INTRODUCCIÓN

A partir de este momento se empezará a introducir en una nueva forma de pensar la programación. Esta nueva forma todo lo que hemos visto hasta ahora (concepto de algoritmos, variables, estructuras de control, fases de análisis y diseño para resolver problemas) y las estructura para crear aplicaciones (en nuestro caso videojuegos) de la forma estándar que la industria actualmente utiliza. Primero se expondrán los conceptos más importantes para que puedas dominar perfectamente el lenguaje técnico que ciertamente requiere esta filosofía de programación.

CONCEPTOS

PARADIGMAS

El estadounidense Thomas Kuhn, los define como:

> *La serie de **prácticas** que trazan los lineamientos de una disciplina científica a lo largo de un cierto **lapso temporal**. El éxito de un paradigma es consecuencia de su efectividad para resolver algún problema.*

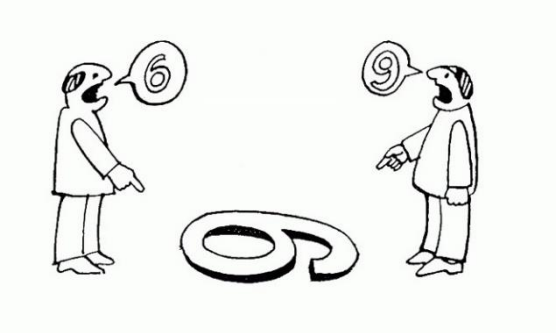
Las prácticas hacen referencia a:

- **Las leyes establecidas y los supuestos teóricos**. Por ejemplo, las leyes de movimiento de Newton forman parte del paradigma newtoniano y las ecuaciones de Maxwell forman parte del paradigma que constituye la teoría electromagnética clásica.
- **El instrumental y las técnicas instrumentales** necesarias para hacer que las leyes del paradigma se refieran al mundo real. La aplicación en astronomía del paradigma newtoniano requiere el uso de diversos telescopios, junto con técnicas para su utilización y diversas técnicas para corregir los datos recopilados.
- **Los principios generales** propios del paradigma que guían el trabajo dentro del paradigma. Por ejemplo, para construir una un puente, se deben seguir ciertos pasos, medidas y métodos preestablecidos para garantizar que el trabajo se realice de manera correcta.

Respecto del ***lapso temporal***, se hace referencia a que un paradigma es el resultado de un proceso social en el cual un grupo de personas desarrollan nuevas ideas, creando nuevos principios y prácticas alrededor de esas ideas. Entonces es totalmente válido que un paradigma pueda aplicarse perfectamente para explicar un aspecto del mundo (en base a las prácticas que define) mientras que para otros fenómenos no se pueda aplicar o sea obsoleto (en cuyo caso es probable que exista otro paradigma diferente que se amolde mejor).

Por este motivo, se puede entender por qué cada vez que surgen nuevos paradigmas: como consecuencia de un proceso natural de evolución o maduración del propio concepto.
Para comprenderlo mejor podemos referirnos a un ejemplo, en la programación de videojuegos: Si quieres programar un videojuego debes realizarlo siguiendo un paradigma. Para ese tipo de problema ¿Hay un solo paradigma aplicable? Probablemente no. ¿Puede ser que un paradigma de programación no se pueda aplicar a un tipo de problema? Probablemente sí. ¿Puede ser que haya un paradigma más adecuado que otros para un tipo de problema? Seguramente que sí.

Observe la imagen de la derecha. Ante un problema, diferentes paradigmas te proponen una visión diferente de cómo interpretarlo. Y no por ello la percepción de alguno de estos paradigmas está equivocado. Pero lo que sí es probable es que alguno de esos paradigmas sea más efectivo que el resto en situaciones específicas.



En las ciencias sociales, un **paradigma se utiliza para explicar la forma en que se entiende el mundo**. Se emplea para mencionar a todas aquellas experiencias, creencias, vivencias y valores que repercuten y condicionan el modo en que una persona ve la realidad y actúa en función de ello.

De esta manera para brindar una definición formal, se considera paradigma a:

- Los marcos de referencia que imponen reglas sobre cómo se deben hacer las cosas, indican qué es válido dentro del paradigma y qué está fuera de sus límites. Un paradigma distinto implica nuevas reglas, elementos, límites y maneras de pensar, o sea implica un cambio.
- Patrones de pensamiento para la resolución de problemas. Un modelo o esquema fundamental que organiza nuestras opiniones con respecto a algún tema en particular. Los paradigmas establecen límites adoptados por los miembros de una comunidad científica para resolver problemas sustentados por los principios, leyes, supuestos teóricos y técnicas que la conforman.

