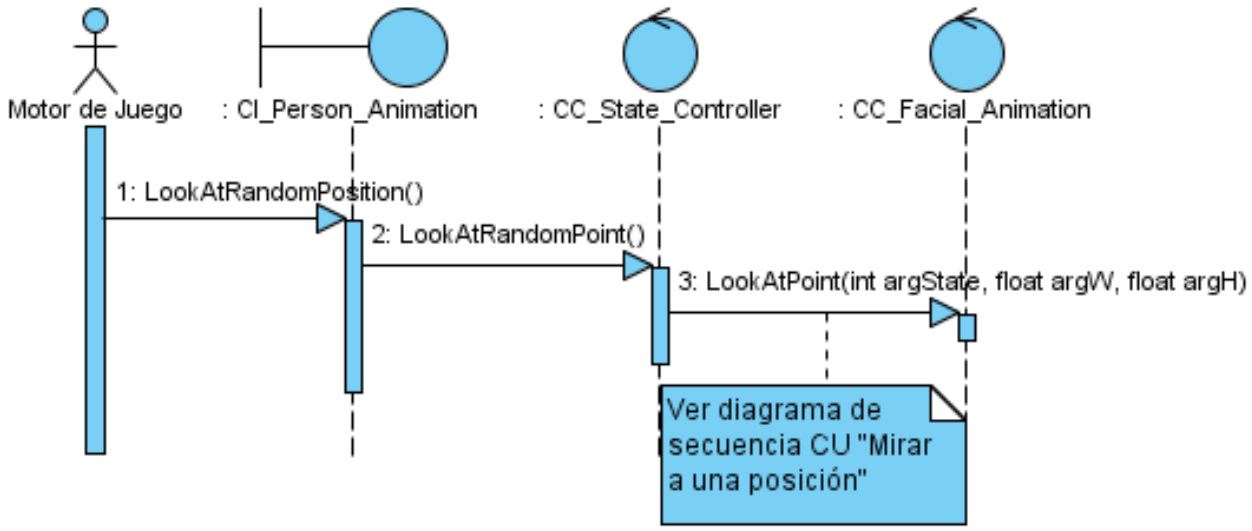

Figura 22. Diagrama de secuencia *"CU Mirar a una posición random"*

3.3 Patrones de diseño

Los patrones constituyen una descripción de clases y objetos que se comunican entre sí, adaptada para resolver un problema general de diseño en un contexto particular. Dentro de ellos se encuentran los patrones GRASP y los GOF. Haciendo uso de estos dos grupos seguidamente se describen los patrones empleados en el diseño de la aplicación enfocados en el uso que se les da en el sistema:

Fachada (GOF): Representado a través de la clase CPersonAnimation que brinda una interfaz unificada y sencilla mediante la cual se puede interactuar con el sistema.

Agente Remoto (GOF): La clase *CExpression* será la encargada de relacionarse con el sistema *Expression Toolkit* y a su vez la clase *COgre* será la que interactúe con el sistema *Ogre*, representándose así la relación con sistemas externos.

Controlador (GRASP): Las clases *CStateController, CFacialAnimation y CSkeletalAnimation* serán las encargadas de controlar los estados, las mallas y animaciones, los huesos y músculos respectivamente.

Polimorfismo (GRASP): De la clase *CState* heredan los distintos estados de ánimo y dado que la misma es abstracta, cada una las hijas redefinirá las funcionalidades de la padre de acuerdo a su comportamiento individual.

3.4 Modelo de Diseño

El Modelo de Diseño permite transformar los requerimientos en un diseño de software, permitiendo la definición de una arquitectura robusta para que el futuro sistema sea adaptable a un ambiente de implementación dado.

3.4.1 Diagrama de Clases del Diseño

En el Diagrama de Clases del Diseño se describe la estructura de la aplicación propuesta mostrando sus clases, atributos y las relaciones entre ellos.

