SIMULACIÓN DE FUEGOS ARTIFICIALES UTILIZANDO PYTHON

José David Alanís Urquieta
Universidad Tecnológica de Puebla
david.alanis@utpuebla.edu.mx

Joel Vázquez Valencia
Universidad Tecnológica de Puebla
david.alanis@utpuebla.edu.mx

Blanca Bermúdez Juárez
Benemérita Universidad Autónoma de Puebla
bbj@solarium.cs.buap.mx

Marisol Calderón González
Universidad Tecnológica de Huejotzingo
marisol_c@hotmail.com

María Luisa Morales Hernández
Universidad Tecnológica de Huejotzingo
marialmor@hotmail.com

Resumen

En este trabajo se presenta la Visualización de Fuegos Artificiales mediante modelado basado en física usando python. La técnica de simulación con visualización del modelado basado en física toma un modelo físico de la realidad, en este caso el comportamiento de un fuego artificial cuando es lanzado al cielo, explota en el aire y se apaga. Estas dos partes se comportan de forma separada y son modeladas mediante ecuaciones diferenciales ordinarias. Estas ecuaciones se solucionan mediante el método de Euler. Para el caso del lanzamiento se calcula la trayectoria y la rapidez. Para el caso de la explosión se calculan las posiciones de las partículas de manera separada y a cierta altura aleatoria se aplica la fuerza

de gravedad y el sistema se recalcula. La instrumentación numérica se realizó mediante el lenguaje de programación Python, aprovechando las ventajas de la programación orientada a objetos e imperativa del lenguaje. Por último, se realiza la renderización de los resultados mediante PyOpenGL, como interfaz de programación de aplicaciones. Para este problema en particular la visualización resulta con un buen grado de realismo gracias a la estabilidad y nivel de convergencia del método numérico utilizado y la técnica de visualización presentada.

Palabra(s) Clave(s): Método de Euler, modelado basado en física, Python, PyOpenGL, simulación con visualización.

1. Introducción

Los fuegos artificiales son un conjunto de luces de colores y detonaciones producido por diferentes dispositivos de pólvora que se usan en fiestas públicas [10]. Existe otra acepción en el ámbito militar, sin embargo, este trabajo se centra en la primera definición. En México y en otros países se usan fuegos artificiales para amenizar las celebraciones públicas. El comportamiento de las luces artificiales produce una visualización colorida y vistosa [1].

Resultaría interesante el poder visualizar los fuegos artificiales sin necesidad de un gasto en pirotécnica o esperar a que exista un evento y captar las imágenes, todo esto para utilizarlo en animaciones, complementos de animación, videojuegos, entre otras. [4] [5].

Así se puede realizar una reducción de costos si se utiliza la simulación, ya que no implicaría gastar recursos en la obtención de las imágenes, o bien generando de alguna manera los efectos, además del tiempo y esfuerzo necesario para lograr el realismo deseado [4].

En este contexto las técnicas de simulación con visualización son una herramienta útil en la construcción de visualizaciones con costos razonables en tiempos relativamente cortos si se les compara con la ingeniería que se tenía que realizar anteriormente y sin tomar en cuenta técnicas paralelas o complementarias como la realidad aumentada, la realidad virtual, animación digital, entre otras. [5]. Se trata

pues de un área desarrollada sobre una base interdisciplinaria que ha ido creciendo, generando un aporte contundente en la producción de animaciones con realismo y que ahorran costos en tiempo, dinero y esfuerzo.

El Modelado Basado en Física, realizada por Baraff y Witkin [4] [5], es una de las técnicas de simulación con visualización, y nace como una respuesta a la necesidad de realismo en la visualización de materiales describiendo su comportamiento a través de las fuerzas físicas con las que se interactúan, tanto externas como internas, considerando la evolución del objeto a través de ciclos de los materiales y su desenvolvimiento con otros objetos [5].

Un área importante del modelado basado en física son los sistemas de partículas. Según Baraff and Witkin [4] se puede definir un sistema de partículas como objetos que tienen masa, posición y velocidad, además de que responden a fuerzas pero que no tienen una extensión espacial. Además, el comportamiento de las partículas es colectivo, es decir, se conectan entre sí. En muchas producciones, sobre todo de películas, se utilizan los sistemas de partículas para visualizar agua, humo, cardumen y parvadas, entre otros muchos [4] [5].