ELEMENTOS DE LOS PARADIGMAS

Sin entrar en detalles, los paradigmas se sustentan en teorías, creencias, valores, leyes, técnicas e hipótesis. Debido a esta formalidad, este último esquema intenta diferenciar conceptos que normalmente son confundidos con paradigmas.

PARADIGMAS DE PROGRAMACIÓN

Existen diferentes concepciones para los paradigmas de programación:

- Un proceso de diseño que va más allá de una gramática, reglas semánticas y algoritmos. Es un conjunto de métodos sistemáticos aplicables en todos los niveles del diseño de programas. Representan un enfoque particular o filosofía para la construcción del software.
- Son propuestas tecnológicas adoptadas por la comunidad de desarrolladores que se enfocan a resolver uno o varios problemas definidos y delimitados.
- Es un modelo básico de diseño y desarrollo de programas, que permite producir programas con unas directrices específicas, tales como: estructura modular, fuerte cohesión, alta rentabilidad, etc.
- Es una colección de modelos conceptuales que en conjunto modelan el proceso de diseño y determinan la estructura de un programa. Esa estructura conceptual de modelos está pensada de forma que los modelos determinan la forma correcta de los programas y controlan el modo en que el desarrollador piensa y formula soluciones, que luego son implementadas en un lenguaje de programación.
- Provee y determina la visión y métodos de un programador en la construcción de un programa o subprograma. Diferentes paradigmas resultan en diferentes estilos de programación y en diferentes formas de pensar la solución de problemas (con la solución de múltiples “problemas” se construye una aplicación o producto de software).

Existen tres cuestiones para tener en cuenta respecto de los paradigmas de programación:

1. Ningún paradigma es mejor que otro, sino que cada uno tiene ventajas y desventajas. También hay situaciones donde un paradigma resulta más apropiado que otro.

2. Para que las características esenciales del paradigma sean efectivamente aplicadas, las características del lenguaje de programación utilizado para implementar la aplicación deben reflejar adecuadamente los modelos conceptuales de ese paradigma. Cuando un lenguaje refleja bien un paradigma particular, se dice que soporta el paradigma, y en la práctica un lenguaje que soporta correctamente un paradigma, es difícil distinguirlo del propio paradigma.

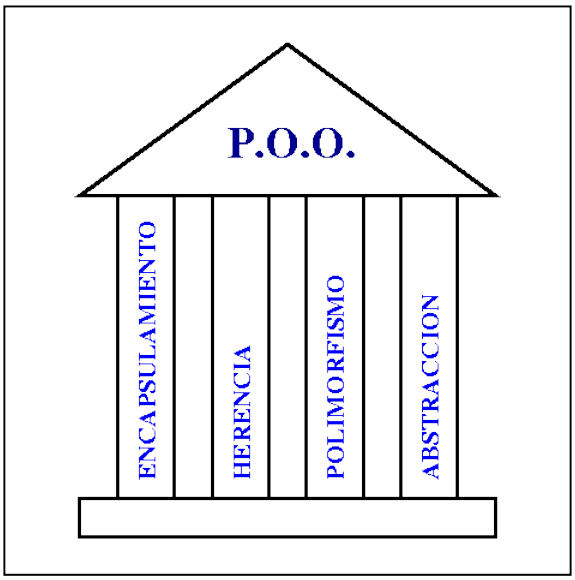
3. Los lenguajes de programación están basados en uno o más paradigmas, por ejemplo: Processing está basado en el paradigma orientado a objetos. El lenguaje de programación Scheme, en cambio, soporta solo programación funcional. Otros lenguajes, como C++ y Python soportan múltiples paradigmas.

EL PARADIGMA ORIENTADO A OBJETOS DE MANERA RESUMIDA

Este paradigma se basa en la idea de que **es posible representar un problema y su solución de una forma similar a la que el cerebro humano procesa la información que le rodea**. En términos generales el **paradigma representa la información del problema y su solución en la forma de entidades denominadas objetos**. Estos objetos **poseen un conjunto de** características cuyos valores son almacenadas en variables especiales denominadas **atributos**. Los atributos generalmente son **utilizados o modificados por** un conjunto de **operaciones u acciones que**

contienen un algoritmo específicamente diseñado para realizar lo que el nombre de la operación indique.