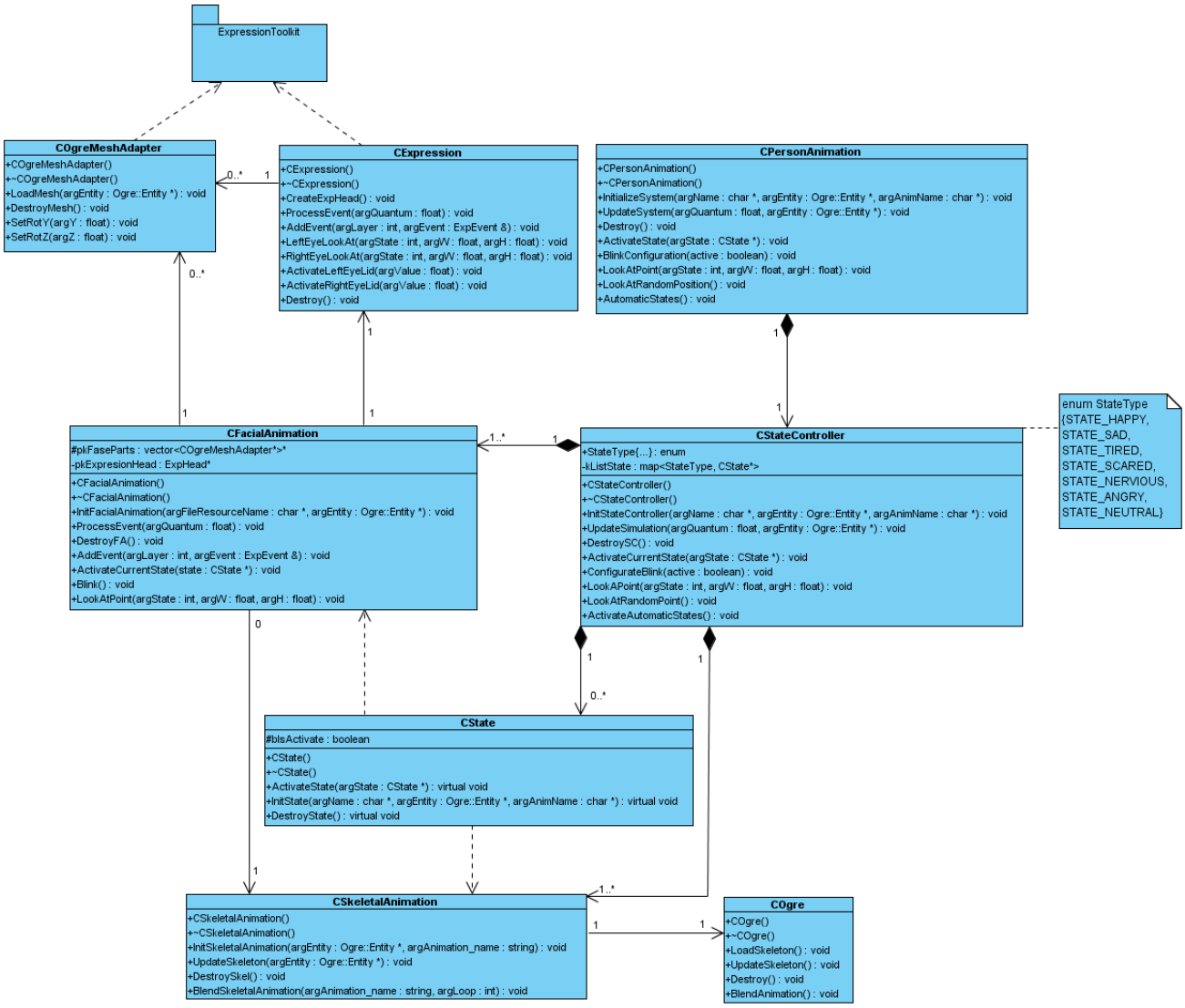

Figura 23. Diagrama de Clases del Diseño

3.4.2 Descripción de las clases del diseño

Durante esta actividad se realiza una descripción minuciosa de las clases que componen el sistema, sus responsabilidades y la funcionalidad de cada uno de los métodos que las componen.