Dado todo lo anterior, el comportamiento de los fuegos artificiales se puede modelar a través de un sistema de partículas.

Así el objetivo primordial de este trabajo es la simulación de los fuegos artificiales con visualización utilizando la técnica de modelado basado en física. La solución que se propone para lograr este cometido es utilizar un sistema de partículas y realizar su visualización.

Si bien es cierto que actualmente existen muchos softwares capaces de simular e incluso de programar sistemas de partículas con relativamente poco esfuerzo tales como Maya, Matlab, 3D Max, Blender, y un largo etcétera., estos softwares tienen una curva de aprendizaje alta, exigen muchos recursos computacionales y en otros casos no son software libre.

Además de lo anterior existen software realizados en el internet, como las que se aprecian en [11] y [12] entre otras muchas, sin embargo, estos softwares necesitan requerimientos especiales, y no es posible acceder al código para utilizarlo en otras aplicaciones con el fin de complementarlas, retomar alguna de

ellas o bien adaptarlas a procesos, animaciones o estudiarlas con los alumnos [4][1]. Las ventajas de realizar la simulación con visualización mediante Python y OpenGL ofrecen que el software pueda ser explorado, adaptado a las necesidades particulares, independientemente de que la licencia del software libre así lo solicite [4].

Aunado a lo anterior la implementación y el uso de este software es un medio por el cual se pueden explorar entre otras cosas: las distribuciones de los datos de manera aleatoria, añadir fuerzas adicionales, reutilizar el código y encontrar patrones de diseño adecuados para el comportamiento de otras simulaciones, etc.

Por último pero no menos importante el uso de la programación imperativa y orientada a objetos explota las bondades de ambos paradigmas, por un lado la potencia, flexibilidad y seguridad del paradigma orientado a objetos, además de su funcionalidad, reutilización de clases y su potencial didáctico, por el otro la transparencia para realizar interfaces con otros lenguajes y bibliotecas gráficas. De esta forma se tiene un recurso didáctico interesante y una aplicación escalable con uso de pocos recursos de forma abierta y legible para los interesados.

Antecedentes

El término sistemas de partículas fue desarrollado en computación gráfica desde la mitad del siglo XX. Se trata de una simulación de estilo Monte Carlo, donde los puntos son los que se comportan de una manera definida y simulan ciertos comportamientos, como una unidad u objeto definido más que como un conjunto de puntos. La definición de un sistema de partículas ha cobrado relevancia sobre todo en las producciones fílmicas, como ya se mencionó debido al ahorro de costos que se tiene en la implementación, uso y reproducción, en contraste con la reproducción de estos sistemas en el mundo real [12] [13].

Una brevísima historia de los sistemas de partículas se remonta a 1962, en un popular juego de video denominado SpaceWar las naves explotaban convirtiéndose en nubes de pixeles, se considera este como el segundo en la historia. En 1978 existió otro juego denominado llamado Asteroid donde las naves explotaban convirtiéndose en líneas discontinuas [13].

Se considera un de los primeros casos de éxito y que se presenta en el artículo Particle Systems A Technique for Modeling a Class of Fuzzy Objects donde se presenta el uso de un sistema de partículas en la super producción de Lucas Film Star Trek II: Wrath of Khan. En este trabajo primeramente se describen varios usos y aplicaciones de los sistemas de partículas y su simulación como son: fuego, agua, hierba, nubes, y objetos difusos en general. Por último, se describe y presenta el ***"efecto génesis"*** sobre una esfera que simula ser un planeta y la colisión entre los proyectiles y el planeta en cuestión (para una referencia completa véase [13]). En [5] se aprecia la relevancia que se otorga a la enseñanza, manejo y aplicación de sistemas de partículas. Se ha tomado como base la técnica de simulación con visualización del modelado basado en física, el cual comienza con la definición del problema como un modelo físico, que parte de la realidad, con ciertos atributos que se desean simular y visualizar.

