Las plataformas de desarrollo de videojuegos como nuevo modelo de espacio virtual

Jaime Munárriz Ortiz
Universidad Complutense de Madrid. Fac. Bellas Artes.

Resumen

Las plataformas de desarrollo de videojuegos han evolucionado desde un entorno para la escritura y depuración de código hacia herramientas para la construcción de espacios virtuales interactivos. El cambio de modelo ha acercado a diseñadores y artistas gráficos al espacio de trabajo del desarrollo de videojuegos, al facilitar el acceso a estas herramientas. Estas nuevas plataformas de desarrollo abren caminos para la creación de experiencias inmersivas y lugares de encuentro multiusuario, como espacios virtuales 3D con contenidos multimedia interactivos.

Palabras clave: Videojuegos, motor de juego, herramientas de desarrollo, espacio virtual, 3D.

1. Introducción

Las herramientas de desarrollo de videojuegos han sufrido una importante transformación en los últimos años, tanto en su aspecto y estructura como en su función y operatividad. De sistemas de integración de componentes de código han pasado a incorporar funciones para el manejo de los elementos visuales del videojuego y construcción de la interfaz, niveles y mapas. De este modo de herramientas de programación se han convertido en herramientas de diseño, integrando el linkado y la depuración del código con la construcción de escenarios y niveles, tareas que antes debían realizarse en editores independientes.

2. Los entornos de desarrollo de videojuegos

IDEs

Los entornos de desarrollo conocidos como IDEs (Integrated Development Environment) se crean ante el aumento de la complejidad en la escritura de código de programación, especialmente con el auge del lenguaje C. El programador deja de trabajar sobre un único archivo en el que escribe y perfecciona el programa que intenta desarrollar, para pasar a controlar un complejo ecosistema de archivos

independientes y paquetes complejos de librerías con problemáticas interdependencias. El proceso de pruebas pasa también de un simple *ejecutar* (*run*) a una cadena de operaciones sucesivas que se ocupan de depurar el código, establecer la conexión con las diversas librerías, compilar todo en un programa ejecutable para finalmente lanzarlo. En este contexto de creciente complejidad para el programador surgen los IDEs que tratan de simplificar el proceso, unificando en un mismo espacio el control y acceso a los diversos archivos y librerías implicados en un proyecto, y facilitando opciones de depuración e integración de funciones y librerías dentro del mismo entorno.

Estos entornos de desarrollo despegan en los primeros ochenta con el auge de los ordenadores de 16 bits y sus nuevos entornos y sistemas operativos, basados fundamentalmente en el lenguage C, aunque se construyen también para otros lenguajes de programación populares como Pascal, Basic, Prolog o Delphi. Borland es la empresa que populariza este tipo de entornos con su pionero TurboPascal.

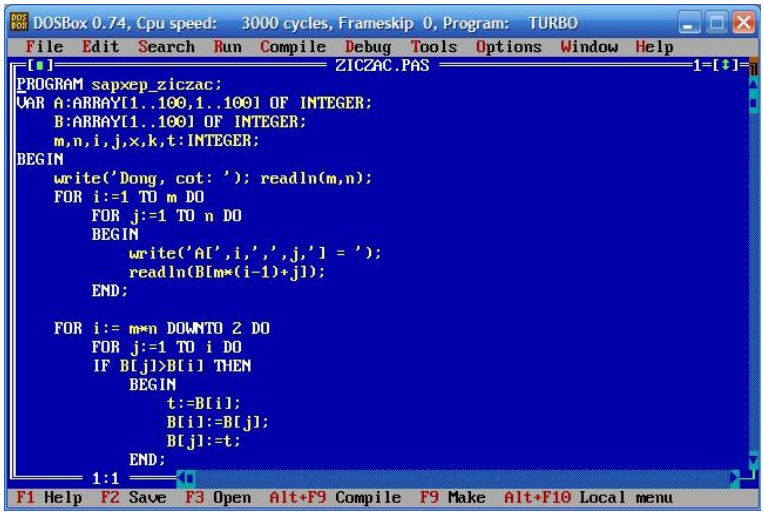

Fig.1. El IDE de TurboPascal

Con la llegada de los sistemas operativos basados en ventanas, iconos, puntero y ratón los entornos de desarrollo adquieren una nueva función, la de diseñar el aspecto visual de las distintas pantallas que se ofrecen al usuario final, especialmente en aplicaciones de ofimática y administración de datos. Estos entornos permiten colocar elementos con componente visual como cajas de texto, botones, listas o imágenes, y facilitan la conexión con bases de datos en la misma aplicación. Con esta ayuda el diseño del aspecto de las aplicaciones puede realizarse desde el mismo entorno de desarrollo. Estos IDEs comienzan a denominarse “visulales”, siendo la familia de entornos de Microsoft la más conocida, como VisualBasic, VisualC++, etc.