Tabla 10: Descripción de la Clase CPersonAnimation

Nombre:	CPersonAnimation
Tipo de clase:	Interfaz

Para cada responsabilidad	
Nombre:	CPersonAnimation()
Descripción:	Constructor de la clase
Nombre:	~CPersonAnimation()
Descripción:	Destructor de la clase
Nombre:	InitializeSystem(char* argName, Ogre::Entity* argEntity, char* argAnimName)
Descripción:	Inicializa los elementos que intervienen en la animación
Nombre:	UpdateSystem(float argQuantum, Ogre::Entity* argEntity)
Descripción:	Actualiza el sistema de animación
Nombre:	Destroy()
Descripción:	Elimina los elementos cargados para la animación
Nombre:	ActivateState(CState* argState)
Descripción:	Activa el estado especificado por parámetro
Nombre:	BlinkConfiguration(boolean active)
Descripción:	Activa la acción de parpadear
Nombre:	LookAtPoint(int argState, float argW, float argH)
Descripción:	Encargado de que el personaje mire en dirección a una posición específica
Nombre:	LookAtRandomPosition()
Descripción:	Encargado de que el personaje mire en cualquier dirección
Nombre:	AutomaticStates()
Descripción:	Activa los estados de ánimo de forma automática

Tabla 11: Descripción de la Clase CStateController

Nombre:	CStateController
Tipo de clase:	Controladora
Atributo:	**Tipo:**
StateType {STATE_HAPPY,STATE_SAD,STATE_TIRED,STATE_SCARED,STATE_NERVIOUS,STATE_ANGRY,STATE_NEUTRAL,}	enum

kListState	map<StateType,CState*>
Para cada responsabilidad	
Nombre:	CStateController()
Descripción:	Constructor de la clase
Nombre:	~CStateController()
Descripción:	Destructor de la clase
Nombre:	InitStateController(char* argName, Ogre::Entity* argEntity, char* argAnimName)
Descripción:	Inicializa el controlador de estado
Nombre:	UpdateSimulation(float argQuantum, Ogre::Entity* argEntity)
Descripción:	Actualiza los valores de los elementos que intervienen en la simulación
Nombre:	DestroySC()
Descripción:	Elimina los elementos que intervienen en la animación
Nombre:	ActivateCurrentState(CState* argState)
Descripción:	Activa el estado de ánimo especificado
Nombre:	ConfigurateBlink(boolean active)
Descripción:	Configura la acción de parpadear
Nombre:	LookAPoint(int argState, float argW, float argH)
Descripción:	Encargado de que el personaje mire en dirección a una posición específica
Nombre:	LookAtRandomPoint()
Descripción:	Mirar en cualquier dirección
Nombre:	ActivateAutomaticStates()
Descripción:	Activa los estados de ánimo de forma automática

Tabla 12: Descripción de la Clase CFacialAnimation

Nombre:	CFacialAnimation
Tipo de clase:	Controladora
Atributo:	**Tipo:**
pkFaseParts	vector<COgreMeshAdapter*>*
pkExpresionHead	ExpHead*

Para cada responsabilidad	
Nombre:	CFacialAnimation()
Descripción:	Constructor de la clase
Nombre:	~CFacialAnimation()
Descripción:	Destructor de la clase
Nombre:	InitFacialAnimation(char* argFileResourceName, Ogre::Entity* argEntity)
Descripción:	Inicializa las animaciones de la cara
Nombre:	ProcessEvent(float argQuantum)
Descripción:	Procesa todos los eventos después de la inicialización
Nombre:	DestroyFA()
Descripción:	Destruye los elementos que intervienen en la animación
Nombre:	AddEvent(int argLayer, ExpEvent& argEvent)
Descripción:	Adiciona un evento en el layer especificado
Nombre:	ActivateCurrentState(CState* argState)
Descripción:	Activar el estado de ánimo especificado
Nombre:	Blink()
Descripción:	Activa el parpadeo
Nombre:	LookAtPoint(int argState, float argW, float argH)
Descripción:	Mirar en dirección de la posición especificada

Tabla 13: Descripción de la Clase CSkeletalAnimation

Nombre:	CSkeletalAnimation
Tipo de clase:	Controladora
Para cada responsabilidad	
Nombre:	CSkeletalAnimation()
Descripción:	Constructor de la clase
Nombre:	~CSkeletalAnimation
Descripción:	Destructor de la clase
Nombre:	InitSkeletalAnimation(Ogre::Entity* argEntity, string argAnimation_name)