Seguidamente se toma este modelo y se genera un modelo matemático que resulta ser generalmente, un conjunto de ecuaciones diferenciales ordinarias o parciales. Se resuelven las ecuaciones usando métodos numéricos instrumentando la solución en computadora [4]. Por último, se realiza la traducción de los resultados numéricos hacia las gráficas por computadora, utilizando varias herramientas computacionales, a este proceso se le denomina rendering. Finalmente se verifica el realismo de las gráficas por computadora, y se corrige el modelo de manera iterativa. Así el modelado basado en física puede ser aplicado a los fenómenos físicos que ocurren en la naturaleza o artificiales [1][5].

En la sección 2 se reseña el modelo físico de los fuegos artificiales, así como la descripción del modelo matemático que representa el comportamiento del sistema en dos fases, antes de la explosión y en la explosión.

En la sección 3 se describen los resultados de la instrumentación numérica realizada en python, así como la visualización realizada para este problema mediante PyOpenGL y las herramientas de Phyton.

En la sección 4 se desarrolla una discusión de los resultados, su interpretación alrededor de la visualización y también de la instrumentación numérica. Además, se trata de valorar las herramientas de software utilizadas para lograr realismo.

Por último, en la sección 5 se presentan las conclusiones del trabajo que incluyen los logros realizados y aportaciones del trabajo, así como los potenciales retos que se tienen para generar trabajos futuros.

2. Desarrollo

Modelo Físico

En este problema se observa un fuego artificial que es lanzado de manera vertical hacia el cenit. Posteriormente y a cierta altura dependiendo de la fuerza y de la carga que contiene, explota generando un conjunto de partículas que se dispersan alrededor del centro, con distintas tonalidades o con una sola tonalidad. Por último, el conjunto de partículas dispersas alrededor del centro se apaga debido a que la energía se disipa y por ende la luminosidad. Finalmente, las partículas apagadas caen por efecto de la gravedad.

Este problema se puede modelar como un problema de fuerzas, luego la formulación más importante es la de la segunda ley de newton, ecuación 1 [3].

$$F = ma \tag{1}$$

Así que el movimiento de la ecuación 1, se da en el eje de las *y*, y no existe movimiento en el eje de las *x*, lo cual significa que la fuerza que está en contra de la subida es la gravedad. Dado esto, es importante considerar el colocar una fuerza que supere la gravedad, para alcanzar cierta altura [3].

Se puede separar antes y después de la explosión, en dos momentos físicos importantes. Luego las ecuaciones para cada momento son la misma ecuación (1) para el momento 1 y la ecuación con gravedad para el momento 2 como se ve a continuación $a = -g$, así de manera general llega un momento a la altura máxima donde las fuerzas están en equilibrio como se ve en ecuación 2 [3][4].

$$F_1 + F_2 = 0 \tag{2}$$

Se sabe además que las ecuaciones 3a y 3b representan ambas fierzas.

$$F_1 = ma, \tag{3a}$$

$$F_2 = mg \tag{3b}$$

Ahora bien, existe un momento de explosión, donde las partículas juntas llegan al punto máximo de altura y en ese momento explota. Dado este modelo físico, se puede generar un modelo matemático.

Modelo Matemático

Una vez descrito el modelo físico a través de las ecuaciones que se presentaron. Dadas las ecuaciones 3a y 3b se tiene que ajustar de manera conveniente los valores de la masa y la fuerza.

El parámetro de aceleración se puede considerar como $\frac{dv}{dx} = a$ y lo mismo para la ecuación 3b [3]. Así se forman dos ecuaciones diferenciales, esta ecuación permite conocer la velocidad en cada punto. Posteriormente la transformación del modelo físico al modelo matemático debe de considerarse en una ecuación diferencial, y la masa igual a 1 unidad [4], ecuación 4.

$$\frac{dv_{y_{ant}}}{dy} = a, v_{y_0} = 0 \tag{4}$$

Esto es una ecuación diferencial ordinaria con condiciones iniciales, de la misma manera para cuando las partículas se someten a la acción de la gravedad después de la explosión, la ecuación 4 se transforma en ecuación 5 [3][5].

