Tarea 1

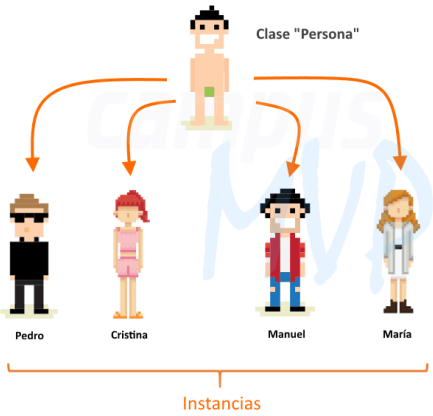
Realiza un resumen del Análisis de la programación visual

Conceptos de programación orientada a objetos.

Clases, Objetos e Instancias.

Una clase es una plantilla. Define de manera genérica cómo van a ser lo objetos de determinado tipo. Debe de tener una serie de atributos y una serie de comportamientos que se implementan como métodos de la clase. Una clase por sí sola no sirve de nada, pues no es más que un concepto, sin entidad real.

Para poder utilizar una clase en un programa lo que hay que hacer es instanciarla. Instanciar una clase consiste en crear un nuevo objeto concreto de la misma. un objeto es ya 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.



Encapsulación

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.

Así, internamente tenemos un dato que es el nombre de la persona y accedemos a él a través de la propiedad pública Nombre que define la clase que representa a las personas. De este modo damos acceso sólo a lo que nos interese y del modo que nos interese.

Abstracción

Nos abstrae de la complejidad que haya dentro dándonos una serie de atributos y comportamientos que podemos usar sin preocuparnos de qué pasa por dentro cuando lo hagamos. Así, una clase debe exponer para su uso solo lo que sea necesario. Cómo se haga "por dentro" es irrelevante para los programas que hagan uso de los objetos de esa clase.

Herencia

Una 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. Todos los objetos de estas clases heredan las propiedades y los métodos. Lo bueno de la herencia es que podemos reutilizar todo lo que tuviésemos en la clase base.

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.

El polimorfismo nos permite utilizar a los objetos de manera genérica, aunque internamente se comporten según su variedad específica.

El polimorfismo puede ser más complicado que eso ya que se puede dar también mediante la sobrecarga de métodos y, sobre todo, a través del uso de interfaces, pero el concepto es el que acabo de explicar.

Características y aplicaciones de eventos.

Modelo de la programación de computadoras, donde se utilizan los eventos que suceden para la determinación del flujo de control de un programa.

No es un tipo de tecnología o lenguaje de programación, sino un enfoque que se implementa durante la etapa de desarrollo del producto.

Básicamente, separa la lógica de procesamiento de eventos del resto del código de un programa.

Usando un procedimiento apropiado de manejo de eventos para tratarlos, normalmente mediante una llamada a una función o método.

Teóricamente, el estilo de esta programación es compatible con todos los lenguajes de programación, aunque puede ser diferente en la forma de implementarse.

En general, en una aplicación controlada por eventos hay un bucle principal que “escucha” los nuevos eventos entrantes, activando una llamada a una función cuando estos se detectan. Por tanto, su funcionamiento se enfoca en los eventos, decidiendo estos qué ejecutar y en qué orden.

Dependencia de eventos

Los eventos en sí pueden ser desde aceptar o rechazar una solicitud de préstamo, denominado evento de alto nivel, hasta que un usuario presione una tecla, que es un evento de bajo nivel.

Orientada al servicio

Se utiliza para escribir programas diseñados para el servicio sin ralentizar la computadora, ya que la orientación al servicio solo consume poco poder de procesamiento. Además, los servicios se ejecutan por lo general en el trasfondo del sistema operativo.

Eventos

Condición que surge durante la ejecución de un programa y que requiere alguna acción por parte del sistema. Cada evento es diferente por naturaleza, algunos requieren que el programa recobre y muestre cierta información, y otros que se inicien algunos cálculos y cambios de estado.

Los eventos incluyen al ratón, al teclado, una interfaz de usuario y las acciones que se deben activar en el programa cuando ocurran. Esto significa que el usuario debe interactuar con un objeto en el programa, como hacer clic en un botón del ratón, usar el teclado para seleccionar un botón.

Controlador de eventos

Es un tipo de función o método que ejecuta una acción específica cuando se activa un evento determinado.

Por ejemplo, podría ser un botón que cuando el usuario haga clic en él muestre un mensaje y cuando vuelva a hacer clic en ese botón cierre el mensaje.