Uno de los objetivos de que un objeto se componga de atributos y operaciones es que ese objeto mantenga un conjunto de características y acciones específicos, **así si algún otro objeto requiere de un dato o de una operación se lo solicita al objeto adecuado**.



Esto es, cuando un objeto necesita trabajar sobre una característica de otro objeto, o cuando requiere que otro objeto realice una operación debe enviarle un mensaje. **Los mensajes son entonces, el mecanismo que utiliza el paradigma para establecer relaciones entre los objetos**. Esta solicitud tiene un conjunto de reglas y normas a seguir al momento de utilizarlas en un lenguaje de programación: **el protocolo de mensajes**. Finalmente, como todo paradigma, su filosofía se estructura sobre reglas, supuestos, técnicas y teorías que están organizadas en lo que se denominan Pilares o **Propiedades del Paradigma Orientado a Objetos: la abstracción, la encapsulación, la herencia y el polimorfismo**.

Estas propiedades **estructuran por un lado la creación, el uso y la destrucción de los objetos alrededor del concepto de clase**; mientras que **por otro lado establece que el protocolo de mensajes es un contrato entre las clases** que indica cómo una clase construye sus operaciones y cómo las debe poner a disposición de otras clases. De esta manera se introducen los conceptos de **interfaz e implementación**.

Más aún, estas propiedades definen formalmente que es un objeto, como se modelan los mismos y los mecanismos **que permiten optimizar la reusabilidad, la facilidad de mantenimiento y aplicar poderosas técnicas de programación**.

EL MODELADO Y LA ABSTRACCIÓN EN EL PARADIGMA ORIENTADO A OBJETOS

El **modelado** es una técnica que consiste en representar información mediante un modelo. Un **modelo** es una abstracción de una realidad. El término **abstracción** hace referencia al proceso de centrarse únicamente en los detalles relevantes del fenómeno estudiado o que es objeto de observación.

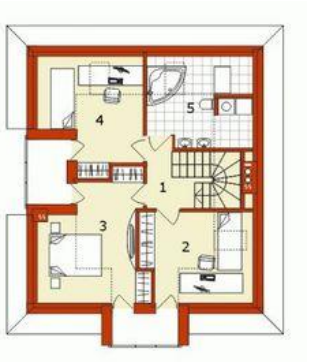
Concepto de modelo

- Un modelo es una abstracción de un sistema
- La abstracción permite ocuparnos de detalles relevantes para un propósito
- Utilidad del modelado: abordar sistemas complejos
- Técnica muy empleada (ej. edificación)

La abstracción es también, una de las propiedades del paradigma orientado a objetos; esto significa que el paradigma orientado a objetos buscará a través de esta propiedad:

- Abstraer un problema (determinar los **aspectos relevantes** del problema resolver) para luego realizar una,
- abstracción de la solución de ese problema (determinar **los aspectos relevantes** de la solución que se propone como solución al problema) y finalmente
- construir una abstracción del sistema informático creado. Esto significa que los **aspectos relevantes** del producto sean programados, probados y documentados en la forma de clases y sus interrelaciones, para que otros desarrolladores puedan interpretarlos.

Los modelos son herramientas para representar una abstracción y debido a su efectividad son utilizados en muchas disciplinas. El ejemplo típico está en la ingeniería civil o la arquitectura. El arquitecto utiliza diversos planos que son abstracciones de una parte del edificio construido o a construir.



Debido a que los profesionales de la construcción se han formado en la traducción de estos planos, los interpretarán exactamente igual (al menos que incurran en un error humano). Esto significa que **una de las características más importantes de los modelos y que define cuan efectivos son es su capacidad de interpretación unívoca**, es decir no dan a lugar a ambigüedades o doble interpretación.

El paradigma orientado a objetos garantiza realizar un proceso de abstracción para resolver problemas y construir software; pero **no ofrece un modelo concreto** por el cual representar dicha abstracción.

Entre los diferentes modelos que se pueden utilizar para representar los elementos del paradigma orientado a objetos se encuentra UML (de Unified Modeling Language, o Lenguaje de Modelado Unificado) que ofrece un conjunto de 13 diagramas para representar diferentes vistas de un producto software. De estos diagramas en esta materia se usarán los diagramas de clases, los diagramas de secuencia y la máquina de estados.

