Universidad Politécnica de Tecámac.

Materia: Programación Visual

Docente: Emmanuel Torres Servín

Grupo: 5322IS

Actividad: Análisis de la programación visual

Integrantes:	Matriculas:
✓ Alemán Pérez Natali Joselin	1321124050
✓ Ángel Velasco Marco Joel	1321124015
✓ Isaac de León Carbajal	1321124054

Contenido

Tarea 1

Conceptos de la Programación Orientada a Objetos

Clases

Una clase es una plantilla. Define de manera genérica cómo van a ser los objetos de determinado tipo. Es un molde del que luego se crean múltiples objetos con características similares.

Objetos

Un objeto es una entidad concreta que se crea a partir de la plantilla que es la clase. Este nuevo objeto tiene ya "existencia" real, puesto que ocupa memoria y se puede utilizar en el programa. Es el conjunto de variables o datos y métodos o funciones relacionadas entre sí.

Mensajes

Es la petición de un objeto a otro para solicitar la ejecución de alguno de sus métodos o para obtener el valor de uno de sus atributos públicos. Siempre hay un receptor, lo que no ocurre a una llamada a procedimiento.

Los Pilares de la POO

Encapsulamiento

Permite que todo lo referente a un objeto quede aislado dentro de éste. Es decir, que todos los datos referentes a un objeto queden "encerrados" dentro de éste y sólo se puede acceder a ellos a través de los miembros que la clase proporcione (propiedades y métodos).

Abstracción

Implica que la clase debe representar las características de la entidad hacia el mundo exterior, pero ocultando la complejidad que conllevan. Esto es, abstraer la complejidad que haya dentro dando una serie de atributos y comportamientos (propiedades y funciones) específicos de un objeto, los que los distinguen de los demás.

Herencia

Es el resultado de crear un modelo a partir de otro. Es decir, cuando una clase hereda de otra obtiene todos los rasgos que tuviese la primera. La clase que hereda de otra obtiene todos los rasgos de la primera y añade otros nuevos y además también puede modificar algunos de los que ha heredado.

A la clase de la que se hereda se le llama clase base, y a la clase que hereda de ésta se le llama clase derivada.

Polimorfismo

El concepto de polimorfismo se refiere al hecho de que varios objetos de diferentes clases, pero con una base común, se pueden usar de manera indistinta, sin tener que saber de qué clase exacta son para poder hacerlo.

También se refiere al a posibilidad de definir métodos diferentes denominados de forma idéntica dentro de una misma clase, con alguna característica que los identifique, como su tipo de retorno o cantidad y/o tipo de parámetros.

Características y aplicaciones de eventos

Los eventos son las acciones sobre el programa. Cuando se produce o dispara un evento sobre un determinado componente, se da inicio a un conjunto de acciones programadas por el programador para ese evento concreto. Algunos ejemplos de los eventos son dar clic a un botón, arrastrar un icono, pulsar una tecla o una combinación de teclas, escribir en una caja de texto, etc.

Los eventos son recogidos por un bucle exterior permanente y distintos procesos se encargan de tratarlos. Habitualmente, este bucle externo permanece oculto al programador que simplemente se encarga de tratar los eventos, aunque en algunos entornos de desarrollo (IDE) será necesaria su construcción.

Los eventos permiten crear código por partes o módulos, que se ejecutarán sólo cuando se dispare cierto evento. Esto logra un mantenimiento más sencillo, ya que si alguno de los eventos no funciona como se espera, sólo se debe revisar y corregir el fragmento de código que corresponde a ese evento, en lugar de tener que buscar en todo el programa. Además, la escritura de código por fragmentos también facilita la reutilización de estos fragmentos.

Características de componentes y métodos visuales y no visuales

Componentes visuales

Un componente es visual cuando tiene una representación gráfica en tiempo de diseño y ejecución (botones, barras de desplazamiento, cuadros de edición, etc.)

Componentes no visuales

Caso contrario, es decir, no cuentan con una representación gráfica (temporizadores, cuadros de diálogo -no visibles en la fase de diseño-, etc).

Métodos

Un método es un bloque de código que tiene definido en su interior un conjunto de instrucciones, las cuales realizan una determinada tarea. Un programa hace que se ejecuten las instrucciones al llamar al método y especificando los argumentos de método necesarios.

Procesos de desarrollo visual en proyectos distribuidos y de escritorio

Un sistema distribuido es un conjunto o grupo de equipos que son independientes entre sí y que actúan como un único equipo de forma transparente y que tienen como objetivo la descentralización del procesamiento o el almacenamiento de información. La distribución permite obtener grandes prestaciones con un coste razonablemente bajo. En la actualidad, la mayoría de los sistemas informáticos son distribuidos y no dependen de un único nodo o equipo para funcionar.

Las aplicaciones de escritorio se pueden descargar desde el Microsoft Store, descargar desde Internet o instalarse con algún tipo de medio (como un CD, DVD o dispositivo de almacenamiento USB). Se abren con un archivo .EXE, .MSI o .DLL y normalmente se ejecutan en el dispositivo, a diferencia de las aplicaciones basadas en web (que se ejecutan en la nube).

Las fases comunes del desarrollo de software son:

- Planificación: El propósito de este paso es averiguar el alcance del problema y determinar las soluciones. En esta etapa se deben considerar los recursos, costos, tiempo, beneficios y otros elementos.
- Requerimientos: Se consideran los requisitos funcionales del proyecto o la solución. También es donde tiene lugar el análisis del sistema y de las necesidades de los usuarios finales para garantizar que el nuevo sistema pueda satisfacer sus expectativas.
- Diseño: Describe en detalle las especificaciones, características y operaciones necesarias que lograrán los requisitos funcionales del sistema propuesto que se implementará.
- Desarrollo: Es cuando se contrata a un programador, ingeniero de redes y / o desarrollador de bases de datos para que realice el trabajo principal en el proyecto.