Funciones de activación

Son funciones que deciden qué código ejecutar cuando se produce un evento específico. Se utilizan para seleccionar qué controlador de eventos emplear al producirse un evento.

Tiempo controlado

Es una tarea preestablecida por hacer.

La actualización de Windows es un ejemplo de tiempo controlado, donde el usuario puede establecer cuándo actualizar o cuándo verificar y descargar la actualización.

Características de componentes y métodos visuales y no visuales.

Componente visual

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

los componentes visuales podemos dividirlos a su vez en dos tipos:

Componentes interactivos: permiten que el usuario final los manipule, ya sea introduciendo datos, seleccionado elementos, etc. De forma que estos componentes pueden recibir el foco, así como los eventos propios del teclado y del ratón.

Normalmente, el propio sistema operativo es el encargado de dibujar el aspecto del componente, haciendo el componente las llamadas correspondientes para que este aspecto cambie.

Componentes gráficos: el propio componente es el encargado de dibujar en la pantalla lo que crea oportuno, bien a través de las funciones básicas de API de Windows o bien a través de otras librerías gráficas.

Estos componentes, no suelen recibir eventos del usuario final, aunque si eventos del propio programador, ya que su cometido no suele ir más allá de mostrar ciertos gráficos o imágenes en la pantalla.

Dado que un componente es un objeto como otro cualquiera, podremos aplicar en él todas las técnicas de la orientación a objetos: encapsulación, herencia y polimorfismo.

La técnica de la herencia, aplicada a los componentes, nos permite personalizar cualquier componente, o porque queremos ampliar las posibilidades del componente.

Componentes no visuales

Los componentes no visuales se pueden colocar en los formularios de la misma manera que los controles, aunque en este caso su posición es irrelevante.

Los componentes no-visuales deben heredar directamente de TComponent, ya que este proporciona las características básicas.

TControl: se trata de la clase padre para todos los componentes visuales, ya sean gráficos o no.

Los componentes no visuales incluyen Time Control, SerialPort y ServiceController, entre otros.

Procesos de desarrollo visual en proyectos distribuidos y de escritorio.

¿Qué es gestión visual?

Conjunto de técnicas orientadas a transmitir información:

- De manera ordenada
- Preferiblemente de manera visual (iconos, símbolos, gráficos)
- De forma auto explicativa
- Que promueva la toma de decisiones y la mejora continua

El objetivo principal es gestionar nuestro trabajo, definiendo:

- Como estamos ahora
- Que hay que hacer
- Quien lo va a hacer
- En qué orden se van a hacer

Y todo aquello de manera dinámica.

Para potenciar el trabajo en equipo:

- Facilita el conocimiento por parte de todo el equipo de trabajo
- Genera un mayor sentido de pertenencia en el proyecto
- Proporciona una oportunidad para que unos miembros informen a otros
- Permite obtener el feedback de la información

Para facilitar la gestión de conocimiento

- Reduce los silos de la información mas importante
- Disminuye los problemas producidos por a falta de comunicación
- Muestra el trabajo en progreso del proyecto
- Ayuda a identificar
    - Riesgos en el proyecto
    - Defectos en procesos o en entregables
    - Detectar cuellos de botella y bloqueos

Requerimientos visuales de proyectos distribuidos y de escritorio

Se pueden utilizar otros lenguajes de modelado como UML para especificar

Los diagramas de UML más utilizados en el modelado de sistemas distribuidos son:

Diagramas de caso de uso, de secuencia, de estado, de actividades, de componente y de despliegue.

Tableros físicos:

Empleado en murales, pizarras, paredes o ventanales.

Es importante lograr una lectura cómoda a 2-3 metros de distancia

Resulta habitual emplear combinación de graficas impresas y material elaborado a mano.

Estratégicamente situado, facilita la visión de todo el equipo

Fomenta las conversaciones espontaneas sobre el estado del proyecto

El material empleado facilita la creatividad y el pragmatismo

Resulta muy fácil de poner en marcha

Tableros virtuales:

Abordable como un tablero Kanban o como un dashboard

Es muy importante la usabilidad, preferible acceso web y responsive desing

La gestión de archivos adjuntos, foros y discusiones potencial en el uso tradicional de tarjetas

No requieres espacios físicos o ubicaciones concretas

Permite consultas en cualquier miento y lugar

Ideal para equipos distribuidos o con alta movilidad

Resulta más fácil actualizar que un tablero físico

Kanban:

Tablero de gestión de tareas, organizadas por columnas

Cada tarea en una nota o tarjeta

Cada estado en una columna

Posibilidad de emplear codificación de colores o divisiones horizontales par mejorar su estructura.