MITOS SOBRE EL MODELADO

Es común una cierta resistencia hacia el modelado. El factor principal que aducen los diseñadores de videojuegos consiste en señalar que el tiempo que se debe dedicar al modelado es importante, por lo que prefieren programar directamente. Esto constituye un error garrafal que puede traer consecuencias graves para la finalización del proyecto del videojuego, ya que, sin importar la experiencia de los diseñadores, la comunicación entre diseñadores sobre un producto debe garantizar que no haya doble interpretaciones.

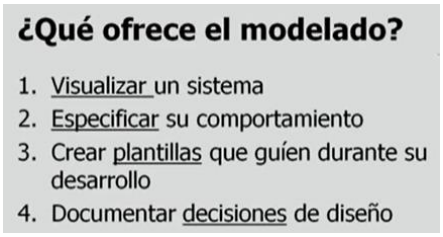
Así como el Game Concept y el GDD (Game Design Document) son modelos utilizados para determinar la idea del videojuego que se quiere construir, a nivel de la programación de ese videojuego se deben usar modelos para que los diseñadores de juegos tengan un plano de lo que se debe construir.

En realidad, la experiencia indica que los desarrolladores que esquivan el uso de modelos son aquellos “principiantes” que les cuesta interpretarlos o construirlos. Sin embargo, a medida que se adquiere experiencia, los diseñadores resaltan sus ventajas, especialmente a la hora de

minimizar errores de programación, minimizar el tiempo de desarrollo y facilitar la integración de componentes.

CARACTERISTICAS DEL MODELADO

El modelado ofrece 4 características esenciales:



1) Una visualización de un sistema: tanto del problema a solucionar como de la solución. En ambos casos estamos refiriéndonos a una aproximación (en el primer caso una aproximación de los objetos de nuestro juego y como ellos interactúan, mientras que del segundo caso un esquema y documentación del software del videojuego)
2) Una especificación de su comportamiento: nos permite indicar detalladamente como actúa el mismo ante diferentes situaciones.
3) Una plantilla que guíe a los desarrolladores durante su construcción: es decir son los "planos" que el arquitecto del software sigue para construir el producto.
4) Documentar decisiones de diseño: siempre las decisiones de diseño deben estar bien documentadas y justificadas. En algunas metodologías la documentación se dejaba al último; o como sucede de forma muy habitual está la tentación de no documentar por el tiempo que incurre. Los modelos permiten minimizar el tiempo de documentación.

LENGUAJES DE MODELADO

Un lenguaje de modelado es una colección de técnicas de modelado. Una **técnica de modelado** posibilita escribir un modelo ya que consta de una colección de símbolos y un conjunto de reglas que definen cómo los símbolos pueden ser combinados de forma válida para formar un modelo:

Técnica de Modelado = colección de símbolos + reglas de composición

Dentro del ámbito del desarrollo de software es posible utilizar diferentes lenguajes de modelado, algunas opciones pueden ser:

- ✓ Código fuente: resulta una elección desaconsejable, puesto que se torna complejo de manejar. Asume que el desarrollador debe conocer el lenguaje de programación utilizado. Por otro lado, es difícil procesar y apreciar el comportamiento debido a la gran cantidad de líneas de código que presumiblemente posea el programa ¿Estoy refiriéndome a que no deba documentar el código fuente? Absolutamente no. Me estoy refiriendo a que no es aconsejable usar el código fuente como modelo de nuestra aplicación. El código siempre debe estar completamente documentado, pero eso no significa que el mismo deba usarse como modelo del sistema. Una opción más conveniente es usar otro lenguaje de modelado y de esa manera poder verificar que el código fuente es la implementación de ese modelo (que el modelo se corresponda con el código fuente)
- ✓ Lenguaje Natural: Se refiere a crear diversos documentos en el cual se describe el comportamiento modelado. La característica principal de este tipo de lenguaje es que es propenso a errores, puesto que se generan documentos muy extensos; como

consecuencia será difícil asegurar la consistencia y que no haya ambigüedad en nuestro modelado.

- ✓ Lenguaje Visual: es el que mayor éxito ha tenido y el que se emplea (o se debería emplear) en la práctica por las empresas al momento de modelar software. Mediante anotaciones gráficas se construyen los modelos. Estas anotaciones tienen un significado concreto y unívoco que no da lugar a ambigüedad, y es una representación atractiva y rápida para observar el comportamiento del software debido que son fáciles de interpretar y procesar.

Con lo expuesto hasta ahora, resulta evidente que de tener que elegir un lenguaje de modelado a utilizar, la opción sería decantarse por los lenguajes de modelado visuales.