- Integración y pruebas: Implica la integración de sistemas y las pruebas del sistema (de programas y procedimientos), normalmente realizadas por un profesional de aseguramiento de la calidad (QA)
- Implementación: Esta fase implica la instalación real del sistema recientemente desarrollado. Este paso pone el proyecto en producción moviendo los datos y componentes del sistema antiguo y colocándolos en el nuevo sistema a través de una transición directa.
- Operaciones y mantenimiento: Conlleva el mantenimiento y las actualizaciones periódicas que el sistema requiera según las necesidades de los usuarios finales.

Requerimientos visuales de proyectos distribuidos y de escritorio

Un requerimiento define las funciones, capacidades o atributos intrínsecos de un sistema de software, en otras palabras, describe como un sistema debe comportarse.

Los requerimientos se pueden dividir en dos grandes categorías:

- Requerimientos funcionales: Son aquellos que describen cualquier actividad que el sistema deba realizar, es decir, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones.
- Requerimientos no funcionales: Definen características o cualidades generales que se esperan de un sistema y establecen restricciones sobre el producto, el proceso de desarrollo de software y restricciones externas que el software debe lograr.

Algunos de los requerimientos visuales para cualquier software con estos elementos son:

- Simplificar: Muchas aplicaciones pecan de querer poner demasiadas cosas, pero un diseño limpio y cuidado, con lo estrictamente necesario, hace que la experiencia de usuario sea mejor.
- Identidad: Hay que saber diferenciar cada una de las cosas, pero todo tiene que mantener una misma identidad.
- Buen uso de los iconos: Aunque no hay que abusar de ellos, el uso de iconos es una buena idea. Además, tienen que ser suficientemente grandes para que se puedan tocar con facilidad.
- Tipografía: No debe ser muy grande ni muy pequeña, ser fácilmente legible y con buena resolución. El contraste con el fondo y la distancia de interlineado son factores fundamentales para que sea fácil de leer.

- Código de colores: Tenemos asimilado que un elemento rojo hace referencia a un error, así como uno verde a un acierto o confirmación. Mal usados pueden generar reticencia o falta de entendimiento por parte de los usuarios.
- Lenguaje visual: El usuario tiene que poder entender las opciones disponibles con una sola imagen, o con el uso de diferentes colores, entre otras técnicas.

Herramientas y lenguajes de programación visual

En la programación visual, los elementos del lenguaje de programación están disponibles en forma de bloques diseñados de manera gráfica, por lo que también se la llama programación gráfica. La apariencia y el etiquetado de los módulos permite identificar qué tarea en el flujo del programa pueden resolver. Los pictogramas sirven para orientar al usuario. Así, no se necesitan estructuras muy complejas ni un alto grado de abstracción.

Algunos de los lenguajes de programación visual son:

- Scratch: Proyecto del Grupo Lifelong Kindergarten del MIT Media Lab. Se ofrece de forma gratuita.
- NEPO, es un software de programación libre inspirado en Scratch. Fue creado por el MIT (Massachusetts Institute of Technology).
- Blockly: Proporciona un editor de programación visual al que se añaden aplicaciones Android, iOS y web. Blockly también utiliza bloques gráficos que encajan entre ellos.
- Grape: Es un entorno de desarrollo gráfico. Permite incluso a los principiantes programar con microcontroladores en pasos simples.
- App Inventor: Esta interfaz gráfica permite programar aplicaciones para teléfonos móviles Android con bloques gráficos.
- Ardublock: Este lenguaje de programación gráfica está especialmente diseñado para programar el microcontrolador Arduino sin introducir texto.
- Pure Data: Este lenguaje de programación visual está orientado tanto a flujos de datos como a los entornos de desarrollo.
- Lego Mindstorms: Su núcleo es la pieza de Lego programable: los motores eléctricos, sensores y piezas de tecnología propios de Lego permiten construir y programar robots y otros sistemas interactivos.

Fuentes de información

Alarcón, J. M. (Julio de 2021). Los conceptos fundamentales sobre Programación Orientada Objetos explicados de manera simple. Campus MVP. Recuperado de: https://www.campusmvp.es/recursos/post/los-conceptos-fundamentales-sobre-programacion-orientada-objetos-explicados-de-manera-simple.aspx

Andres (s / f). ¿Qué es la programación orientada a eventos? TIC. Recuperado de: http://contenidos.sucerman.com/nivel3/dispositivos/unidad1/leccion2.html

Cortijo, F. (s / f). Componentes. Elvex. Recuperado de: https://elvex.ugr.es/decsai/builder/intro/3.html

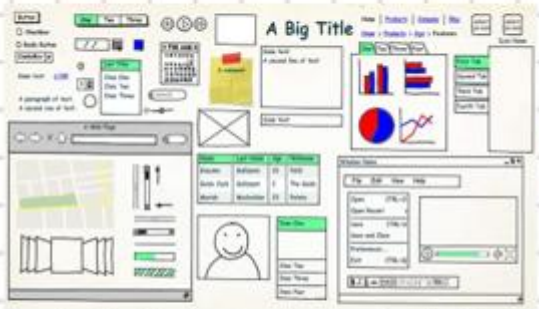
Desarrollo Web (Septiembre de 2020). Programación visual: la entrada más sencilla al mundo digital. IONOS. Recuperado de: https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/programacion-visual/

Sistemas (Junio de 2020). La importancia de una arquitectura distribuida. ILimit. Recuperado de: https://www.ilimit.com/blog/importancia-arquitectura-distribuida/

Gómez, C. (Junio de 2020). Requerimientos de software. Diario de QA. Recuperado de: https://www.diariodeqa.com/post/quiero-ser-qa-que-debo-aprender-requerimientos-de-software

Unitel (s / f). Los 10+1 Requisitos a tener en cuenta en el diseño de una APP. Unitel. Recuperado de: https://unitel-tc.com/10-requisitos-diseno-crear-app/

Tarea 2



Diseño.