Herramientas y lenguajes de programación visual.

En la programación visual se emplean bloques gráficos con diferentes colores y formas geométricas que pueden acoplarse o enlazarse intuitivamente del mismo modo que un puzle o un juego de construcción.

La programación visual es un lenguaje de programación que permite a los humanos describir procesos mediante ilustraciones. Mientras que un lenguaje de programación típico basado en texto hace que el programador piense como un ordenador, un lenguaje de programación visual permite al programador describir el proceso en términos que tengan sentido para los humanos.

El enfoque gráfico en la programación permitirá al ordenador procesar las representaciones espaciales en dos o más dimensiones. Esta programación gráfica se denomina Lenguaje de Programación Visual en el que se definen imágenes mediante un lenguaje basado en texto.

Se utiliza con frecuencia en los diseños de sistemas de ingeniería para traducir la información de los eventos y las vibraciones durante las pruebas de los motores de automóviles en una lectura visual.

Las herramientas de programación gráfica se utilizan para emplear los diagramas de bloques, los instrumentos virtuales y los conectores. También puede utilizarse para supervisar y controlar el proceso automatizado. Además, puede analizar algunas funciones matemáticas como el uso de procesamiento insignificante. Permitirá también a los usuarios tener acceso a bases de datos de información sobre su terreno, edificios y demografía o utilizarlo en el diseño de sistemas celulares.

Scratch (MIT): Permite el desarrollo de aplicaciones interactivas a modo de aprendizaje como pequeños juegos, etc. También permite el desarrollo de aplicaciones para controlar dispositivos robóticos Lego

Bolt (Ludiq): Plugin de Unity incluido gratuitamente a partir de la versión 2019 para el desarrollo visual de scripts como alternativa a C#

ArduinoBlocks (Didactronica): Permite el desarrollo de scripts para ejecutarse en placas de desarrollo Arduino de modo alternativo a C++

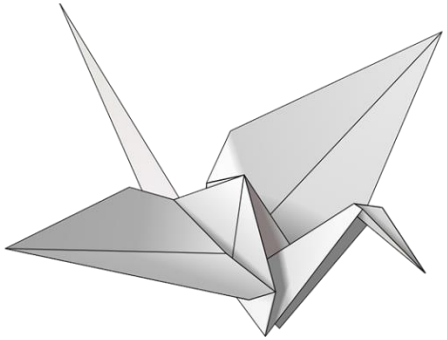
MakeCode (Microsoft): Plaraforma visual de desarrollo para múltiples micro-dispositivos robóticos de modo alternativo a Javascript / Python.

AppInventor (MIT): Plataforma online que permite el desarrollo visual de aplicaciones para dispositivos móviles.

Tarea 2

Realiza un mapa mental del tema Diseño de programación visual abarcando los subtemas

Relaciones visuales, sistémicas o geométricas

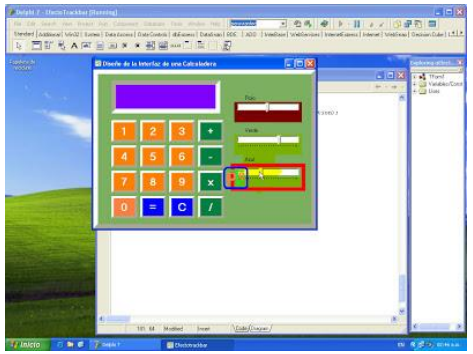


Uso de reglas

Concepto de diseño, multimedia e interfaces de programación visual

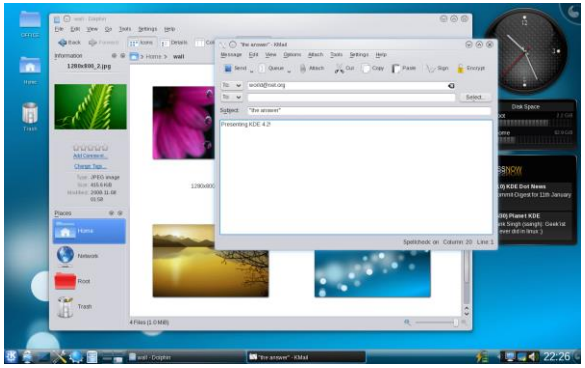
Medios digitales

Experiencia de usuario práctica, rica y efectiva.



