**Fase de Análisis: Diseño de un Juego Frogger en Python**

**1. Equipo a Utilizar**

* **Desarrolladores:**
  + 1-2 programadores con experiencia en Python y Pygame
  + Conocimientos en programación orientada a objetos
  + Familiaridad con diseño de interfaces gráficas
* **Diseñador gráfico:**
  + Creación de sprites y assets para el juego
  + Experiencia en diseño de videojuegos
* **Testers:**
  + 1-2 personas para pruebas de jugabilidad y detección de errores

**2. Personal Informático**

* **Hardware:**
  + Computadoras con sistema operativo compatible con Python (Windows, Mac, Linux)
  + Procesador Intel Core i5 o superior
  + 4 GB de RAM o más
  + Tarjeta gráfica integrada o dedicada
* **Software:**
  + Python 3.x
  + Librería Pygame
  + Editor de código (Visual Studio Code)
  + Software de diseño gráfico (Adobe Photoshop, GIMP)

**3. Estudios de los Datos de Entrada**

* **Teclado:**
  + Flechas de dirección para controlar al jugador (rana)
  + Tecla Enter o Espacio para iniciar el juego
  + Tecla Escape para pausar o salir del juego
* **Mouse:**
  + Clic para seleccionar opciones en el menú principal

**4. Estudios de los Datos de Salida**

* **Gráficos:**
  + Representación visual del escenario del juego (carretera, río, meta)
  + Sprites para la rana, autos, troncos y otros elementos
  + Indicador de nivel y vidas restantes
* **Sonidos:**
  + Efectos de sonido para saltos, croar de la rana, choques y éxito
  + Música de fondo

**5. Relaciones entre la Salida y la Entrada**

* Las acciones del jugador (teclas o mouse) controlan los movimientos de la rana en el juego.
* Los eventos del juego (colisiones, llegar a la meta) generan cambios en los gráficos y sonidos.

**6. Descomposición del Problema en Módulos**

* **Módulo principal:**
  + Inicialización del juego y carga de recursos
  + Bucle principal del juego
  + Manejo de eventos (teclado, mouse)
  + Dibujo de gráficos y actualización de la pantalla
* **Módulo de jugador:**
  + Control de movimiento de la rana
  + Detección de colisiones
  + Gestión de vidas
* **Módulo de escenario:**
  + Creación y representación del mapa del juego (carretera, río, meta)
  + Generación y movimiento de autos y otros elementos
* **Módulo de gráficos:**
  + Carga y gestión de sprites y assets
  + Dibujo de elementos en la pantalla
* **Módulo de sonido:**
  + Carga y reproducción de efectos de sonido y música

**Nota:** Esta descomposición es una propuesta inicial y puede modificarse durante el desarrollo del juego.

Esta fase de análisis proporciona una base sólida para el diseño y desarrollo del juego Frogger en Python. Define los recursos necesarios, las características del juego y la estructura modular del código. A partir de aquí, se puede proceder a la fase de diseño detallado y la implementación del juego.

**Fase de Programación: Diseño del Algoritmo para el Juego Frogger en Python**

**1. Descripción General**

En esta fase se establece el algoritmo principal del juego Frogger, definiendo el flujo de acciones y la lógica que lo gobierna. Se utilizan técnicas de programación estructurada y diseño descendiente para organizar el código de manera clara y eficiente.

**2. Algoritmo Principal ( Estructua)**

Python

def main():

# Inicialización del juego

inicializar\_juego()

# Bucle principal del juego

while True:

# Manejo de eventos

procesar\_eventos()

# Actualización del juego

actualizar\_juego()

# Dibujo de gráficos

dibujar\_juego()

# Control de fin de juego

if juego\_terminado():

break

# Finalización del juego

finalizar\_juego()

# Funciones auxiliares para cada tarea específica.

**3. Explicación del Algoritmo**

1. **Inicialización del juego:** Carga recursos, configuraciones y crea objetos del juego.
2. **Bucle principal:**
   * **Manejo de eventos:** Lee entradas del jugador (teclado, mouse) y las procesa.
   * **Actualización del juego:** Mueve la rana, verifica colisiones, actualiza la lógica del juego.
   * **Dibujo de gráficos:** Actualiza la pantalla con los elementos del juego.
   * **Control de fin de juego:** Evalúa si se cumplen las condiciones para terminar el juego.
3. **Finalización del juego:** Libera recursos y muestra la pantalla final.

**4. Programación Estructurada y Diseño Descendiente**

* El algoritmo se divide en funciones modulares con tareas específicas.
* Cada función realiza una acción bien definida y se conecta con otras funciones para lograr el flujo general del juego.
* Se utilizan estructuras de control (if-else, while) para organizar la lógica y la toma de decisiones.

**5. Resultado**

El algoritmo principal y las funciones auxiliares conforman la base para la implementación del código en Python. La programación estructurada y el diseño descendiente facilitan la comprensión, el mantenimiento y la extensibilidad del programa.