$$\frac{dv_{y_{des}}}{dy} = -g, v_0 = 0 \tag{5}$$

Ahora bien, ya que se conoce la velocidad en cada punto, es decir, en cada posición, es posible conocer de la misma forma la posición, aplicando la misma forma de la ecuación diferencial pero ahora a la posición, y tomando la velocidad como parámetro de la ecuación 4 y 5 se obtienen ecuaciones 6a y 6b.

$$\frac{dy}{dt} = v_{ant}, y_{0_{ant}} = 0 \tag{6a}$$

$$\frac{dy}{dt} = v_{des}, y_{0_{des}} = 0 \tag{6b}$$

Ahora las ecuaciones (4) y (5) proveen los parámetros necesarios para el cálculo de la posición que hacía falta [4].

La explosión se tratará como un problema de partículas de manera aleatoria con movimiento en el eje *0*. Se tiene una fuerza explosiva que "avienta" a las partículas hacia distintos lugares del escenario propuesto.

Ya que ahora se tiene un modelo matemático, se procederá a implementar una solución a este mediante la computadora, utilizando para ello una implementación de métodos numéricos.

Instrumentación Numérica

El método que se utilizó para realizar la solución es el método de Euler para solución de ecuaciones diferenciales ordinarias. Dada la simplicidad del método, su orden de complejidad bajo y la naturaleza de la ecuación diferencial [2].

El método de Euler tiene como objetivo obtener una aproximación a un problema bien planteado con valores iniciales, ecuación 7.

$$\frac{dy}{dt} = f(t,y), a \leq t \leq b, y(a) = \alpha \quad (7)$$

Se trata entonces de encontrar aproximaciones a la solución en los puntos específicos del intervalo [a, b], llamados puntos de red [2].

Se estipula primeramente que los puntos son igualmente espaciados y se toma un entero positivo y los puntos de red en el intervalo para construir la aproximación, ecuación 8.

$$t_i = a + ih, \text{para cada } i = 0,1,2,\ldots,N \quad (8)$$

Así la distancia común para los puntos denotada por $h = (b-a)/N$ se denominará tamaño de paso. Seguidamente y después de aplicar el teorema de Taylor y ajuste al error se obtienen ecuaciones 9a y 9b.

$$\omega_0 = \alpha \quad (9a)$$

$$\omega_{i+1} = \omega_i + hf(t_i, \omega_i) \quad (9b)$$

La ecuación 9b es también llamada ecuación de diferencias. Luego se construye el método de Euler como se muestra en la figura 1 [2][4].

Para aproximar la solución del problema de valor inicial
$y' = f(t,y),\ \ a \le t \le b,\ \ y(a) = \alpha$

En (N+1) números uniformemente espaciados en el intervalo [a,b]:

ENTRADA extremos a, b; entero N: condición inicial α

SALIDA aproximación ωa y en los (N+1) valores de t
Paso 1 Tome $h = \frac{b-a}{N}$
$t = a$
$\omega = \alpha$

Paso 2 Para i=1, 2, …, N haga pasos 3 y 4

Paso 3 Haga $\omega = \omega + hf(t,\omega)$;
$t = a + ih$
Paso 4 SALIDA (t,ω)

Figura 1 Método de Euler para solucionar Ecuaciones Diferenciales Ordinarias.

Visualización y Rendering

Al realizar la instrumentación numérica del método de Euler, en Python, se encuentran los parámetros de la velocidad y de la posición de las partículas antes y después de la explosión, en el eje de movimiento (eje *y*) [4] [6].

Primeramente, se elige la API, en este caso OpenGL. Entonces es necesario incluir las librerías de OpenGL a Python, para lo cual se utilizó PyOpenGL, que es una API que los interconecta, siendo multiplataforma y bastante usada para realizar aplicaciones y de visualización mediante Python [8] [9].

Para realizar la explosión, se modeló como un evento aleatorio con movimientos en el eje *z*, es decir, hacia el espectador. Para lograr esto, se debe tomar en cuenta que las coordenadas en el eje z cambian de forma aleatoria, incluso hacia atrás del origen que se ha planteado para este modelo [4][7].