Requieren del lenguaje multimedia para ser explotados

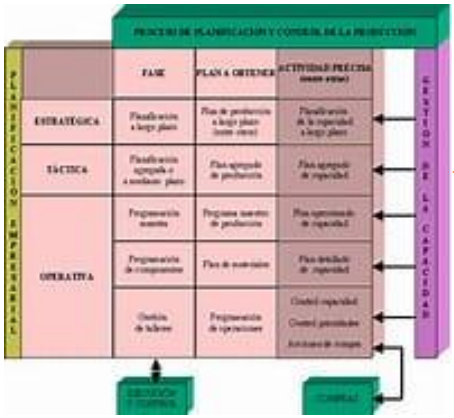
Forma en la que interrelacionemos con el usuario



Aparece el concepto de la programación por componentes

Se basa el diseño

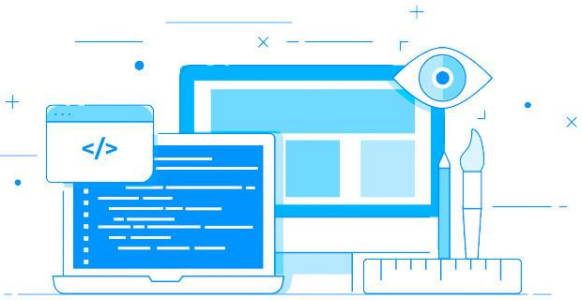
Se construye el proyecto



Decisión por decisión

Proceso de diseño e integración de contenidos y componentes.

Procedimiento organizado



Sistema de planificación aplicable

Estrategia flexible y puede tomar forma con base a las metas

Descomposición de sistemas ya confirmados en componentes funcionales o lógicos

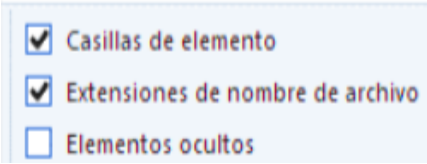
Sistema en plataforma modular

Interactividad entre componentes

Las etiquetas pueden presentar modificadores que se llaman atributos que permitirán definir diferentes posibilidades de la instrucción

Presentar documentos de forma más o menos atractiva al destinatario final

El usuario final puede dar su opinión de la página al autor



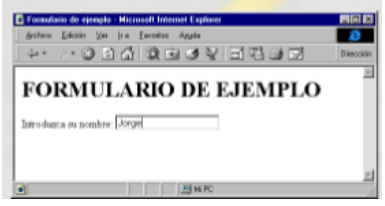
Las casillas de verificación, en algunas ocasiones el programador de la aplicación determina que el control permita solo 2 estados.

Marcado

Desmarcado

Los botones; utilizados para activar alguna función.

El propósito de la caja de texto es permitir al usuario la entrada de información textual para ser usada por el programa



Diseñar la parte grafica del sitio

Reconocer la estructura de bloques del documento

Organizar el archivo layout para su recorte

Recortar y exportar imágenes

Generar la estructura HTML del documento

Proceso de construcción de maquetado.

Replicar el proceso en todas las paginas del sitio

Volcar los contenidos de texto de imágenes en la estructura HTML

Estilizar los contenidos por medio de CSS

Testear el maquetado en diferentes navegadores

Tarea 3

Realiza un mapa conceptual del tema Desarrollo de aplicaciones con programación visual.

Eventos visuales

Los elementos se arrastran y sueltan en el flujo del programa para integrarlos.

Paradigma en el que tanto la estructura como la ejecución de los programas van determinando por los sucesos que ocurran en el sistema.

Definir eventos que manejaran el programa y las acciones que se realizan al producirse cada uno de ellos.

Conjunto de arreglo espaciales de símbolos de texto y gráficos con una interpretación semántica que es usada para comunicar acciones en un ambiente.

Componentes y herramientas visuales de entornos de desarrollo

Conjunto de procedimientos y herramientas que se utilizan para desarrollar un Código fuente o programa. Herramienta de desarrollo de software utilizado para escribir, generar, probar y depurar un programa

NETBEANS: funciona en diversas plataformas que permite agilizar el desarrollo de grandes aplicaciones de diversa índole soportando una gran cantidad de lenguajes de programación.

ECLIPSE: plataforma de desarrollo para aplicaciones tanto de escritorio como web, flexible en su configuración y de fácil adaptación a los requerimientos de desarrollo mediante la incorporación de plugins adicionales de acuerdo con las diferentes necesidades.