Multimedia.

Interfaces.

Proceso de diseño.

Proceso de construcción de maquetado.

Diseño de Programación Visual.

Interactividad entre componentes.

Integración de contenidos y componentes.

Fuentes de Información

Equipo de redacción de Drew (Agosto de 2019). ¿Qué son y para qué sirven las integraciones de Software?. Recuperado de: https://blog.wearedrew.co/que-son-para-que-sirven-integraciones-de-softwares

Pingüino Digital (s/f). Interfaz de usuario. ¿Qué es, utilidad y para qué sirve?. Recuperado de: https://pinguinodigital.com/blog/interfaz-de-usuario/

Joel Frax (s/f). Proceso de diseño. Recuperado de: http://joelfrax.com/diseno/proceso%20de%20diseno.html#:~:text=El%20dise%C3%B1o%20es%20el%20proceso,de%20obtener%20los%20objetivos%20estipulados.

Rana negra desarrollo web (s/f). La importancia y el proceso del diseño y maquetación web. Recuperado de: https://www.rananegra.es/blog/importancia-proceso-diseno-maquetacion-web

Freepik (s/f). Imágenes de vectores. Recuperado de: https://www.freepik.es/fotos-vectores-gratis/multimedia

José A. Senso (Junio de 2015). Diseño web: wireframes y mackups. Recuperado de : https://blogs.ugr.es/tecweb/mockup-arquitectura-web/

Tarea 3

Desarrollo de aplicaciones web con programación visual.

¿Qué es la programación visual?

El término programación visual se refiere a la programación en la que se utiliza más de una dimensión para expresar la semántica. Los lenguajes de programación visual permiten a los usuarios crear programas mediante la manipulación de elementos gráficos, en lugar de especificarlos exclusivamente de manera textual.

¿Qué es el entorno de desarrollo integrado?

Es un programa informático compuesto por un conjunto de herramientas de programación, usualmente en un entorno visual, de forma que el programador no necesite introducir múltiples comandos para compilar, interpretar, depurar, entre otras. Habitualmente cuentan con una avanzada interfaz gráfica de usuario (GUI). Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

Algunos componentes del entorno de desarrollo

Son:

- Un editor de texto.
- Un compilador.
- Un intérprete.
- Un depurador.
- Un cliente.
- Posibilidad de ofrecer un sistema de control de versiones.
- Factibilidad para ayuda en la construcción de interfaces gráficas de usuario

¿Con qué herramientas cuenta el entorno de desarrollo?

Las propiedades de los componentes visuales de proyectos de Software

Son:

- **Accesibilidad:** Garantizar que la funcionalidad sea de fácil acceso para la mayoría de los usuarios, incluidos aquellos que tienen discapacidades.
- **Apariencia:** Es en gran parte la forma en que el usuario final se interrelaciona con la aplicación. Un elemento visual recibe su apariencia gracias al diseño que se le haya dado.
- **Comportamiento:** Son funciones asociadas al componente que pueden invocarse para que el componente realice distintas acciones.
- **Datos:** Es la información que se muestra en un elemento visual, el cual proporciona una manera accesible de ver y comprender

Los procesos de la programación visual

Son:

- **Análisis del problema:** Se analiza el problema considerando la especificación de los requisitos dados por el cliente.
- **Diseño del algoritmo:** Se diseña una solución que conducirá a un algoritmo que resuelva el problema.
- **Codificación:** La solución se escribe en la sintaxis del lenguaje de alto nivel y se obtiene un programa.
- **Compilación, Ejecución y Verificación:** El programa se ejecuta, se comprueba rigurosamente y se eliminan

- Colaboración en línea.
- Pruebas unitarias.
- Soporte para una gran variedad de frameworks.
- Características de navegación.
- Creación de interfaces.

tendencias, valores y datos en general.

- **Diseño:** Es esencialmente el aspecto del elemento. Para ello, se hace uso de la línea, la forma, el objeto, el espacio, la tipografía, la textura y el color.
- **Estilos:** Proporcionar un estilo consistente a los elementos visuales de una aplicación, le puede otorgar una identidad visual para que sea fácilmente reconocible.

todos los errores que puedan aparecer.

- **Documentación:** Registro de las fases del ciclo de vida del software, esencialmente el análisis, diseño y codificación, unidos a manuales de usuario y técnicos.

¿QUÉ SON LOS EVENTOS?

Los Eventos son las acciones sobre el programa, como por ejemplo:

- ✓ Clic sobre un botón
- ✓ Doble clic sobre el nombre de un fichero para abrirlo
- ✓ Arrastrar un icono
- ✓ Pulsar una tecla o una combinación de teclas
- ✓ Elegir una opción de un menú
- ✓ Escribir en una caja de texto
- ✓ Simplemente mover el ratón

Cuando se produce o dispara un evento en programación permite al usuario realizar una serie de acciones lógicas para un determinado programa. sobre un determinado componente elemento que presta un servicio de comunicación cuando se diseñan interfaces., se inicia un conjunto de acciones programadas por el usuario para ese evento concreto.

FUNCIONALIDAD DE LOS EVENTOS VISUALES

Este maneja una serie de componente elemento que presta un servicio de comunicación cuando se diseñan interfaces o controles con propiedades que se pueden cambiar para que el componente se muestre en pantalla de forma diferente o actúe de otra manera dentro de la aplicación. Para cambiar las propiedades de un componente, primero debemos seleccionar el componente deseado en la lista de componentes, también debemos tener en cuenta que hay algunos valores de propiedades de algunos componentes que no son modificables y otros que sí lo son.