Descripción:	Inicializa el esqueleto
Nombre:	UpdateSkeleton(Ogre::Entity* argEntity)
Descripción:	Actualiza el esqueleto
Nombre:	DestroySkel()
Descripción:	Destruye el esqueleto
Nombre:	BlendSkeletalAnimation(string argAnimation_name, int argLoop)
Descripción:	Activa la animación del esqueleto

Tabla 14: Descripción de la Clase CState

Nombre:	CState
Tipo de clase:	Controladora
Atributo:	**Tipo:**
blsActivate	boolean
Para cada responsabilidad	
Nombre:	CState()
Descripción:	Constructor de la clase
Nombre:	~CState()
Descripción:	Destructor de la clase
Nombre:	ActivateState(boolean argState)
Descripción:	Activa un estado
Nombre:	InitState(char* argName, Ogre::Entity* argEntity, char* argAnimName)
Descripción:	Inicializa un estado
Nombre:	DestroyState()
Descripción:	Destruye un estado de ánimo

Tabla 15: Descripción de la Clase COgreMeshAdapter

Nombre:	COgreMeshAdapter
Tipo de clase:	Entidad

Atributo:	Tipo:
bIsLoaded	bool
cName	char*
cParentName	char*
argMesh	Ogre::Mesh*
Para cada responsabilidad	
Nombre:	COgreMeshAdapter()
Descripción:	Constructor de la clase
Nombre:	~COgreMeshAdapter()
Descripción:	Destructor de la clase
Nombre:	LoadMeshData(Ogre::Entity* argEntity)
Descripción:	Carga la malla
Nombre:	DestroyMesh()
Descripción:	Destruye la malla
Nombre:	SetRotY(float argY)
Descripción:	Cambia el eje *y* de rotación de la malla
Nombre:	SetRotZ(float argZ)
Descripción:	Cambia el eje *z* de rotación de la malla

Tabla 16: Descripción de la Clase CExpression

Nombre:	CExpression
Tipo de clase:	Interfaz que interrelaciona el sistema con la herramienta "*Expression*"
Para cada responsabilidad	
Nombre:	CExpression()
Descripción:	Constructor de la clase
Nombre:	~CExpression()
Descripción:	Destructor de la clase
Nombre:	CreateExpHead()
Descripción:	Encargado de llamar al constructor de la controladora ExpHead perteneciente al

	Sistema Expresión
Nombre:	ProcessEvent(float quantum)
Descripción:	Actualiza los eventos del sistema según el quantum de tiempo indicado
Nombre:	AddEvent(int layer, ExpEvent & event)
Descripción:	Adiciona el evento especificado en el layer indicado
Nombre:	LeftEyeLookAt(int argState, float argW, float argH)
Descripción:	Activa el ojo izquierdo
Nombre:	RightEyeLookAt(int argState, float argW, float argH)
Descripción:	Activa el ojo derecho
Nombre:	ActivateLeftEyeLid(float argValue)
Descripción:	Activa el párpado izquierdo
Nombre:	ActivateRightEyeLid(float argValue)
Descripción:	Activa el párpado derecho
Nombre:	Destroy()
Descripción:	Destruye las animaciones

Tabla 17: Descripción de la Clase COgre

Nombre:	COgre
Tipo de clase:	Interfaz que interrelaciona el sistema con el motor gráfico Ogre
Para cada responsabilidad	
Nombre:	COgre()
Descripción:	Constructor de la clase
Nombre:	~COgre()
Descripción:	Destructor de la clase
Nombre:	LoadSkeleton()
Descripción:	Carga el esqueleto
Nombre:	UpdateSkeleton()
Descripción:	Actualiza el esqueleto
Nombre:	Destroy()

Descripción:	Destruye el esqueleto
Nombre:	BlendAnimation()
Descripción:	Activa la animación

3.5 Conclusiones

Al concluir este capítulo quedó concebido de una forma bien detallada el diseño completo del sistema, se planteó la secuencia de pasos a seguir en cada acción representada mediante diagramas de interacción y traducida a través de mensajes entre clases. También quedaron definidas las clases que intervendrán en la implementación de la aplicación, así como una breve descripción de los métodos que las conforman.