BUILDER C++: desarrollador de aplicaciones en entornos gráficos visuales mediante la programación orientada a objetos. Acelera radicalmente el desarrollo de interfaces de usuario.

DELPHI: entorno de desarrollo flexible y robusto que permite el desarrollo de sistemas y aplicaciones complejas que incluye herramientas para facilitar la estructura de código y su diseño mediante el uso de componente visuales.

Propiedades de los componentes visuales de proyectos de software: accesibilidad,

apariencia, comportamiento, datos, diseño y estilos.

La **Accesibilidad** se refiere al grado en el que las propiedades de un programa de ordenador o sistema informático permiten que sea utilizado por la variedad más amplia de usuarios.

La **Apariencia** de los componentes visuales tiene una representación gráfica en tiempo de diseño y ejecución (botones, barras de scroll, cuadros de edición, etc.)

En el **Comportamiento** puede ser introduciendo datos, seleccionando elementos, eventos del teclado y el ratón, dibujar en la pantalla.

El **Diseño** de los componentes visuales se representa de diferente manera como texto, imagen, mask, efectos, botón, casilla de verificación, deslizador, barra de desplazamiento, desplegable, campo de input, vista Scroll.

El **diseño** de un componente visual puede ser lineal, con forma, tiene color y textura.

Reconocer la funcionalidad de los eventos visuales.

Menos costoso

Proporciona a los usuarios de negocios una solución de desarrollo de software gráfico sin código y de bajo código.

Bases de conocimiento accesible

La simplicidad y su fácil desarrollo reducen la dependencia de los desarrolladores d software expertos, es relativamente fácil de aprender para la mayoría de los desarrolladores con una sencilla interfaz de arrastrar y soltar.

Desarrollo de software más rápido

Permiten el desarrollo rápido de aplicaciones en sus plataformas de bajo código y sin código. Esto es útil para las empresas que necesitan velocidad y flexibilidad para mantenerse al día con la demanda y en un rango más alto con su competencia.

Curva de aprendizaje

Las herramientas de programación visual suelen utilizarse para introducir a los jóvenes estudiantes en el mundo del desarrollo de software.

Proceso de programación visual

Conceptualización

Teniendo la idea, es el momento para determinar la necesidad o el problema que puede solventar dicha aplicación.

Debemos establecer el valor añadido que puede ofrecer nuestra aplicación, su utilidad para los usuarios y los objetivos que deseamos alcanzar con ella.

Definición

Se especifica y determina todas las características.

Se definirán al detalle las funcionalidades que la aplicación debe poseer para alcanzar las metas propuestas y cumplir con el cometido para el que ha sido creada.

Diseño

Se debe elaborar un wireframe donde podamos visualizar cómo quedarían las diferentes pantallas de la aplicación, antes de comenzar a diseñar.

Desarrollo

Dar forma a la aplicación y crear la estructura básica para el funcionamiento de esta.

Hay que considerar que lenguaje de programación se va a utilizar, con qué sistema operativo será compatible, que recursos gráficos se usaran y sus funcionalidades.

Hacer prototipos que pueden ser probados a medida que se avanza en la creación de la aplicación para testearla, revisarla y mejorar la app.

Tarea 4

Realiza una investigación tomando en cuenta los siguientes temas y puntos.

Conceptos de videojuegos

Conceptos y tipos de Game Designer, storyboard

En etapas iniciales del desarrollo se puede encargar de definir el concepto de juego, el universo o la historia.

En etapas más avanzadas se encarga de decidir que reglas, objetivos o mecánicas harán el juego divertido para el jugador. Su labor principal es pensar, documentar y comunicar las decisiones de diseño que convierten un videojuego en una experiencia divertida.

Profesional que se encarga del diseño del contenido del juego y de elaborar las mecánicas de este en periodo de pre-producción y del diseño del gameplay, escenarios, storyline y personajes en periodo de producción.

Actividades principales:

- Tener ideas originales
- Construir y conceptualizar proyectos
- Crear narraciones interactivas: historias, tramas y personajes
- Desarrollar la mecánica del juego: mapas escenarios y nivel de dificultad
- Diseñar la interfaz del usuario: control y menús

Con ayuda de STORYBOARD se representan, en forma de esquema, cómo se va a desarrollar una escena.

Ayuda a visualizar ideas y conceptos o cómo se interactúa con un cliente. También se utiliza en publicidad, para ilustrar los anuncios, y es una herramienta muy útil tanto para directores de arte como para diseñadores gráficos.