Fuentes de información:

Andrés (s / f). ¿Qué es la programación orientada a eventos? TIC. Recuperado de: http://contenidos.sucerman.com/nivel3/dispositivos/unidad1/leccion2.html

Software de programación (s/f). Entorno de desarrollo integrado. Recuperado de: https://sites.google.com/site/softwaredeprogramacion2/entorno-de-desarrollo-integrado

Ivan Valencia Santiago (Enero de 2018.). Herramientas para programadores: entornos de desarrollo integrado. Recuperado de: https://seguridad.cicese.mx/dutic/32/Herramientas-para-programadores:-Entornos-de-desarrollo-integrado

Microsoft (Mayo de 2022). Accesibilidad (Conceptos básicos de diseño). Recuperado de: https://docs.microsoft.com/es-es/windows/win32/uxguide/inter-accessibility

CEAC (Marzo de 2018). Desarrollo de interfaces: como crear componentes visuales. Recuperado de: https://www.ceac.es/blog/desarrollo-de-interfaces-como-crear-componentes-visuales

Tableau (s/f). Los diferentes tipos de visualizaciones. Recuperado de: https://www.tableau.com/es-mx/learn/articles/data-visualization

Imborrable (Octubre de 2020). Identidad visual: Que elementos la componen y como aplicarla. Recuperado de: https://imborrable.com/blog/identidad-visual/

Informática y tecnología (s/f). Fases del proceso de programación. Recuperado de: https://sites.google.com/site/programacionvisualcetis50/Inicio/contenido/principios-de-programacion/fases-del-proceso-de-programacion

Tarea 4

Introducción

Los videojuegos se han convertido en un parte esencial en el mundo del entretenimiento, los cuales traen consigo efectos especiales, explosiones, acción trepidante y demasiadas cosas pasando en la pantalla. Su objetivo es abarcar al mayor público posible con un producto visualmente impresionante.

Un videojuego es un programa de computación, creado para el entretenimiento, basado ella interacción entre una o varias personas y un aparato electrónico (ya sea un ordenador, un televisor, una videoconsola, actualmente un teléfono celular), el cual ejecuta dicho videojuego. En muchos casos, estos recrean entornos y situaciones virtuales en los cuales el jugador puede controlar a uno o varios personajes (o cualquier otro elemento de dicho entorno), para conseguir uno o varios objetivos por medio de unas reglas determinadas. Este avance en el ámbito del entretenimiento permite al usuario convertirse de cierto modo, en un personaje que afronta un problema y que vive en un mundo ficticio.

Para crear un videojuego hay que seguir todo un proceso con muchas fases: además de la parte de programación, también hay mucho trabajo de diseño y creatividad que requiere conocimientos y habilidades digitales.

Es por ello que en la siguiente investigación se darán a conocer diversos conceptos, procesos e información relacionada con los videojuegos y sus derivados.

Conceptos de videojuegos

Un videojuego es un software creado para el entretenimiento en general y basado en la interacción entre una o varias personas y un aparato electrónico que lo ejecuta; este dispositivo electrónico puede ser una computadora, un sistema arcade, una videoconsola, un dispositivo portátil o un teléfono móvil, los cuales son conocidos como "plataformas".

Los videojuegos son interactivos; es decir, los usuarios se involucran activamente con el contenido. La interacción entre la o las personas y el videojuego se produce mediante un dispositivo denominado control, que permite convertir las órdenes del jugador en acciones dentro del juego. Los controles pueden ser muy diversos y su forma y funcionalidad varía según la plataforma.

Concepto de Game Designer

Es el profesional encargado de diseñar todos los elementos que componen el juego, desde el concepto, las mecánicas o los niveles, entre otros. Se encarga de diseñar la historia, los personajes, los escenarios o las reglas de un videojuego.

Algunas de sus funciones son:

- Define la mecánica del videojuego
- Gestiona proyectos con otros departamentos
- Delimita las reglas y niveles
- Crea los personajes
- Define el escenario
- Fija las recompensas
- Elige las habilidades
- Marca los patrones de interacción

Tipos de Game Designer

Producer: Es el diseñador principal. Define las líneas generales del mundo del juego, sabiendo cómo va y a donde al igual que todas las decisiones pasan por él.

Content Designe: Encargado del contenido. Crea los elementos del mundo, realiza muchas propuestas, tomándose solo las que se necesitan.

Game Writer: Escribe diálogos, guion. Cuenta la historia, hace el concepto.

System Designer: Crea las reglas del juego, el sistema. Es aquel que construye el gameplay.

Level Designer: Crea el espacio del juego y construye una narrativa por medio de este. Cuerpo del juego.

Usability Designer (diseñador gráfico especialista en feedback): Se encarga de que el sistema y todo tenga una lógica y funcione correctamente, se asegura que el juego se comunique con el usuario. Básicamente su centro es la comunicación.

Concepto de Storyboard

Es un compuesto de dibujos secuenciales que ilustran los planos de una obra audiovisual, mediante la cual se pueden previsualizar las escenas que van a grabarse. El storyboard incluye también una serie de indicaciones en el pie de cada fotografía. En ellas se destacan los detalles más complicados o los más difíciles de explicar.

La función principal del guion gráfico es hacer que todos los miembros puedan imaginar y previsualizar el resultado final de la obra. De esta manera, podrán trabajar de forma coordinada para llevar a buen puerto el proyecto.