Capítulo 4 "Implementación del Sistema"

4.1 Introducción

En el presente capítulo se explican determinados elementos relacionados directamente con la implementación del sistema. Se expone el estándar de codificación o reglas específicas del lenguaje empleado, así como los diagramas de componentes y de despliegue; el primero con la finalidad de dar una visión de la organización del código, y el segundo para modelar cómo y dónde debe ser desplegado el sistema.

4.2 Estándares de codificación

El estándar de codificación comprende todos los aspectos de la generación del código necesarios para una mejor comprensión del mismo y permite una mayor desenvoltura en el uso de las funcionalidades implementadas.

- ✓ **Constantes**

Las constantes se nombran con mayúsculas, utilizándose "_" para separar las palabras.

Ejemplo: const int MY_CONST_ZERO = 0;

- ✓ **Enumerados**

Para los enumerados se utilizan las iniciales del nombre del enumerado en las constantes y se escriben utilizando mayúsculas.

Ejemplo: StateType

```
{
STATE_HAPPY,
STATE_SAD,

STATE_TIRED,
};
```

- ✓ **Listas e iteradores STD**

Para los tipos de datos utilizados de la librería estándar de C++ (*vector*, *map*, *multimap*, etc.), se utilizan los prefijos *List, Map y MultiMap* según la estructura, así como el prefijo *Iter* para los iteradores. Además el nombre lleva el tipo de dato a almacenar en la estructura en cuestión:

Ejemplo 1: vector<COgreMeshAdapter*>* pkBodyParts;

Ejemplo 2: map<StateType,CState*> *List*StatebyType;

- ✓ **Declaración de variables**

Los nombres de las variables comenzarán con un identificador del tipo de dato al que corresponden y en caso de ser argumentos de algún método, se les antepone el prefijo "*arg*".

- o **Tipos simples**

Ejemplo:

```
bool bIsLoaded;
int iVertextCount;
float fName;
char cName;
char* cName;  //arreglo de caracteres
```

- o **Instancias de tipos creados**

Ejemplo:

```
ListStatebyType   kListState;   //objeto de una clase
CState*   pkCurrentState;   //puntero a objeto
IMyInterface* pIName;  //puntero interfaces
```

- ✓ **Clases**

Se utiliza el indicador "*C*" para indicar que es una clase.

Ejemplo: class *C*ClassName;

- ✓ **Interfaces**

Se utiliza el indicador "*I*" para indicar que es una interfaz.

*Ejemplo: I*MyInterfaceName

- ✓ **Métodos**

En el caso de los métodos, se les antepone el identificador del tipo de dato de devolución, y en caso de no tenerlo (void), no se les antepone nada. Los constructores y destructores, como lo exigen los compiladores, llevan el nombre de la clase y son los únicos que omitirán el tipo de dato de retorno.

- o **Constructor y destructor**

Ejemplo: CClassName (bool arg_bVarName, float& arg_fVarName);
~CClassName ();

- o **Funciones**

Ejemplo: bool bFunction1 (…);
int* piFunction2 (…);
CClassName* pkFunction3 (…);

- o **Procedimientos**

Ejemplo: void Procedure4 (…);

4.3 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones, siendo los componentes todos los tipos de elementos de software que intervienen en el desarrollo de la aplicación. A continuación se muestra el diagrama de componentes donde se observa la distribución de las clases del sistema:

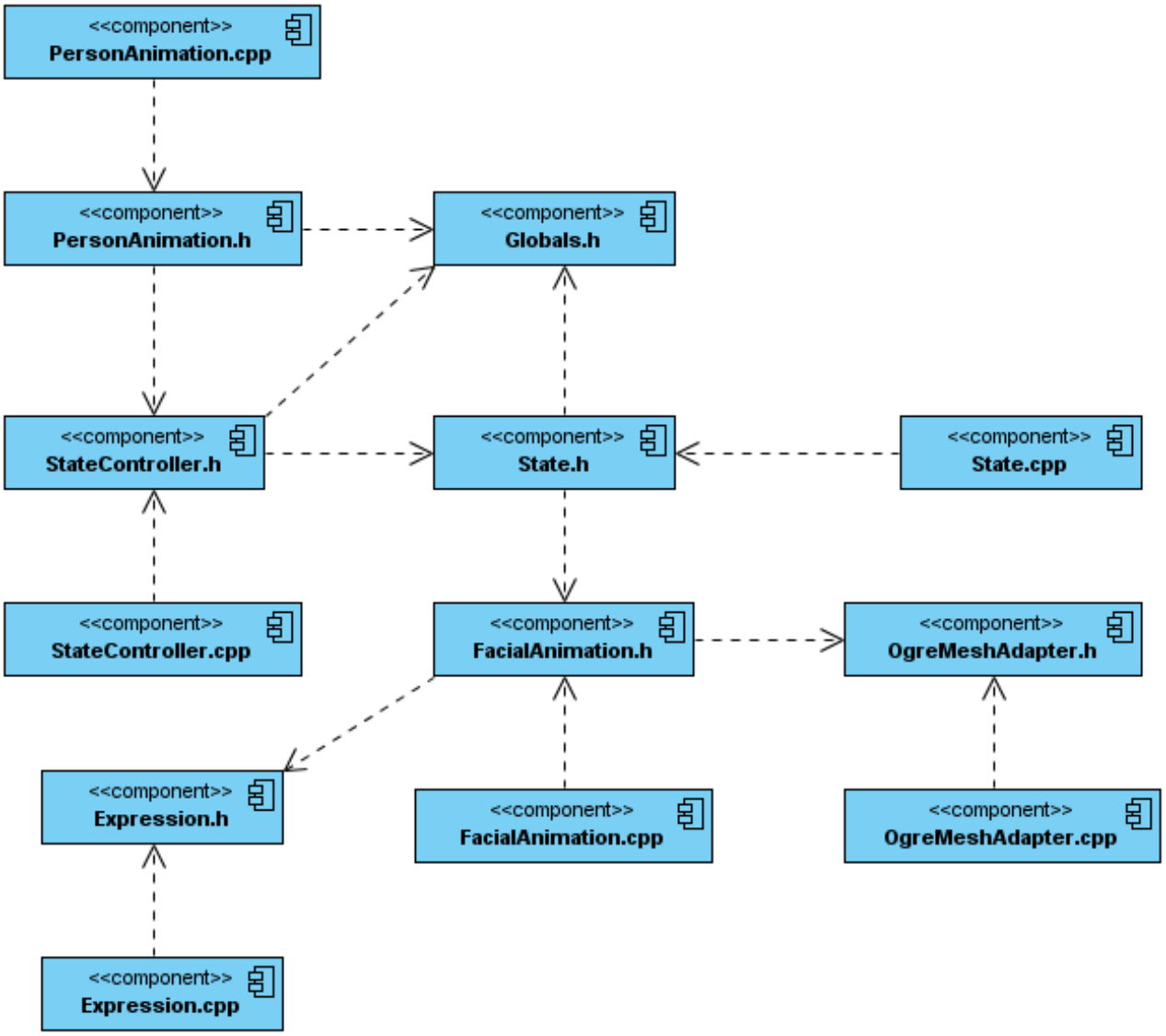

Figura 23: Diagrama de componentes

4.4 Diagrama de Despliegue

El Diagrama de Despliegue muestra la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. En el caso del sistema en cuestión las relaciones físicas finales entre los componentes de hardware y software son representados a través de un solo nodo, el cual sería una PC.

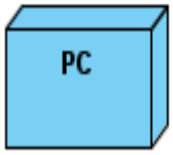

Figura 24: Diagrama de despliegue

4.5 Conclusiones

En este capítulo se definió un estándar de codificación que permite la comprensión del código empleado en la implementación y también se elaboraron los principales diagramas que intervienen en la organización y despliegue del sistema.

Conclusiones

Al término de la presente investigación, realizada con el objetivo de diseñar un sistema de animación facial muscular para el software de entrenamiento “Indicios Aduaneros”, se puede arribar a la conclusión de que el objetivo general fue alcanzado conjuntamente con las tareas de investigación trazadas para dar cumplimiento al mismo.