Por último, estos resultados que se obtienen mediante la instrumentación numérica son adecuados a las variables construidas en la implementación para la visualización, así se logra el rendering, además de aprovechar la conexión e importación de las librerías gráficas [9].

Al tener una buena aproximación y convergencia del método numérico es posible tener un buen grado de realismo en la simulación con visualización, aprovechando las características de OpenGL, tales como generación de puntos y partículas de diversos tamaños, transformación de coordenadas, translación, entre otras muchas características gráficas [6].

3. Resultados

Dada la instrumentación del método de Euler mediante Python se tiene un diagrama de clases que se puede ver en la figura 2, hay una generalización de la clase partícula, es decir, se toma una clase llamada partícula y se crean los objetos, cientos o miles de estos objetos y se posicionan en un lugar de origen para comenzar la visualización.

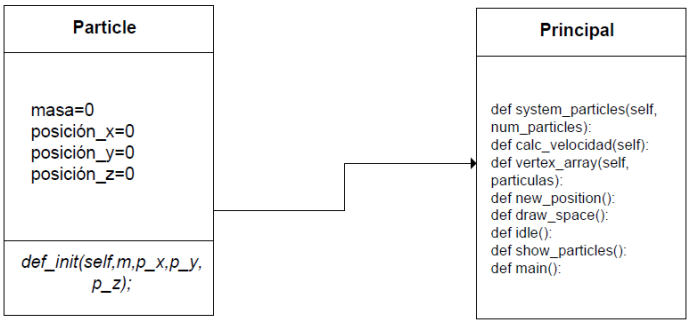

Figura 2 Diagrama de Clases de la Simulación de Fuegos Artificiales.

Cabe mencionar que la generación de un arreglo de objetos para controlar las partículas resulta adecuada y se adapta a la implementación que se está realizando [7][8].

Las clases necesarias para crear un mundo virtual se importan de OpenGL[9]. Así desde la clase principal se controla la visualización de las partículas utilizando PyOpenGL, adaptando las estructuras de datos al problema [4].

A continuación, se muestran los resultados de la visualización, en la corrida del programa en figura 3, 4, 5 y 6 [4] [6].

La visualización comienza con la subida hasta el punto máximo donde es "aventado" hacia la vertical, las translaciones del sistema son programadas por medio de las funciones propias de OpenGL, véase la figura 3.

En un segundo momento las partículas llegan al punto máximo, iniciando la explosión con un movimiento alrededor de la esfera con centro donde llego después de haber sido lanzadas hacia la vertical en el eje *y*, obsérvese la figura 4.

En un tercer momento las partículas se continúan separando cada vez más hasta llegar al punto de máxima exclusión, todo el movimiento que se puede observar esta sobre el eje *z*, como se puede ver en la figura 5.

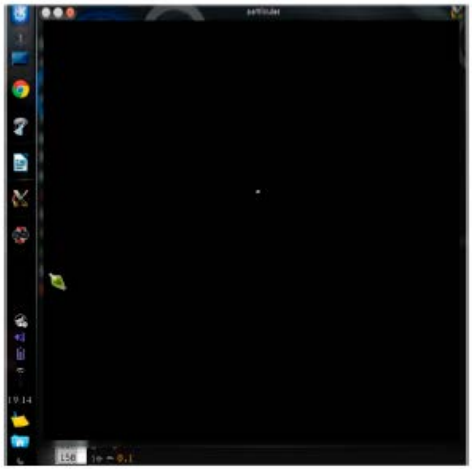

Figura 3 Fuego artificial llegando hacia el punto más elevado.

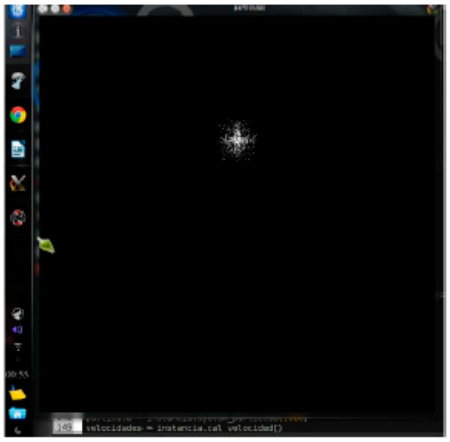

Figura 4 Partículas comenzando la explosión.