Tipos de Storyboard

Storyboard animado: Pueden ser bocetos individuales con el fin de crear sensación de tiempo y movimiento. Si se le añade música y diálogo, las animaciones pueden dar un sentido de flujo visual y del tiempo mucho mayor. Las viñetas se convierten en bocetos individuales filmados con el fin de crear la sensación de movimiento. Estos incluyen diálogos y en ocasiones audio provisional, las animaciones permiten a un cineasta ver si la obra funciona visualmente.

Este tipo de guion gráfico es muy útil para detectar errores antes de la fase de realización audiovisual.

Storyboard digimatics. Sustituyen los bocetos con imágenes digitales unidas entre sí con el fin de crear sensación de tiempo y movimiento. Es similar al storyboard animado, ya que también utiliza imágenes digitales para crear la sensación de movimiento. Sirve para realizar películas de prueba y suele utilizarse en publicidad para previsualizar spots publicitarios y ver su impacto.

El objetivo último es presentar la idea al director de arte o al cliente final. Así, es posible realizar cambios sobre el guion sin que suponga un coste elevado.

Storyboard de miniaturas: Suele ser pequeño y reunido en una sola hoja de papel. se dibuja mucho más rápido que el resto de las variedades de storyboard y utiliza garabatos en el panel de acciones, así como bocetos antes de crear los detalles. Suelen ser esbozos iniciales que sirven de guía a los creativos para desarrollar un guion gráfico más detallado. Es la forma más sencilla y rápida de mostrar la esencia de un guion.

Storyboard tradicional: Son dibujos a lápiz o boli creados por un artista bajo la supervisión del productor o director. Seguirá las pautas indicadas por el cineasta, el cliente o productor. Puede representar las acciones y los personajes de forma más o menos realista e incluir anotaciones sobre los aspectos visuales y narrativos más importantes.

Tipos y características de motores de videojuegos

Crystal Space

Características

- Renderizado en Portales y Sectores
- Arboles BSP
- Zbuffering
- Radiosidad
- Detección de colisiones
- Lightmaps
- Bumpmapping
- Phong
- Gouraud

- Sprites 2D y 3D
- Superficies de Bezier
- Mipmapping

Fly 3D

Características

- Renderizado en Portales y Sectores
- Arboles BSP y PVS
- Sistema de Plugins
- A* optimizado
- Detección de colisiones
- Lightmaps
- Meshes animadas

Unreal

Características

- Renderizado en Portales y Sectores
- Arboles BSP, PVS, LOD
- Sistema de Plugins
- Detección de colisiones
- Lightmaps
- Meshes animadas
- Radiosidad
- Bumpmapping
- Phong y Gouraud
- UnrealScript
- Raytracing
- Escalabilidad
- Mipmapping
- Enveloped lighting

Genesis3D

Características

- Renderizado en Portales
- Arboles BSP y LOD
- Sistema de Plugins
- Detección de colisiones
- Radiosidad
- Luces multicolores y dinámicas

Torque V12

Características

- Renderizado en Portales y Sectores
- Meshes LOD
- Detección de colisiones
- Radiosidad
- Texture Mapping
- Lightmaps

Quake2

Características

- Renderizado en Portales y Sectores
- Arboles BSP
- Zbuffering
- Detección de colisiones
- Radiosidad
- Lightmaps
- Bumpmapping
- Phong y Gouraud
- Mipmapping
- Dll's

Irrlicht

Características

Motor 3D multiplataforma de alto rendimiento de código abierto para crear aplicaciones en tiempo real 3D. Sus características principales son ser fácil de utilizar, extremadamente rápido, extensible y libre de fallos.

JPCT

Características

Es un motor grafico 3D con API para Java. Requiere Java 1.1 o superior y puede ser usado para aplicaciones applets. Soporta software de renderizado así como hardware de renderizado vía OpenGL (Java 1.4).

Multiplataforma: W-buffer de 32 bits

Apocalyx

Características

Sencillo engine escrito en OPENGL, interface de scripting LUA, simulación del agua, ropa, etc.

Tipos y características de lenguajes de videojuegos

C++

Se trata del lenguaje más compatible con la mayoría de los motores de juego y tiene un tiempo de ejecución bastante rápido. Por otro lado, permite a los desarrolladores tener un control amplio sobre el hardware, la gestión de la memoria y los gráficos, y, aunque al principio puede resultar complejo de utilizar, una vez te haces a él, podrás manejar cualquier otro lenguaje.

C Sharp

Es un poco menos flexible y compatible que C++, pero algunos motores como Unity permiten programar con él y no está limitado a un determinado sistema operativo o plataforma; se pueden crear juegos para iOS, Android, Windows Play Station y Xbox.

Java

Se trata de un lenguaje frecuentemente utilizado y presenta muchas similitudes con C++. Su principal característica es la versatilidad, ya que se puede utilizar en todas las plataformas, dispone de gran cantidad de frameworks para el desarrollo 3D, ofrece módulos de código abierto y su modelo se puede actualizar constantemente.

JavaScript

Este es uno de los lenguajes más utilizados en el desarrollo de videojuegos web y de navegador. La mayoría de los motores de videojuegos son compatibles con JavaScript, y cuenta con múltiples frameworks para 3D y una gran variedad de bibliotecas. Además, algunos motores de videojuegos como Unity lo utilizan, por lo que podremos usarlo para crear todo tipo de scripts dentro del juego.

Python

A pesar de no ser un lenguaje de programación exclusivo para la creación de videojuegos, Python es un lenguaje muy flexible y potente para esto. Su ejecución es mucho más simple que la de otros lenguajes (permite plasmar ideas complejas con pocas líneas de código), y su framework Pygame permite a los desarrolladores crear prototipos de sus videojuegos de manera rápida y sencilla, y funciona prácticamente en todas las plataformas y sistemas operativos.