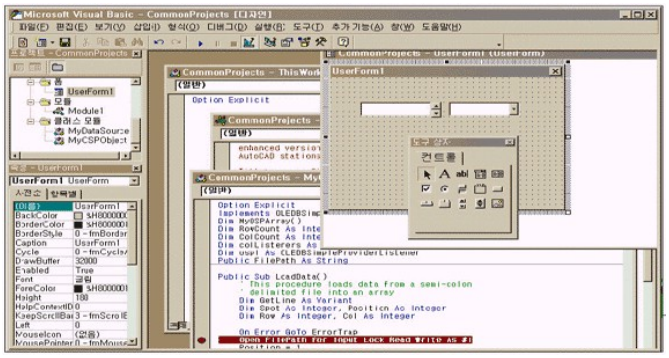

Fig2. El IDE de VisualBasic

Los IDEs para el desarrollo de videojuegos

Los videojuegos se desarrollan inicialmente en código máquina en su gran mayoría, necesitando aprovechar al máximo la potencia del procesador. El programador trabaja simplemente con un editor de texto y un ensamblador, que compila el código escrito en lenguaje ensamblador, más inteligible para el ser humano, en código máquina, las instrucciones concretas que finalmente ejecutará el procesador.

Con el paso a sistemas de 16 bits, el lenguaje C pasa a ser predominante. Los juegos comienzan a escribirse en C, reservando la escritura de código máquina para las partes más críticas del bucle central del programa, en las que podría ralentizarse la ejecución. Sin embargo según los sistemas operativos ganan en complejidad, se llega a demostrar que los compiladores pueden generar un código tan eficaz desde C como el que puede escribir directamente un humano en ensamblador, con lo que poco a poco el desarrollo de videojuegos se traslada completamente al lenguaje C. Sólo juegos que no necesitan gráficos veloces en tiempo real se escriben en otros lenguajes, como es el caso de las aventuras conversacionales.

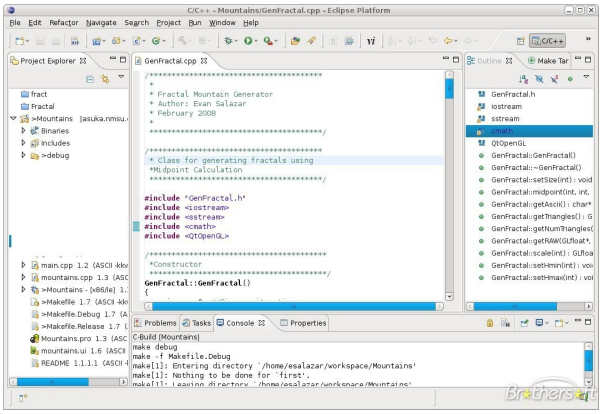

Fig. 3. El IDE de Eclipse con lenguaje C.

Motor de juego

Estos videojuegos más complejos, que corren sobre sistemas más potentes, centran su desarrollo en la elaboración de un motor propio, encargado de la ejecución del núcleo central del juego. Este motor se ocupa de comprobar las acciones del usuario mediante su uso de elementos de entrada del interface, de calcular los efectos sobre los elementos del juego, recalcular posiciones y estados, y generar los gráficos y sonidos que permitan visualizar las acciones y cambios del sistema. Son especialmente importantes en los juegos en 3D, que requieren un aprovechamiento máximo de los recursos de cálculo del procesador y la tarjeta gráfica. Cada empresa o programador importante se ocupa de desarrollar un motor avanzado, que le pueda ofrecer ventajas sobre la competencia, generando una experiencia de juego más sofisticada, con gráficos e inteligencia artificial superiores a los demás. Los juegos de ID software Doom y Quake sientan un modo de desarrollo basado en la elaboración de un motor propio que se convertirá en el estándard de la industria, aunque todavía de forma embrionaria: "it could have gotten fixed if we had reorganised the company into an engine development team and a game development team (…) that would have been an optimal strategy for the company, but we did not really analyse the problem back then. All we really knew was how to work on the game together. " [1].