Es un conjunto de viñetas, en ellas se representa de forma grafica y sencilla distintos elementos.

Tipos y características de motores de videojuegos y lenguajes de videojuegos

Un motor de videojuego hace referencia a una serie de librerías de programación que permiten el diseño, la creación y la representación de un videojuego.

Para elegir correctamente un motor de videojuego es identificar sus capacidades gráficas, ya que son las encargadas de mostrar las imagenes2D y 3D en pantalla.

La facilidad de aprender a usar el motor de videojuegos y la facilidad para exponer el juego a diferentes plataformas.

Motor de físicas

hace posible aplicar aproximaciones físicas a los videojuegos para que tengan una sensación mas realista en la interacción de los objetos con el entorno. Simula atributos físicos volumen, peso, aceleración, gravedad, etc.

Motor de sonido

Los sonidos y la banda sonora de un videojuego. E el encargado de cargar pistas, modificar su tasa de bits, quitarlas de reproducción, sincronizarlas.

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.

C++

Es uno de los mas utilizado en el sector por los profesionales. Popular en los títulos AAA, se utiliza en PlayStation y Xbox, y en juegos independientes.

Es el lenguaje mas compatible con la mayoría de los motores de juego y tiene un tiempo de ejecución bastante rápido

C Sharp

Lenguaje de programación popular, sobre todo en entornos Windows.

Es menos flexible y compatible que C++.

No esta limitado a un determinado sistema operativo o plataforma; se puede crear juegos para iOS, Android, Windows Play Station y Xbox.

Java

Es muy utilizado y tiene similitudes con C++.

Es muy versátil, se puede utilizar en todas las plataformas, dispone de gran cantidad de frameworks para el desarrollo 2D.

Módulos de código abierto

JavaScript

Lenguaje mas utilizado en el desarrollo de videojuegos web y de navegador. La mayoría de los motores de videojuegos son compatibles con JavaScript.

Cuenta con múltiples frameworks para 3D y una gran variedad de bibliotecas.

Python

Es un lenguaje muy flexible y potente para esto.

Su ejecución es mucho mas simple que la de otros lenguajes y su framework Pygame permite a los desarrolladores crear prototipos des sus videojuegos de manera rápida y sencilla, y funciona prácticamente en todas las plataformas y sistemas operativos.

Lua

Es un lenguaje de programación sencillo, rápido y fácil de aprender. Compatible con lenguajes mas complejos y de rápida ejecución, también se usa para aplicaciones web y procesamiento de imágenes.

Especial para proyectos independientes y programadores que estén empezando en la programación.

Metodologías de desarrollo de videojuegos.

Metodología SUM tiene como objetivo desarrollar videojuegos de calidad en tiempo y costo, así como la mejora continua del proceso para incrementar su eficacia y eficiencia.

Obtener resultados predecibles, administrar eficientemente los recursos y riesgos dl proyecto, y lograr una alta productividad del equipo de desarrollo

SUM fue concebida para que se adapte a equipos multidisciplinarios pequeños y para proyectos cortos con alto grado de participación del cliente.

Su ciclo de vida consiste en cinco fases secuenciales son: concepto, planificación, elaboración, beta y cierre.

Se detecta, mediante entrevistas a las empresas de desarrollo de videojuegos

más relevantes a nivel nacional, las distintas carencias existentes. Entre ellas se encuentra la falta de formalización de una metodología de desarrollo. A partir

del conocimiento que se obtiene del uso de metodologías agiles en la industria

de videojuegos a nivel mundial.

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

Es lo primero que atrae como jugadores en una consola.

Es la puerta de entrada, la invitación que nos hacen los diseñadores para entrar al mundo que han construido.

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.

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.

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.

Entorno: Lo primero para tener en cuenta es dónde se va a jugar el juego que estás diseñando. Debes tener en cuenta las posibilidades y limitaciones que te ofrece la plataforma. No es lo mismo hacer juegos para smartphones que para una consola o que para un PC.

Contenido: Un buen diseño de UI proporciona al jugador toda la información necesaria para que pueda interactuar con el juego y que todo sea fluido.

Diseño Visual: Los videojuegos, casi siempre, entran por los ojos. Un apartado visual feo o denso en la interfaz del juego puede resultar contraproducente y sacar al jugador de la experiencia inmersiva que quieres proporcionarle. Debes definir el estilo de arte.