Lua

Lua es un lenguaje de programación sencillo, rápido y fácil de aprender. Compatible con lenguajes más complejos y de rápida ejecución, también se usa para aplicaciones web y procesamiento de imágenes. Este lenguaje es especialmente útil para proyectos independientes y programadores que estén empezando en la profesión.

Metodologías de desarrollo de videojuegos

Fase de Concepción

Todo comienza con una idea a partir de la cual se conformarán los aspectos fundamentales. Se determina el género o géneros del videojuego, cómo será el proceso de juego (game play), y también se constituye un guión gráfico (story board) en el que se tratan todo tipo de ideas preconcebidas que pueden ir adaptándose, como por ejemplo el estilo de los personajes, el ambiente, la música, etc.

Fase de Diseño

Se empieza definiendo los elementos que componen el juego. Se desarrolla la historia, se crean bocetos de guiones para determinar los objetivos, se deciden los personajes principales, el contexto, etc.

Utilizando estos esbozos de guiones los artistas se ponen manos a la obra para crear conceptos del aspecto del juego, la forma en que se visualizarán los personajes, los escenarios, objetos, etc. Su trabajo es presentar propuestas visuales para ir dando forma a la idea original. También se describen los elementos

sonoros de los que consta el juego: efectos de sonidos, ambientación, música, voces, etc. Paralelamente se especifica el funcionamiento general del videojuego, algo que depende del género, ya que señalan la forma en que las entidades virtuales interactúan dentro del juego.

Finalmente, se hace el diseño de la programación, que describe la manera en la que se implementará el videojuego, el lenguaje o lenguajes de programación que se utilizarán, las metodologías que se seguirán, etc.

Fase de Planificación

Esta etapa tiene como objetivo identificar las diferentes tareas para desarrollar el videojuego. Se reparte el trabajo entre los distintos componentes del equipo de desarrollo, se fijan plazos de entregas, se planifican reuniones de seguimiento, etc.

Fase de Producción

Se empieza la producción con el objetivo de crear el juego, como mínimo en una versión inicial o prototipo a mejorar gradualmente.

Si finalmente se logra ensamblar correctamente todas las piezas entonces esta fase culmina (por ahora).

Fase de Pruebas

En esta etapa se corrigen los errores del proceso de programación y se mejora la jugabilidad a medida que se prueba el juego.

Generalmente encontraremos dos tipos: las pruebas alpha, realizadas por un pequeño grupo de personas generalmente involucradas en el desarrollo, y las pruebas beta, realizadas por un equipo externo de jugadores. Las primeras tienen el objetivo de corregir defectos graves y mejorar características fundamentales no contempladas en el documento de diseño, mientras que las segundas se enfocan en detectar fallos menores y perfilar la experiencia de usuario.

Fase de Distribución/Márketing

Es el proceso de crear las copias del juego ya finalizado y llevarlo a las tiendas (ya sean físicas o digitales) para que los jugadores puedan comprarlo o hacerse con él.

Fase de Mantenimiento

La fase de mantenimiento es el momento de arreglar nuevos errores, mejorarlo, etc. Esto se hace sacando parches o actualizaciones al mercado.

Sin embargo, es también una oportunidad para seguir sacándole partido. Ya sea en forma de micro transacciones, suscripciones de pago o incluso con expansiones completas que añaden nuevas características al videojuego sin modificar en profundidad el motor de este.

Proceso de diseño de interfaces de videojuegos en 2d y 3d.

Proceso de interfaz en 2D

- La interfaz gráfica de usuario requiere las siguientes características:
- Brindar funcionalidades típicas de un programa de escritorio, tales como abrir y guardar archivo.
- Utilizar el método de manipulación directa de componentes gráficos sobre un área de trabajo, realizar conexiones y configurar sus propiedades.
- Permitir la posibilidad de simular, graficar datos y programar la Unidad Controladora.
- Ser accesible remotamente por medio de protocolo TCP/IP.

Para lograr estas características, es necesaria la selección de un lenguaje de programación que permitiera la manipulación de imágenes y gráficos en 2D, así como el desarrollo de interfaces gráficas de usuario; la programación de sistemas Web y de red, facilidad de implementación, buena documentación, multiplataforma, variedad de tecnologías y bajo licencia de software libre.

Proceso de interfaz en 3D

Hay una gran variedad de sistemas para crear interfaces en 3D, en este caso hablaremos del proceso de creación de una interfaz en el sistema 4D-GIFT

Se suministra en una misma plataforma dos niveles funcionales: por un lado, puede ser considerado como un sistema de fácil uso para la generación de interfaces 3D independientes de la aplicación. Por otro lado, la estructura de datos subyacente permite sobrepasar la separación entre los objetos gráficos pertenecientes al interfaz y a la aplicación. De esta forma los objetos del interfaz son accesibles, pudiendo aprovechar las formas definidas en la aplicación, así como utilizar objetos provenientes de un modelado independiente, que pueden ser integrados en la escena como controles activos.

Desarrollo de prototipos de videojuegos

Concepto, tipos y características de los motores de videojuego.

Un motor de videojuego es un framework o un conjunto de herramientas que ayudan a agilizar el proceso de desarrollo de un videojuego los cuales permiten el diseño, la creación y la representación de un videojuego.

Los motores proveen herramientas al programador, que le permiten dedicar menos tiempo a aspectos poco importantes para la idea general del videojuego.

Algunas de las funcionalidades más importantes son:

El motor de físicas

Es el que hace posible aplicar aproximaciones físicas a los videojuegos para que tengan una sensación más realista en la interacción de los objetos con el entorno. En otras palabras, es el encargado de realizar los cálculos necesarios para que un objeto simule tener atributos físicos como peso, volumen, aceleración, gravedad.

El motor de sonido

Es el encargado de cargar pistas, modificar su tasa de bits, quitarlas de reproducción, sincronizarlas entre otras cosas.