Desarrollo del motor (*engine*)

El desarrollo de un motor de juego independiente del juego en sí permite a los desarrolladores centrarse en los datos que conformarán la experiencia de ese producto particular: gráficos, historia, escenarios, niveles. El motor puede licenciarse a otras compañías para el desarrollo de otros productos con contenido diferenciando, convirtiéndose en una fuente de ingresos importante y diferenciada. Los estudios pequeños no pueden dedicar tanto tiempo a la creación de un motor propio que pueda competir con los mejores, y con este sistema pueden crear un producto con los máximos niveles de calidad, centrados sólo en el diseño y creación de contenidos.

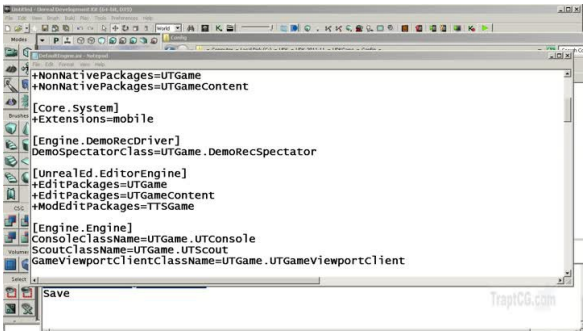

Fig. 4. Desarrollo en un motor de juego

Level Editors

Los motores de juego son en sus comienzos sobre todo un conjunto de librerías de código. El desarrollo de los gráficos del juego y de los escenarios y niveles requiere

por tanto el uso de otras herramientas. Resulta habitual para las empresas desarrollar un editor de niveles de uso interno que facilite la creación de los contenidos necesarios en ese producto. En algunos casos los editores de niveles se incluyen junto al juego principal, lo que contribuye a crear comunidades de usuarios que fabrican sus propios contenidos adicionales, añadiendo valor añadido al producto inicial.

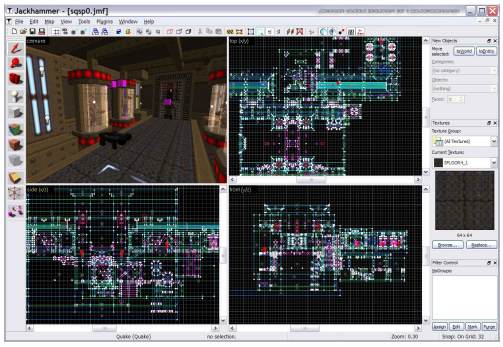

Fig. 5. Editor de niveles de Quake

Engines actuales

Los motores de juego actuales han incorporado las funciones de edición de niveles y escenarios. Los motores modernos permiten importar elementos gráficos y modelos 3D dentro de un interfaz interactivo y visual. La tarea de elaborar e integrar los contenidos necesarios para el desarrollo del juego se simplifica, al poder elaborar los escenarios dentro del mismo motor y poder comprobar así el funcionamiento adecuado de los elementos y evaluar la experiencia del jugador sin necesidad de compilar y salir del entorno.

El aspecto de estos motores ha cambiado por completo. De herramientas de trabajo para el programador, con ventanas de edición de código, depuración, linkado y control de librerías se han convertido en programas de diseño gráfico con ventanas de trabajo en el espacio tridimensional, subordinando las tareas de edición del código a las de diseño e integración de elementos multimedia.

Antecedente: Director

En el pasado encontramos un antecedente similar a este proceso de transformación de una herramienta de desarrollo: la herramienta de prueba de elementos de juego (sprites animados) y edición de mapas y niveles VideoWorks se convirtió poco a poco en la herramienta de desarrollo de aplicaciones multimedia más importante, hacia 1995, cuando la invención del soporte CDRom impulsó toda una generación de software interactivo con elementos multimedia. “is a standlone system that incorporates a virtual dope sheet to animate multiple sprites on the screen “ [2] En Director la parte más importante del interfaz se dedicaba a construir los niveles y pantallas de la aplicación, dejando las tareas de edición y depuración de código subordinadas a la elaboración de los aspectos visuales y el diseño de la interfaz gráfica de usuario.