Arquitectura de la información: Definir qué elementos son de mayor o menor importancia para el usuario y organizarlos de tal forma que todo resulte en un diseño de interfaz coherente y relevante.

https://www.fing.edu.uy/sites/default/files/biblio/22811/asse_2009_16.pdf

https://www.tokioschool.com/noticias/diseno-interfaz-videojuego/

Desarrollo de prototipos de videojuegos

Concepto, tipos y características de los motores de videojuego

Es un framework o un conjunto de herramientas que ayudan a agilizar el proceso de desarrollo 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, pero que son de suma importancia para la experiencia del usuario final, es decir, con ayuda de los motores, los programadores pueden enfocarse en desarrollar buenos juegos sin perder tiempo en otras tareas.

Entre las herramientas de los motores de videojuegos podemos distinguir algunas de suma importancia como: motor de renderizado, física de videojuegos y detección de colisiones, scripting, motor de sonidos, inteligencia artificial y administradores de memoria.

Unreal Engine: Fue creado por Epic Games. Entre las empresas que lo utilizan se encuentran Electronic Arts y Ubisoft. Utiliza el lenguaje de programación C++.

Unity 3D: Se trata de una de las innovaciones más importantes creadas por la comunidad científica y de videojuegos y permite jugar a complejos videojuegos en 3D sin necesidad de instalarlos en el ordenador. Los videojuegos creados con el motor Unity 3D se pueden jugar en un navegador con el reproductor Unity Web Player, eliminando la necesidad de instalar el videojuego.

Frostbite Engine: Este motor para videojuegos creado por Digital Illusions CE se utiliza para crear videojuegos de acción en primera persona. La nueva versión del motor Frostbite Engine es Frostbite 3.

Decima Engine: Alberga herramientas y características para crear inteligencia artificial, física, lógica y mundos en el desarrollo, así como compatibilidad con 4K y HDR.

Luminous Studio: Es un motor de videojuegos multiplataforma desarrollado y usado internamente por Square Enix. Con este motor se desarrolla el juego Final Fantasy.

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

Los sistemas de desarrollo para videojuegos, la realidad virtual y la realidad aumentada se utilizan para crear aplicaciones sofisticadas con un esfuerzo relativamente pequeño, ya que incluyen una gran cantidad de componentes listos para ser usados junto con una metodología de desarrollo bien probada.

Programa de juego principal

La lógica del videojuego debe ser implementada a través de diversos algoritmos.

Es una estructura distinta de cualquier trabajo de renderizado o de sonido.

Renderización

Proceso en el cual se generan los gráficos 3D por computadora a fin de mostrar en pantalla el aspecto visual del videojuego. Genera gráficos en 3D por varios métodos y se ocupa de mostrar escenarios, modelos, animaciones, texturas, sombras, iluminaciones y materiales.

Se basan en una o varias API, que proporcionan una abstracción del software en la GPU.

Las bibliotecas de bajo nivel proporcionan acceso independiente del hardware. Con la llegada del procesamiento de físicas por aceleración de hardware.

Polígonos

Todo elemento que tenga una composición tridimensional es clasificado por polígonos. Es el procedimiento más primitivo en la creación de un mundo tridimensional y su complejidad geométrica varía de acuerdo con las capacidades técnicas del hardware. Dicha complejidad se puede clasificar en una composición poligonal baja, media y alta. Esto dará como resultado un determinado nivel de detalle y distancia de dibujado.

Audio

El audio de un videojuego se llega a manejar de muchas maneras y esto depende de las capacidades que tenga el motor.

En algunos casos, los motores de videojuegos pueden exigir configuraciones exactas, aunque el método más conocido es la administración de audio mediante el bucle de música, o bien modificar el tono cuando se trata de voces o efectos de sonido.

Los componentes en relación con el audio se encargan de manipular algoritmos que tienen que ver con la carga, modificación y salida del audio a través del sistema de altavoces del usuario.

Motor físico

El motor físico es responsable de emular las leyes de física en forma realista dentro del motor de videojuego. Específicamente, proporciona un conjunto de funciones para simular acciones reales a través de variables como la gravedad, la masa, la fricción, la fuerza, la flexibilidad1 y las colisiones, actuando sobre los diversos objetos dentro del juego al momento de su ejecución.

Inteligencia Artificial

La inteligencia artificial es quien provee de estímulo al videojuego. Su elaboración es crítica a la hora de lograr un sistema de juego pulido y que entretenga. Puede tornarse muy compleja y es necesario tener en cuenta ciertas variables, tales como crear comportamientos programados, delimitar su visión del mundo tridimensional, su interacción en él, la toma de decisiones y con ello lograr una consistencia lógica y coherente en la que el jugador debe responder de una manera esperada.