Teniendo esta primera decisión de diseño adoptada, la siguiente será determinar cuáles técnicas de modelado utilizar. Estas van desde aquellas que son puramente textuales hasta las puramente formales.

- Las técnicas más textuales tienen como principales ventajas que son muy fáciles de entender. Sin embargo, una desventaja es que tienden a ser imprecisos, ambiguos y no ofrecen verificación inteligente y control de calidad.
- Las técnicas puramente formales como, por ejemplo, el álgebra de procesos, ofrecen muchas posibilidades de verificaciones formales, pero como consecuencia son más difíciles de entender y utilizar.

Como se ha mencionado anteriormente, en la asignatura usaremos el Lenguaje de Modelado Unificado (UML) porque ofrece lo que se denominan diferentes diagramas (modelos visuales). Sus diagramas se ubican en forma intermedia a las textuales y las formales. Así, por ejemplo, los diagramas de clases se ubican aproximadamente en el medio de estas dos clasificaciones. Tienen la ventaja de ser razonablemente fáciles de entender, mientras que, al mismo tiempo, es lo suficientemente formal para lograr un buen nivel de precisión y posibilidades de verificación.

LOS ELEMENTOS CENTRALES DE LA ABSTRACCIÓN DE OBJETOS

Como se ha mencionado anteriormente, el paradigma orientado a objetos se sustenta en 4 pilares. El pilar denominado abstracción es la base para todo el desarrollo orientado a objetos. Consiste en la formalización de los conceptos necesarios que permiten generar soluciones orientadas a objetos. La abstracción consiste en aislar elementos de su contexto o del resto de los elementos que lo acompañan para facilitar su estudio capturando las características esenciales del mismo.

En el ámbito de la programación orientada a objetos la abstracción permite representar un problema y/o su solución en términos de un modelo conformado por clases, objetos y la interrelación entre ellos.

El modelo orientado a objetos permite construir una representación para analizar, describir, explicar simular o predecir un fenómeno y se sustenta alrededor de las siguientes definiciones:

- **Clase**: modelo, molde, plano o maqueta a partir del cual se pueden generar objetos. Las clases son entidades utilizadas para analizar el problema y diseñar la solución. Toda clase posee 3 elementos importantes: el nombre de la clase, los atributos y las operaciones.

Mediante las clases se pueden determinar los actores (clases) que participan en el problema estudiado (o en la solución) así como las características y acciones que estas entidades poseen y que contribuyen de alguna manera para que se cumplan los objetivos abordados por la solución planteada (los requisitos). Las clases son programadas en un lenguaje de programación orientado a objetos.

- **Objeto**: Los programas se ejecutan en memoria, adoptando el nombre de procesos. En el caso de los programas orientados a objetos estos procesos se denominan objetos que adquieren los atributos y operaciones de la clase a partir de la cual se ha creado, de esta manera en memoria pueden existir muchos objetos generados a partir de la misma clase pero cada una con su propia identidad que la separa de las demás (esto puede asemejarse al hecho de que a partir de un mismo plano se pueden construir varias viviendas, todas iguales por las especificaciones del plano pero en definitiva cada una es una construcción individual)
- **Relaciones**: Indica la forma en que los objetos colaboran entre ellos para realizar alguna tarea específica. Esto genera un nuevo concepto que es el **mensaje**: es el mecanismo por el cual un objeto en memoria solicita a otro que ejecute una operación. Entonces las relaciones indican la forma en que los objetos envían mensajes a otros objetos.
- **Interfaz**: Las operaciones de un objeto que pueden ser solicitadas por otro objeto se denominan **servicios**. Para que los servicios de un objeto puedan ser invocados por otro objeto; el objeto que desea ponerlos a disposición de los otros objetos debe definir una interfaz. La interfaz es simplemente un mecanismo por el cual se determina si una operación se halla disponible para ser invocada por otro objeto (las operaciones dentro del objeto siempre son servicios para las otras operaciones del mismo objeto, esto es, las operaciones de un objeto siempre pueden ser invocadas por las otras operaciones del mismo objeto). El concepto de interfaz también se aplica a los atributos, esto es; si no se especifica una interfaz para un atributo, el mismo no podrá estar disponible para otros objetos.

Desde un aspecto conceptual, hemos descripto en que consiste la propiedad abstracción del paradigma orientado a objetos. Resulta natural para los desarrolladores que recién se inician en este paradigma que haya elementos que no se comprendan. Por este motivo en las próximas entregas se profundizará en detalle mediante ejemplos propios del mundo de los videojuegos.