El scripting

Todos los motores de videojuegos tienen un lenguaje de programación que permite implementar el funcionamiento de los personajes y objetos que forman parte del videojuego.

Dentro de las diferentes opciones de motores de videojuegos podemos distinguirlos en populares y motores propietarios o privados que son los creados por empresas importante de videojuegos para diseñar sus títulos más populares.

Integración de motores de videojuegos con programación visual de acuerdo con los requerimientos del videojuego.

Dirección de Arte según la Tecnología Gráfica usada

La tecnología de representación de gráficos o el engine o motor del juego que elijamos van a ser muy importante a la hora moldear las características visuales que queremos brindar a nuestro juego. Idealmente estas incógnitas deberán despejarse durante la fase de preproducción. Igualmente las limitaciones y requerimientos técnicos existentes en la plataforma objetivo también influirán mucho en el tipo de gráficos que podemos crear.

Arte 2D según Tecnología usada

Píxel Art: Los gráficos del juego se construyen como mosaicos. Este estilo es la herencia de las limitaciones gráficas que tenían los primeros ordenadores y videoconsolas, así que estamos hablando de una de las formas más tempranas y básicas de hacer gráficos en un juego.

Se usa con la intención de dar un aspecto retro y/ nostálgico. En la actualidad podemos encontrar juegos con este aspecto visual pero que funcionan sobre un motor gráfico 3D, donde los sprites se aplican y se mueven mediante planos 3D.

Ilustración 2D: La evolución del Píxel Art, donde el arte ya no solo no se restringe al número de pixeles en pantalla si no que tiende a parecerse más a una ilustración, dando un resultado similar a lo que podemos ver en los dibujos animados. En este caso también es habitual hacer uso de un motor 3D.

Vectorizados: Es la antesala de los gráficos en 3D, puesto que su base es la misma, solo que simplificando el sistema de coordenadas de 3 a 2. Hoy en día no se usan demasiado, aparte de en juegos con tecnología Flash.

Arte 3D según Tecnología usada

Polígonos: La mayor parte de los juegos hoy en día usan esta tecnología, que nos permite aplicar de forma fácil perspectiva y profundidad. Aquí hablamos de la representación de vértices y texels en un espacio tridimensional.

Voxels: Realmente una subcategoría de gráficos 3D, donde la unida mínima de representación pasa a ser una caja 3D.

Gráficos 2.5D, Pseudo 3D o Cámara Fija: ya sean sprites aplicados a planos 3D o modelos 3D completos presentados a través de una cámara con una vista fija, dando la sensación de juego 2D (ya sea un tipo de perspectiva isométrica, lateral, etc.).

Transición narrativa y lenguaje visual de videojuegos.

El arte, además de tener que ser interesante y atractivo también debe apoyar al diseño del juego, adaptándose y dando soluciones a las necesidades del gameplay, potenciando las mecánicas de juego y siendo un fin para comunicarse con el jugador. Por lo tanto, el diseñador deberá prestarle especial atención y tener en cuenta también como va a ser el apartado gráfico desde el principio, trabajando junto con el equipo de arte para que ambas áreas se complementen. Algunas de las características de diseño y mecánicas que pueden influir en los elementos visuales son:

- Dar indicaciones y feedback al jugador de lo que está ocurriendo en el juego
- Facilitar la lectura y navegación en el mundo del juego (por ejemplo, cuando guiamos al jugador a través de un pasillo haciendo uso de la iluminación).
- Comunicar y hacer sentir al jugador de una manera concreta: claustrofobia, miedo, grandiosidad, dinamismo, etc.

Ambientación y diseño de personajes

Son las bases que normalmente usamos para contar una historia en el juego y para definir la narrativa, siendo habitual que el hilo conductor de la misma sea nuestro avatar.

Proceso de desarrollo de videojuego acorde a los elementos de programación visual.

Planeación

Un videojuego nace de una idea. A partir de este momento inicia la etapa de planeación.

Una vez establecido, todos las personas y equipos de los departamentos involucrados pueden empezar a probar el concepto. Aquí se toman las ideas que se han compartido y se analiza la viabilidad del proyecto para ser producido por el estudio. En este punto surgen preguntas como:

- ¿Cuál es el costo aproximado del proyecto?
- ¿Es necesario un nuevo motor de juego?
- ¿Cuántas personas necesitamos en el equipo?
- ¿Cuál es la fecha estimada de lanzamiento?

La prueba de concepto es vital para el éxito de un juego, pues pone en perspectiva cuáles son las posibilidades del proyecto y avanzar, con ellas en mente, el desarrollo.

Preproducción

En esta etapa, los escritores, diseñadores, artistas, ingenieros, leads de proyecto y otros departamentos trabajan y colaboran entre sí para definir la manera en que darán vida al videojuego.

Durante este proceso colaborativo se llevan a cabo juntas y discusiones entre varios equipos. Por ejemplo: escritores y leads trabajando la narrativa de la historia; ingenieros estableciendo qué pueden realizar con la tecnología disponible; diseñadores y artistas asegurándose que la paleta de colores, visuales y arte sigan la misma línea preestablecida; etc.

Es en este proceso donde los desarrolladores trabajan prototipos de personajes, ambientes, interfaces, etc.

Producción

- Los modelos de personajes se diseñan y renderizan hasta que luzcan como deben.
- El diseño de audio crea todos los sonidos del mundo del juego.
- Los diseñadores de nivel crean los ambientes de forma que sean atractivos para los jugadores.
- Se graba el doblaje con los actores.
- Los programadores escriben enormes códigos para darle vida a los elementos del juego.