Scripting

Procedimiento que se utiliza normalmente en situaciones donde es necesario explicar algo de manera controlada. Actualmente se las utiliza a fin de representar la historia que tendrá el videojuego, permitiéndole al desarrollador tomar el control

de la escena y manipularla, tales como colocar objetos o añadir eventos que el jugador no controla.

Transición narrativa y lenguaje visual de videojuegos.

Por que el uso de la narrativa; la manera de contar historias hace que el usuario se enamore, entretenga, enganche y se convence con el juego, además de que estas también venden, educan, posiciona, etc.

Lenguaje audiovisual: codificación que permite narrar.

Encuadres, ángulos, movimientos de cámara, elipsis, transacciones

Toma: unidad ininterrumpida de espacio tiempo.

Escena: acciones que ocurren en un mismo espacio y tiempo.

Secuencia: escenas que componen una gran acción.

Banda sonora

Música: ritmo, volumen, intensidad, generan atmosfera se conectan con la psicología del jugador.

Efectos atmosféricos e incidentales: apoyan las acciones, pueden cambiar el sentido.

Diálogos: personajes, narrador, incidentales

El discurso del entretenimiento se concentra en generar emociones mediante el diseño de una narrativa.

Selección de acontecimientos extraídos las vidas de los personajes que se componen para crear una secuencia estratégica que produzca emociones especificas y expresan una visión concreta del mundo.

Solo presentar acciones que refuercen las emociones que queremos generar o que apoyen la visión del mundo que creamos.

Un videojuego y una historia son una abstracción de la vida.

La narrativa funciona, en beneficio de game desing.

Explicar el proceso de desarrollo de videojuego acorde a los elementos de programación visual

PREPRODUCCION-Diseño y conceptualización

Todo comienza con una idea. Es en base a ella con la que empieza a construir las bases de lo que será el videojuego.

Los roles involucrados en esta primera etapa comúnmente son las cabezas que posteriormente dirigirán al equipo de trabajo durante la producción:

Productor ejecutivo, director creativo, director de juego, director de narrativa, director de animación, director de arte, arquitecto de sistemas.

Son ellos quienes se dan a la tarea de comenzar a definir todos los fundamentos del videojuego y de su producción.

PRODUCCION

Donde se concentra la mayor parte del trabajo de producción y es donde el equipo suele crecer, a veces exponencialmente. La duración de ésta fase depende principalmente del tipo de juego que se produce y puede ir de unos cuantos meses a varios años.

First Playable

También conocido como FP, se refiere a la versión del juego con una primera iteración de assets y funciones de jugabilidad. El FP se enfoca principalmente en construir las bases artísticas y desarrollos técnicos que definen la experiencia esencial del juego.

Alpha

Implementar todas las funcionalidades esenciales de jugabilidad y contar con la mayoría los assets del juego.

El equipo de desarrollo se dedicará a terminar de construir y pulir las funcionalidades implementadas en el código fuente, así como finalizar todos los assets artísticos.

En este punto las pruebas y la retroalimentación son importantes para encontrar problemas en el videojuego.

Beta

Marca el juego finalizado en funcionalidades, gameplay y cuenta con todos los assets. Asimismo, se considera que está libre de bugs mayores que prevengan el lanzamiento del juego.

Esta fase se enfoca en encontrar y aplastar el mayor número de bugs así como optimizar la eficacia del juego antes de su lanzamiento.

POST PRODUCCION

Originalmente el proceso de desarrollo finalizaba con el lanzamiento del videojuego; no había seguimiento ya que no existía el concepto de soporte continuo, parches de actualizaciones, nuevas características, mejoras, etc.

Mantenimiento

Son los procesos continuos enfocados en mantener balanceados los sistemas de juego y optimizada la infraestructura técnica. Las actividades más comunes son: arreglo de bugs, optimizaciones y balance de reglas, soporte y mejoras en infraestructura, mantenimiento de servicios en línea, etc.

Conclusión

Un desarrollo exitoso no sólo demanda de expertos técnicos y artísticos sino también de una idea original combinados con claros procesos de producción.

https://es.slideshare.net/josuerodrigo/la-narrativa-interactiva-del-videojuego

https://www.industriaanimacion.com/2019/09/las-etapas-para-desarrollar-un-videojuego/