Luego de:

- ✓ Realizar un estudio sobre las diversas técnicas de animación facial existentes.
- ✓ Investigar sistemas que implementan animaciones faciales empleando modelos musculares.
- ✓ Diseñar una aplicación que cumpliera con los requisitos definidos.

Se obtuvieron los siguientes resultados concretos:

- ✓ Se diseñó un sistema óptimo y extensible de animación facial cuyo diagrama de clases permite adicionar otras expresiones o estados de ánimo.

Recomendaciones

Para el futuro trabajo con este sistema y dado la utilidad del mismo se recomienda:

- ✓ Concluir la implementación del sistema de animación.
- ✓ Hacer extensivo para proyectos de producción de software con características similares.
- ✓ Incluirle al sistema el módulo de animación de esqueleto.

Bibliografía Citada

1. **Suárez Roldán, Pamela Karla.** *Animación y Visualización de Fenómenos Naturales.* Departamento de Ingeniería en Sistemas Computacionales, Universidad de las Américas Puebla. Puebla, Mexico : s.n., 2003.

2. Wikipedia. *The Free Encyclopedia.* [En línea] http://en.wikipedia.org/wiki/Facial_animation.

3. **Bardsley, Daniel.** *Automatic Facial Animation Analysis.* University of Nottingham. 2007.

4. **Noh, Jun-yong y Neumann, Ulrich.** *A Survey of Facial Modeling and Animation Techniques.* Integrated Media Systems Center, University of Southern California. California : s.n.

5. Terra. *Terra.* [En línea] 13 de Junio de 2005. http://www.terra.es/.

6. Steve DiPaola. *A body of work.* [En línea] http://www.dipaola.org/.

7. SABIA. *Sistemas Adaptativos y Bioinspirados en Inteligencia Artificial.* [En línea] http://sabia.tic.udc.es/.

8. Multiverse. *Changing the <virtual> world.* [En línea] 31 de Julio de 2008. http://update.multiverse.net/.

9. Apple. *Apple.* [En línea] 2007. http://lists.apple.com/.

10. Amabilis. *Skeletal Animation.* [En línea] http://www.3d-canvas.com/.

11. 3D Programming Weekly: Graphics, Games. *Skeletal Animation #1.* [En línea] 01 de Febrero de 2006. http://www.3dkingdoms.com/.

12. OGRE Manual v1.6 ('Shoggoth'). *Animation.* [En línea] 04 de Noviembre de 2008. http://www.ogre3d.org/.

13. **Wang, Carol L.Y. y Forsey, David R.** *Langwidere: A New Facial Animation System.* Department of Computer Science, University of Calgary and University of British Columbia. Calgary and Vancouver : s.n.

14. IEEE Xplore. *IEEE Xplore.* [En línea] http://ieeexplore.ieee.org/.

15. **Pasternak, Gedalia.** The Expression Toolkit. *An Open-Source Procedural Facial Animation System.* [En línea] http://expression.sourceforge.net/.

16. Definición.de. *Definición de tiempo.* [En línea] 2008. http://definicion.de/tiempo/.

17. Server Proton. *Robótica.* [En línea] 15 de Noviembre de 2001. http://proton.ucting.udg.mx/.

18. MasterMagazine. *Definición de Tiempo Real.* [En línea] 2004. http://www.mastermagazine.info/.

19. DAC (División de Arquitecturas de Computadores). *Realidad Virtual y Animación 2008/2009.* [En línea] 30 de Octubre de 2008. http://dac.escet.urjc.es/.

20. **Rueda Chacón, César Julio.** *Aplicación de la Metodología RUP para el desarrollo rápido de aplicaciones basado en el estándar J2EE.* Facultad de Ingeniería, Universidad de San Carlos de Guatemala. Guatemala : s.n., Marzo de 2006.

21. **González Cornejo, José Enrique.** DocIRS. *¿Qué es UML?* [En línea] http://www.docirs.cl/.