Esta etapa no tiene un tiempo definido. Dependiendo del tipo de producción, esta etapa puede tomar muchos años para completarse debido a muchos factores, como, por ejemplo: cambios de forma constante; rehacer partes ya terminadas que no satisfacen los objetivos del proyecto.

Periodo de prueba

Cada elemento, detalle y mecánica del juego debe someterse al control de calidad antes del lanzamiento. Para que un videojuego esté listo para su versión alpha, primero debe pasar por las manos de testers para que se identifiquen cuestiones como:

- ¿Hay áreas o niveles con muchos bugs?
- ¿Todo se está renderizando correctamente?
- ¿El personaje se queda atorado permanentemente en un lugar?

- ¿Los diálogos son atrapantes y realistas?

Generalmente hay distintos tipos de testers. Unos se enfocan en tratar de “romper” el juego; otros analizan la dificultad del juego para ver si es muy fácil o difícil. Este equipo tiene que asegurarse que el videojuego sea divertido y atractivo para generar ventas.

Pre-lanzamiento

Se tiene que elaborar una estrategia de mercadotecnia para vender el juego.

Teasers, imágenes promocionales, tráilers cinematográficos, gameplays, streamings de influencers de videojuegos y demás, son estrategias que los estudios lanzan para promocionar el juego y que la audiencia lo conozca.

Lanzamiento

Por fin se define una fecha de lanzamiento y el equipo debe trabajar para pulir todos los detalles que sean necesarios.

Durante esta etapa, los desarrolladores hacen una lista de bugs que tienen que erradicar, de mayor a menor gravedad. Comienzan con aquellos bugs que pueden “crashear” el juego, hasta llegar a aquellos que representan problemas menores.

Cuando el juego esté lo suficientemente limpio, es hora de lanzarlo al público.

Post-lanzamiento

Los jugadores comienzan a identificar bugs u otros errores dentro del videojuego, por lo que los estudios tienen que solucionar estos problemas.

Además, se ofrecen actualizaciones de software y, en muchos casos, DLC, para que los jugadores tengan contenido extra dentro del mundo de este nuevo videojuego.

Conclusión

Los videojuegos tienen la capacidad de influenciar al usuario y provocar todo tipo de emociones. Justamente por esto, aquellos que son independientes son esenciales para el futuro de la industria. Ya que sus desarrolladores no están limitados por distribuidores de ninguna clase, y pueden plasmar su visión con total libertad.

A nivel personal, este trabajo de investigación y desarrollo nos ha formado bastante sobre posibles casos futuros en el campo de los videojuegos. Al mismo tiempo, me ha hecho darme cuenta del trabajo que lleva solo la parte visual de un simple juego.

Me interesa también sobre todo la parte de la comercialización, ya que me parece interesante ver cómo juegos con mecánicas sencillas pueden llegar a rentabilizarse tanto con millones de descargas.

Creo que es necesario saber moverse bien en términos de monetización, ya que creo que hoy en día un diseñador puede crear una app sin necesidad de la ayuda de un programador, y por lo tanto necesita saber cómo monetizar su trabajo.

El proceso para crear videojuegos es muy largo, incluso podría tardar años en desarrollar uno, pero es muy interesante el desarrollar uno y el nuevo campo de posibilidades que se ha abierto gracias a la fuerte demanda de videojuegos.

Fuentes de información

Ceibal formación. (s / f). ¿Qué es un videojuego?. Ceibal. Recuperado de: https://blogs.ceibal.edu.uy/formacion/faqs/que-es-un-videojuego/

Judith Vives. (02/05/2018). ¿Cómo se hace un videojuego?. La vanguardia. Recuperado de: https://www.lavanguardia.com/vida/juniorreport/20180502/443116327302/como-videojuego.html

Icedancer. (s / f). Videojuego. Wiki juegos. Recuperado de: https://videojuegos.fandom.com/es/wiki/Videojuego

Comunidad Campus S.L. (22/06/2022). Tipos de Story board. Treintaycincomm. Recuperado de: https://35mm.es/tipos-storyboard/

Productora audiovisual. (16/01/2019). ¿Qué es un story board?. Dosis videomarketing. Recuperado de: https://www.dosisvideomarketing.com/que-es-storyboard/

Micky Jerzy. (18/06/2012). Motores gráficos. Slideshare. Recuperado de: https://es.slideshare.net/FADSXD/motores-graf

Library. (s / f). Diseño de la interfaz gráfica de usuario. Library. Recuperado de: https://1library.co/article/dise%C3%B1o-interfaz-gr%C3%A1fica-usuario-etapas-dise%C3%B1o-entorno-desarrollo.q5mgng7y

Mariano Perez. (Junio 1996). Desarrollo de interfaces de usuario basados en 3D. El sistema 4D-gift. Researchgate. Recuperado de: https://www.researchgate.net/publication/318787677_DESARROLLO_DE_INTERFACES_DE_USUARIO_BASADOS_EN_OBJETOS_3D_EL_SISTEMA_4D-GIFT

Alberto Carrasco Carrasco. (04/07/2018). ¿Qué es un motor de videojuegos?. Blogs. Recuperado de: https://blogs.upm.es/observatoriogate/2018/07/04/que-es-un-motor-de-videojuegos/

Uriel Ruelas. (13/07/2017). ¿Qué es un motor de videojuegos? (Game engine). Coding or not. Recuperado de: https://codingornot.com/que-es-un-motor-de-videojuegos-game-engine

Pedro P. Fernandez. (13/03/2018). Producción visual en videojuegos Parte I. Medium. Recuperado de: https://medium.com/@vancorso/producci%C3%B3n-visual-en-videojuegos-parte-i-492755789b68

Uniat. (s / f). Etapas del desarrollo de videojuegos. Uniat. Recuperado de: https://www.uniat.edu.mx/etapas-del-desarrollo-de-videojuegos/