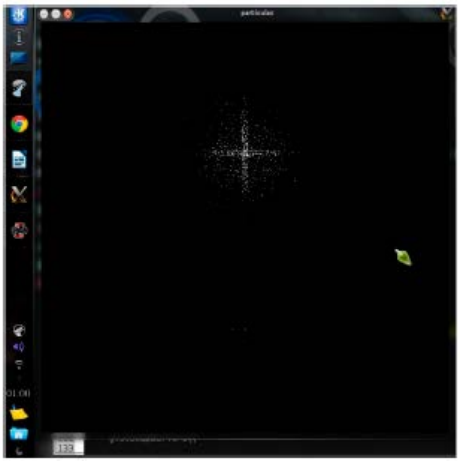

Figura 5 Partículas llegando a su máximo alrededor de esfera, centro en altura máxima.

Por último, las partículas se detienen y por efecto de la gravedad comienzan a caer cuando la explosión termina. Además, las partículas se apagan y comienzan a morir, para este caso van desapareciendo y se pintan del color de fondo, como se aprecia en la figura 6.

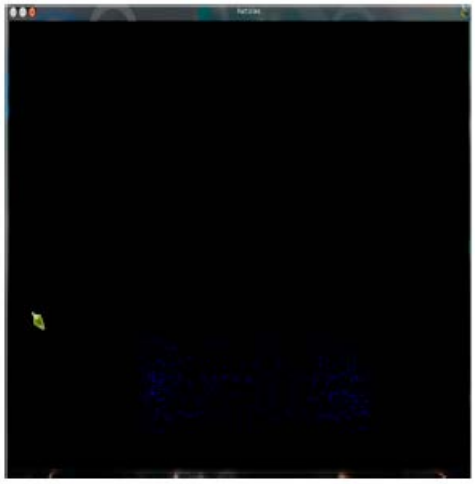

Figura 6 Fin de la explosión y comienza la muerte de las partículas.

4. Discusión

En los resultados que se han presentado, existen dos aspectos importantes a considerar: por un lado, está el comportamiento y número de las partículas para lograr un buen grado de realismo, y el otro aspecto es la forma como se considera el sistema de forma computacional y como se adapta la programación de manera conveniente en la solución del problema planteado.

En el contexto del comportamiento de las partículas se observa como hasta antes de la explosión no existe mayor problemática, dado que las ecuaciones están bien definidas y se resuelven obteniendo un grado de realismo aceptable, lo mismo sucede con el momento después de la explosión las partículas caen por gravedad y se apagan para ir al fondo de la animación.

La problemática de la explosión es resuelta con un modelo aleatorio, se puede observar un patrón muy bien definido, que en un momento dado puede repetirse y generar una visualización no muy atractiva al espectador, dado el movimiento de las mismas o que siempre se visualice el mismo , esto se puede resolver si se

utilizan distribuciones de probabilidad con variable aleatoria discreta o continua, que provoquen que el movimiento sea diverso, sin perder de vista que se forma una esfera alrededor del punto de la altura máxima.

Por otra parte, el lenguaje de programación, la API y la API gráfica utilizadas, se adaptan bastante bien a este problema en particular y a problemas similares al planteado en este trabajo. El paradigma de Python orientado a objetos permite el uso de estructura de datos que no requiere grandes prestaciones para simular una partícula, ya que como se mencionó, mediante la generación de miles de partículas la simulación tiene un grado de realismo aceptable para este problema en particular.

Además de todo lo anterior, los objetos gráficos que se pueden utilizar, gestionar y visualizar mediante OpenGL son fáciles de implementar con el paradigma orientado a objetos, adicionalmente la complejidad espacial de la solución del problema es reducida comparada con otros lenguajes estructurados e incluso orientados a objetos como Java. La citada complejidad se puede reducir, aún más, aprovechando las características de Python, dado que se realiza un mayor número de operaciones, tanto matemáticas, como sobre los datos con relativamente pocas instrucciones y número pequeño de líneas de código.