22. Free Download Manager. *Sitio de Descargas de Software.* [En línea] 2004. http://www.freedownloadmanager.org/.

23. Vicerrectorado de Informática y Comunicaciones. *Servicios Informáticos U.C.M.* [En línea] 2003. http://www.sisoft.ucm.es/.

24. Microsoft. [En línea] 2009. http://www.microsoft.com/.

25. ANSI. *American National Standards Institute.* [En línea] http://www.ansi.org/.

26. NVIDIA. *Developer Zone.* [En línea] http://developer.nvidia.com/.

27. fotografiad.com. *Blog de fotografía digital.* [En línea] 20 de Marzo de 2008. http://www.fotografiad.com/interpolacion-de-imagenes/ .

28. Rhinoceros. *NURBS Modeling for Windows.* [En línea] 2009. http://www.es.rhino3d.com/.

Anexos

Anexo 1. Ejemplo de expresiones generadas por *"Expression Toolkit"*

Glosario de Términos

AGR: Aduana General de la República.

ANSI: El Instituto Nacional Estadounidense de Estándares (ANSI) es una organización sin ánimo de lucro que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas en los Estados Unidos de América. [25]

AU: Unidad de Acción muscular.

Bump Mapping: Es una técnica de gráficos 3D que permite darle un efecto de relieve a las superficies de los objetos. [26]

CASE: Acrónimo de Computer Aided Software Engineering; y en su traducción al español significa Ingeniería de Software Asistida por Computación. Define el uso de las computadoras para el desarrollo de software en todo su ciclo de vida.

Expression Toolkit: Sistema de animación facial de código abierto basado en el modelo anatómico de la cara.

FACS: Sistema de Codificación de Acción Facial.

FEM: Método de elemento finito (FEM, acrónimo de Finite Element Method) es un enfoque numérico que permite hacer una aproximación de la física de un objeto arbitrario complejo.

IDE: Un Entorno de Desarrollo Integrado, IDE, es un programa que incorpora un conjunto de herramientas y que tiene como objetivo ser de soporte a un programador para construir software. Este puede incorporar un lenguaje o varios de ellos; es considerado un programa de aplicación que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Interface: Sistema de Animación Facial en Tiempo Real

Interpolación: Es el proceso por el cual se calculan valores numéricos desconocidos a partir de otros ya conocidos, todo esto mediante la aplicación de algoritmos concretos. Simplificando, significa añadir píxeles para magnificar una imagen. [27]

Langwidere: Sistema de animación facial que integra un modelo jerárquico de splines con músculos simulados basado en las deformaciones de determinadas áreas de una superficie.

Morphing de imágenes: Es un efecto especial que se logra en las imágenes y animaciones convirtiendo una fotografía en otra a través de transiciones.

MVC: Patrón arquitectónico Modelo Vista Controlador que separa los datos, la interfaz de usuario y la lógica del control en tres elementos: la vista, el control y el modelo.

NURBS: Acrónimo de Non-Uniform Rational B-Splines. Son representaciones matemáticas de geometría en 3D capaces de describir cualquier forma con precisión, desde simples líneas en 2D, círculos, arcos o curvas, hasta los más complejos sólidos o superficies orgánicas de forma libre en 3D. [28]

OGRE 3D: Object-Oriented Graphics Rendering Engine, motor de renderizado 3D encaminado a escenas, escrito en el lenguaje de programación C++. [12]

Offsets: Dirección en memoria de un objeto.

QPL: Licencia Open Source que incorpora varios aspectos de la GPL.

Renderizado: Es la acción de asignar y calcular todas las propiedades de un objeto (imágenes fijas o de video) antes de mostrarlo en pantalla o guardarlo en un dispositivo de almacenamiento.

Shaders: Programas que permiten transformar un vértice o un pixel. Sirven para manipular efectos de iluminación, sombreado, animaciones y efectos de partículas.

Spline: Curva definida por segmentos mediante el uso de polinomios.

UML: Acrónimo de Unified Modeling Language. Es un lenguaje de modelado de alto nivel, pensado para diseñar software de describiendo los atributos de los objetos, las relaciones existentes entre ellos y las interacciones.

Vertex program: Es el código que funciona en cada vértice y que no puede añadir o restar vértices.