VARIABLES, TIPOS DE DATOS Y OPERADORES, Y ESTRUCTURAS DE CONTROL DE FLUJO

Variables, tipos de datos y operadores, y estructuras de control de flujo

Introducción
La programación es un lenguaje que se ha convertido en parte integral de la vida moderna. Desde la creación de software hasta el diseño de páginas web, la programación ha crecido enormemente en los últimos años. En este artículo, vamos a discutir dos conceptos clave en la programación de videojuegos: **variables, tipos de datos y operadores, y estructuras de control de flujo**. Estos conceptos son fundamentales para cualquier programador de videojuegos, ya que permiten crear código robusto y efectivo.

Explicación
Variables, tipos de datos y operadores:

Variables:
Las variables son contenedores que almacenan valores que pueden ser en el código. En el contexto de los videojuegos, una variable puede ser utilizada para almacenar la puntuación del jugador, la posición del personaje, el estado del juego, etc. Una variable puede ser declarada usando una palabra clave específica, seguida de un nombre descriptivo y un signo igual para asignar un valor. Por ejemplo:

```
int score = 0;
```

Tipos de datos:
Los tipos de datos son los diferentes tipos de valores que se pueden almacenar en una variable. Según Baron (2021), en el contexto de los videojuegos, algunos tipos de datos comunes son los números enteros, los flotantes, los booleanos y las cadenas de texto. Los números enteros son utilizados para valores enteros, como la cantidad de vidas que tiene un jugador. Los flotantes son utilizados para valores decimales, como la posición del personaje en el juego. Los booleanos son utilizados para valores verdaderos o falsos, como si un objeto está en colisión con otro objeto. Las cadenas de texto son utilizadas para almacenar texto, como el nombre del jugador. Algunos ejemplos de tipos de datos son:

```
int age = 25;
float positionX = 5.0f;
bool isGameOver = false;
string playerName = "John";
```

Operadores:
Los operadores son símbolos que se utilizan para realizar operaciones matemáticas o lógicas en el código. En el contexto de los videojuegos, algunos operadores comunes son el operador de asignación (=), el operador aritmético (+, -, *, /), el operador de comparación (==, !=, >, <), y el operador lógico (&&, ||). Los operadores pueden ser utilizados para realizar cálculos matemáticos, como la velocidad del personaje, o para verificar si dos valores son iguales, como la posición actual del jugador y la posición del objetivo.

A continuación, se presentan algunos ejemplos comunes de operadores en la programación de videojuegos:

- **Operadores aritméticos (+, -, *, /)**: se utilizan para realizar operaciones matemáticas en los valores de las variables. Por ejemplo, si queremos mover un personaje a una nueva posición, podríamos utilizar el operador aritmético de suma (+) para añadir la distancia a la posición actual del personaje. Así, si la posición actual es (2, 3) y queremos mover al personaje 2 unidades hacia la derecha y 1 unidad hacia arriba, podríamos usar el siguiente código:

```
int posX = 2;
int posY = 3;
int moveX = 2;
int moveY = 1;

posX = posX + moveX;
posY = posY - moveY;
```

El resultado final sería que el personaje se mueve a la posición (4, 2).

- **Operadores de comparación (==, !=, >, <)**: se utilizan para comparar dos valores y obtener un resultado lógico (verdadero o falso). Por ejemplo, si queremos comprobar si un personaje ha llegado a un objetivo, podríamos comparar su posición actual con la posición del objetivo. Si ambas posiciones son iguales, entonces el personaje ha llegado a su destino. Podríamos utilizar el siguiente código:

```
int goalX = 10;
int goalY = 5;
int posX = 8;
int posY = 3;
if (posX == goalX && posY == goalY) {
  // El personaje ha llegado al
objetivo
} else {
  // El personaje aún no ha llegado al
objetivo
}
```

En este caso, estamos utilizando el operador lógico && (AND) para comprobar si ambas condiciones se cumplen al mismo tiempo.

- **Operadores lógicos (&&, ||):** se utilizan para combinar varias condiciones lógicas. Por ejemplo, si queremos comprobar si un personaje está en colisión con un objeto, podríamos utilizar una combinación de operadores lógicos y de comparación. Podríamos utilizar el siguiente código:

```
bool isColliding = false;
int charX = 5;
int charY = 5;
int objX = 8;
int objY = 4;
int objWidth = 2;
int objHeight = 2;
if (charX + 1 >= objX && charX <= objX
+ objWidth && charY + 1 >= objY && charY
<= objY + objHeight) {
  isColliding = true;
}

if (isColliding) {
  // El personaje está en colisión con
el objeto
} else {
  // El personaje no está en colisión
con el objeto
}
```

En este caso, estamos utilizando una combinación de operadores de comparación (>=, <=) y lógicos (&&) para comprobar si el personaje está dentro de los límites del objeto y, por lo tanto, en colisión con él.