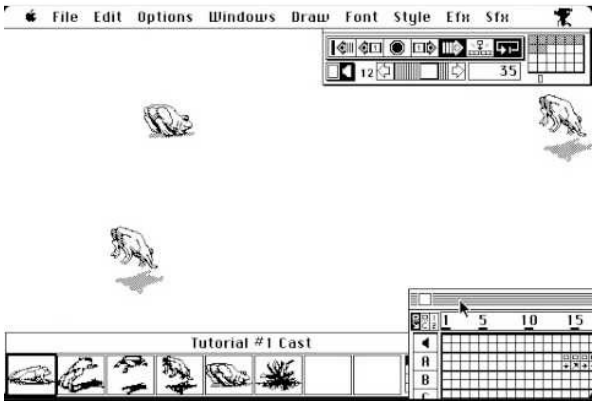

Fig. 6. Prueba de sprites en Macromind VideoWorks

3. Espacio virtual

Espacio 3D

El motor de videojuego se asemeja de repente a una aplicación de modelado y animación 3D, más que a un IDE de programación. Tenemos una pantalla de usuario dividida en distintas vistas correspondientes a los distintos ejes XYZ y cámara, una librería de elementos gráficos y sonoros, y la posibilidad de probar el funcionamiento del nivel en desarrollo, entrando en el modo de prueba en el que podemos posicionarnos en el espacio tridimensional, movernos y desplazarnos por el terreno, e interactuar con los elementos que lo conforman.

El motor de desarrollo de videojuegos se convierte de este modo en un espacio de construcción y testeo de niveles, mapas y espacios virtuales, que nos permite experimentar la inmersión en estos espacios facilitando su creación y depuración.

Fig. 7. El IDE de Unity3D

Espacios virtuales y juegos multijugador en línea

La evolución de estas herramientas de desarrollo de videojuegos coincide con el florecimiento del modelo de espacio virutal como lugar de encuentro multiusuario. Los juegos, que pasaron de la pantalla en 2D a la simulación de espacios en 3D, han incorporado como componente fundamental la experiencia multijugador. La experiencia de colaboración o competición con otros seres humanos es uno de los elementos más importantes en el éxito de los juegos modernos.

Se ha explorado el modelo de espacio virtual como posible lugar de encuentro en plataformas sociales, siendo el más conocido el caso de Second Life. El encuentro e intercambio de opiniones y experiencias entre distintas personas, sin embargo, parece desarrollarse de forma más adecuada en aplicaciones con interfaz tradicional, que facilitan la escritura y lectura de mensajes y el intercambio de enlaces, vídeos o música (Facebook, Twitter o Instagram).

Los juegos multiusuario en línea parecen ofrecer un lugar de encuentro más apetecible para pequeñas comunidades de amigos dispersos en la geografía, ya que proporcionan una experiencia lúdica que permite además el intercambio de opiniones, anécdotas e ideas a modo de conversación de chat, o con conexión directa de sonido en paralelo a la acción del juego. Los espacios virtuales de encuentro no parecen ofrecer alicientes suficientes que motiven a las personas a incorporarse y participar en la interacción como actividad principal. Los juegos multiusuario por el contrario sí ofrecen una primera motivación lúdica, enriquecida por la posibilidad de encuentro e intercambio social.

Fig. 8. Experiencia multiusuario en SecondLife

4. Arte y videojuegos

Experimentos artísticos

Los nuevos motores de videojuegos se están utilizando de forma sorprendente para experiencias artísticas que exploran sus posibilidades de interacción dentro de

contextos diferentes a los propios del videojuego. Estos motores facilitan la construcción de espacios inmersivos, de simulaciones de entornos bi y tridimensionales, acercando al artista de los nuevos medios una tecnología que no podría abarcar desde su taller. Los experimentos artísticos pueden abrir nuevas ideas para el desarrollo de experiencias innovadoras con el usuario, interactividad e interfaz.

Fig. 9. Espadaysantacruz: Invaders project [3]]

Espacio sonoro

Los entornos de desarrollo permiten la incorporación de sonidos en el espacio tridimensional. Esta simple herramienta permite la construcción de espacios sonoros, próximos a las prácticas artísticas vinculadas al arte sonoro y la ecología acústica. Podemos construir lugares inmersivos en los que nuestro deambular transcurre entre masas sonoras y eventos acústicos que conforman un paisaje sintético, un espacio en el que sumergirnos rodeados de capas de elementos, sonidos que configuran el espacio.