5. Conclusiones

En este trabajo se ha presentado la simulación con visualización de Fuegos Artificiales mediante la técnica de simulación del modelado basado en física utilizando Python. Cada etapa del método ha sido presentada y se ha obtenido una visualización mediante el uso de PyOpenGL. Es importante decir que el paradigma orientado a objetos resulta adecuado en la visualización y que se adapta a las necesidades de objetos gráficos y estructuras necesarias para su gestión dentro de la mencionada simulación, obteniendo una visualización con un realismo aceptable.

En el caso de la explosión y el comportamiento de las partículas se puede ampliar el trabajo hacia la experimentación con otras distribuciones de partículas que generen otras figuras, que, si bien deben tener un comportamiento aleatorio, están

delimitadas por las fuerzas físicas a una esfera con centro en el punto máximo de la altura. Además de lo anterior se ha experimentado con un sistema de partículas en el orden de los millares, se puede optimizar la gestión de las partículas para lograr manejar sistemas de mayor orden.

Por último, al obtener una clase que simula el comportamiento de las partículas en un fuego artificial, es posible generar más fuegos artificiales de distintos colores y comportamientos.

Se puede, además, interactuar con otros elementos del mundo virtual que se crea, aprovechando las ventajas del lenguaje de programación y la forma como se está tratando el problema. Así este método se puede extender a otras aplicaciones de software que involucren el uso de sistemas de partículas.

6. Bibliografía y Referencias

[1] P. Baker, D. Hearn, Gráficos por computadora con OpenGL. 3ra. Edición. Prentice Hall.

[2] R. L. Burden, J. D. Faires, Análisis Numérico. Novena Edición. 2011. Cenage Learning.

[3] R. A. Serway, Física. 2V. Cuarta Edición. 1997. McGraw Hill. México.

[4] J. Vázquez Valencia, J. D. Alanís Urquieta, Implementación de Sistema de Partículas en Python con herramienta gráfica opengl. 2014. Tesina de TSU en TICS, Universidad Tecnológica de Puebla,

[5] A. Witkin, D. Baraff, Physically Based Modelling. Carniege Mellon University. Online ACM Siggraph '97 Course notes.

[6] Wright, S. Richard, OpenGL Super Bible. 1965. Addison-Wesley.

[7] https://docs.python.org/2/library/math.html. Agosto 2015

[8] https://docs.python.org/2/library/random.html. Agosto 2015

[9] http://serdis.dis.ulpgc.es/~atrujill/iga/ficheros/Intro_GLUT.pdf. Agosto 2015

[10] http://dle.rae.es/?id=IYNxrTR. Junio 2016

[11] http://www.notdoppler.com/fireworks.php. Agosto 2016

[12] www.fwsim.com. Agosto 2016

[13] http://www.di.ubi.pt/~agomes/tjv/teoricas/09-particles.pdf. Agosto 2016

7. Autores

José David Alanís Urquieta, es Licenciado en Ciencias de la Computación por la Benemérita Universidad Autónoma de Puebla, Maestro en Ciencias de la computación con especialidad en cómputo científico y Doctor en Tecnologías de la Información y Comunicación por la Universidad Popular Autónoma del Estado de Puebla.

Blanca Bermúdez Juárez, es Licenciado en Computación por la Benemérita Universidad Autónoma de Puebla, Maestra en Sistemas de Información, Universidad de las Américas Puebla. Doctorado en Matemáticas por la Universidad Autónoma Metropolitana. Profesor con Perfil Deseable PRODEP y Miembro del Sistema Nacional de Investigadores Nivel I.

Joel Vázquez Valencia, es Ingeniero en Tecnologías de la Información y Comunicación, por la Universidad Tecnológica de Puebla, actualmente trabaja en la empresa Autotodo Mexicana como Desarrollador de Aplicaciones.

Marisol Calderón González, tiene el título de Licenciado en Ciencias de la Computación por la Benemérita Universidad Autónoma de Puebla y la Maestría en Sistemas Computacionales por la Universidad Popular Autónoma del Estado de Puebla.

María Luisa Morales Hernández, tiene el título de Licenciado en Ciencias de la Computación por la Benemérita Universidad Autónoma de Puebla y la Maestría en Ingeniería Administrativa.