Estos son solo algunos ejemplos de cómo se utilizan los operadores en la programación de videojuegos. Según Colonna (2022), dependiendo del tipo de juego y de las mecánicas que quieras implementar, es probable que utilices otros operadores en combinación con otras herramientas de programación para obtener el resultado que buscas.

Estructuras de control de flujo:
En el siguiente nivel de complejidad se encuentran las estructuras de control de flujo. Estas son herramientas que permiten modificar el flujo de ejecución del programa. En otras palabras, nos permiten tomar decisiones y ejecutar diferentes bloques de código dependiendo de ciertas condiciones. Las estructuras de control de flujo más comunes son las siguientes:

- **if/else:** se utiliza para ejecutar un bloque de código si se cumple una condición y otro bloque de código si no se cumple la condición.

En los videojuegos, esto se puede utilizar para comprobar si el jugador ha realizado una acción específica y, en función de eso,

ejecutar diferentes acciones. Por ejemplo:

```
if (playerHasCollectedKey) {
unlockDoor();
} else {
displayMessage("La puerta está cerrada.
Necesitas una llave para abrirla.");
}
```

En este ejemplo, se comprueba si el jugador ha recogido la llave. Si es así, se ejecuta la función "unlockDoor" para desbloquear la puerta. Si no es así, se muestra un mensaje que indica al jugador que necesita una llave para abrir la puerta.

- **switch/case:** similar al if/else, pero se utiliza cuando se tienen varias opciones a evaluar.

En los videojuegos, esto se puede utilizar para comprobar el estado del juego o el estado de un objeto y, en función de eso, ejecutar diferentes acciones. Por ejemplo:

```
switch (gameState) {
case GameState.Start:
displayMessage("Bienvenido al juego.");
break;
case GameState.Playing:
updatePlayerPosition();
updateEnemies();
break;
case GameState.GameOver:
displayMessage("Game Over.");
break;
}
```

En este ejemplo, se utiliza un switch/case para comprobar el estado del juego. Si el estado es "Start", se muestra un mensaje de bienvenida. Si el estado es "Playing", se actualiza la posición del jugador y de los enemigos. Si el estado es "GameOver", se muestra un mensaje de "Game Over".

- **while/do-while:** se utilizan para ejecutar un bloque de código repetidamente mientras se cumple una condición.

En los videojuegos, esto se puede utilizar para ejecutar una acción hasta que se cumpla una condición específica. Por ejemplo:

```
while (playerIsAlive) {
updatePlayerPosition();
checkForCollisions();
}
```

En este ejemplo, se utiliza un while para actualizar continuamente la posición del jugador y comprobar si ha chocado con algún objeto hasta que el jugador muera.

- **for:** similar al while/do-while, esta estructura se utiliza para realizar iteraciones y ejecutar un bloque de código un número determinado de veces.

En los videojuegos, esto se puede utilizar para realizar una acción varias veces, como actualizar la posición de varios objetos en una pantalla. Por ejemplo:

```
for (int i = 0; i < numberOfEnemies;
i++) {
updateEnemyPosition(i);
}
```

En este ejemplo, se utiliza un for para actualizar la posición de todos los enemigos en la pantalla. El número de iteraciones se determina por la variable "numberOfEnemies".

Al igual que las variables y los operadores, las estructuras de control de flujo son herramientas fundamentales de la programación. Sin ellas, no podríamos crear programas capaces de tomar decisiones y adaptarse a diferentes situaciones.

Cierre

En resumen, **las variables, los tipos de datos, los operadores y las estructuras de control de flujo son los fundamentos de la programación.** Conocerlos es esencial para poder programar cualquier tipo de aplicación, desde un sencillo programa de consola hasta un complejo videojuego, pues son la base sobre la cual se construye cualquier programa.

Referencias bibliográficas

- Baron, D. (2021). *Game Development Patterns with Unity 2021 - Second Edition: Explore practical game development using software design patterns and best practices in Unity and C# (2ª ed.)*. Reino Unido: Packt Publishing.

- Colonna, A. (2022). *Code Gamers Development: Essentials: A 9-Week Beginner's Guide to Start Your Game-Development Career*. Independently Published.