Juegos como Narcosis o Mirrormoon [4] presentan espacios sonoros inmersivos como parte fundamental de su mecánica de juego, aplicando este concepto a la experiencia del usuario.

nuevos lugares

La creación de espacios virtuales inmersivos abre la posibilidad de construir nuevas experiencias de usuario, lugares imaginarios, realidades no existentes o mundos hiperrealistas, utopías o distopías en los que el usuario puede sumergirse, enfrentado a un entorno de creación especialmente diseñado para la interacción. Las tecnologías del videojuego han acercado la realidad virtual inmersiva a los creadores experimentales, simplificando la tecnología y el hardware necesario para crear experiencias inmersivas ricas en contenidos multimedia. Cualquier mundo imaginado puede ser creado, sólo en la cabeza del artista están los límites.

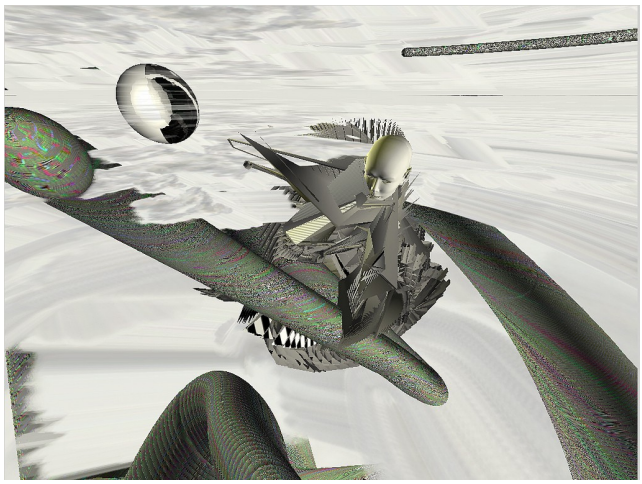

Fig. 10. Rosa Menkman. Xilitla. Pieza experimental en Unity3D [5]

En estos nuevos espacios encontramos como desafío la introducción de narrativas, asociaciones y deslizamientos en los significados que posibiliten una experiencia con sentido, un desarrollo de la interacción que aporte contenido y significado a los lugares creados. Narrativa, historia, referentes, acciones condicionadas, pistas y acciones desencadenadas por eventos conforman los elementos de una nueva forma de configurar experiencias y vivencias para el actor/espectador, el usuario privilegiado de estos nuevos medios de creación.

5. Espacio sonoro

Pieza Sound Garden

Dentro de este marco elaboro la pieza "Aab Garden - the Shadow". Se propone como lugar de deriva, espacio para la experiencia de piezas sonoras de caracter ambiental y envolvente. Tanto el espacio construido como la experiencia sonora tratan de situarse en ese difícil lugar entre primer plano y fondo, sin destacar en el protagonismo pero configurando una experiencia coherente, compleja y con sentido. La pieza se desarrolla en el motor de Unity3D, atendiendo sobre todo a la posibilidad de integrar la experiencia del usuario en un navegador de Internet, gracias al plugin que permite esta distribución de medios.

La pieza desarrolla contenidos y estéticas desarrolladas anteriormente en "Spiral Garden/Spiral Hell" y la experiencia de la Deriva Aural, presentada en el Auditorio 400 del Museo Reina Sofía en 2012. La experiencia sonora se abre desde el original sonido estéreo, pasando por la experiencia cuadrafónica del Reina, hasta un micro-universo explorable con sonido multipuntual en esta versión explorable.

6. Conclusiones

La tecnología de los motores de juego actual, convertidos en entornos de desarrollo de espacios virtuales, nos permite imaginar nuevas experiencias inmersivas de usuario, construyendo espacios navegables y entornos sonoros interactivos. Estas posibilidades para la creación deben ser exploradas, en la construcción de nuestro nuevo imaginario audiovisual y como nuevos modos de encuentro e intercambio.

1. John Romero Interview: Doom, Quake and His Career Either Side. (2013)
http://www.gamestm.co.uk/features/john-romero-interview-doom-quake-and-his-career-either-side/

2. Judson Rosebush. A HISTORY OF COMPUTER ANIMATION. p51 (1992) http://vasulka.org/archive/sitemap.html

3. The Invaders Project by Espadaysantacruz Studio. (2013) http://www.espadaysantacruz.com/Invaders-project

4. MirrorMoon. (2012)
http://www.mirrormoongame.com/

5. Menkman, Rosa. Xilitla. (2014)
http://xilitla.beyondresolution.info/