Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**PGE: Parcial I**

* **Asignatura:** Programación Genérica y Eventos
* **Alumnos:** Gonzalo Rementeria y Manuel Nicolais
* **Tema:** Parcial I – Programación Genérica y Eventos
* **Fecha de entrega:** 13/09/24.
* **Software:** Godot Engine

**Instrucciones:**

1. **Identificación de Problemas de Usabilidad:**
   * Seleccionar una aplicación de software con la que interactúen regularmente (puede ser un software de código abierto o uno de uso académico).
   * Identificar las características específicas que hacen que el software sea difícil de usar o poco amigable para el usuario.
   * Documentar estas características y justificar por qué representan un problema desde la perspectiva del usuario.
2. **Propuesta de Mejora:**
   * Basándose en los problemas identificados, cada equipo desarrollará un prototipo funcional en C++ que incluya mejoras a través de:
     + **Callbacks:** Implementación de funciones de retorno para mejorar la interacción dinámica en la interfaz.
     + **Bucles:** Uso de bucles eficientes para gestionar las interacciones continuas y el flujo de la interfaz.
     + **Eventos Paint:** Modificación de la apariencia de la interfaz en tiempo real según las interacciones del usuario.
     + **Funciones de Conversión:** Integración de funciones que faciliten la adaptación de formatos o tipos de datos según las necesidades del usuario.
3. **Desarrollo del Prototipo:**
   * Crear un prototipo en C++ que implemente las soluciones propuestas. Este prototipo no tiene que ser un programa completo, pero sí debe demostrar claramente cómo las mejoras propuestas resuelven los problemas de usabilidad identificados.
4. **Documentación:**
   * Cada equipo deberá entregar un informe que incluya:
     + Una descripción del software analizado y los problemas de usabilidad encontrados.
     + Una explicación detallada de las soluciones propuestas y cómo se implementaron en el prototipo.
     + El código fuente del prototipo en C++ con comentarios que expliquen cada sección relevante.
     + Capturas de pantalla o videos que muestren el prototipo en funcionamiento, resaltando las mejoras implementadas.

**Introducción:**

En el presente informe, en el contexto de la materia, Programación Genética y Eventos se detallarán diversas funcionalidades o características de la interfaz gráfica del programa de software libre seleccionado, en este caso **Godot Engine**, motor utilizado para el desarrollo de videojuegos 2D y 3D de código abierto.

Icono

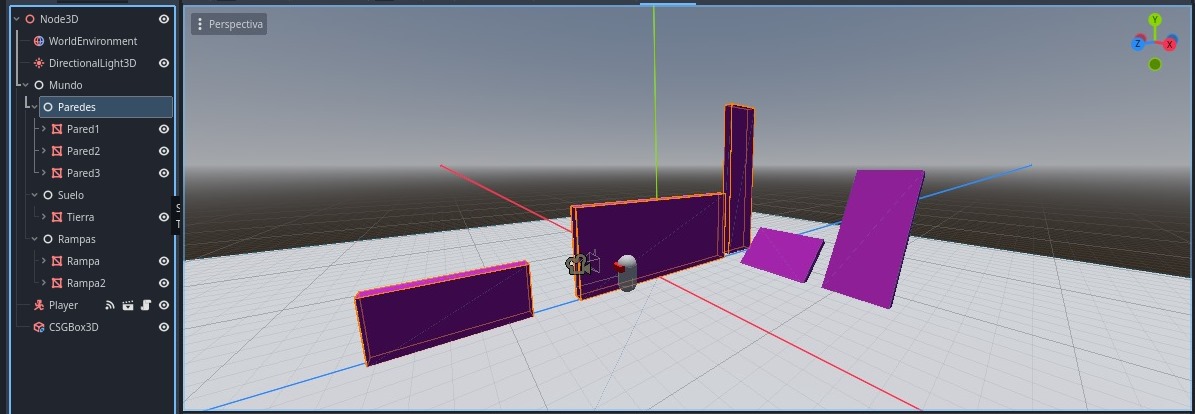
Descripción generada automáticamente

**Desarrollo:**

En esta sección del informe se detallarán las que se consideran características poco amigables o que representen algún tipo de problema para el usuario con respecto a la interfaz de usuario o atributos del motor. Se debe tener en cuenta que estas características están siendo analizadas utilizando la última versión de Godot hasta la fecha en Windows, la cual es **Godot 4.2**. A continuación se detallara el aspecto a mejorar para la comodidad del usuario de este motor.

1. **Contornos en los objetos que pertenecen a un nodo especifico:**

El motor Godot funciona puramente utilizando un sistema de nodos, con hijos y padres asignados a estos, el problema que podemos apreciar es cuando en una escena hacemos uso de varios objetos. En el momento de organizar los objetos por nodos para mantener un orden en el proyecto, el usuario no puede apreciar cuando selecciona, por ejemplo, un Nodo Vacío, la selección de objetos que pertenezcan a este nodo. Podría ser más intuitivo para el usuario poder apreciar todos los objetos que pertenecen a ese nodo de forma de un contorno alrededor de estos como se puede ver en la imagen.



**Ilustración 1 – ejemplo del cambio a aplicar**

Actualmente en Godot 4.2 lo que se ve en la **ilustración 1** no se ve implementado en el motor, Cuando seleccionamos el Nodo “Paredes” desde los objetos, se puede ver como en la Escena los tres objetos que pertenecen a ese nodo se ven resaltados por un recuadro naranja(Pared1, Pared2, Pared3).

**Implementación en C++**

La forma en la que se implementó este cambio utilizando los conceptos antes mencionados es la siguiente.

**Clase Nodo:**

Esta clase representa un nodo dentro del sistema. Cada nodo tiene un nombre y un estado (seleccionado o no seleccionado).

**Función convertirEventoAManejador**

Esta función toma un evento (como "seleccionar" o "deseleccionar") y el nodo afectado, y devuelve una función de callbacks que se encargará de ejecutar la acción correspondiente sobre el nodo.

**Función DespachadorEventos**

Crea un mapa de nodos, y permite al usuario seleccionar o deseleccionar nodos

**Estas funciones se pueden apreciar en el código proporcionado desde el GitHub proporcionado al final del documento**

En la implementación de este código se utilizaron los conceptos vistos en la unidad 1 de la materia Programación Generica y Eventos junto con otras funciones proporcionadas por el entorno de trabajo, los cuales por mencionarlos son:

Bucle despachador de eventos: Captura eventos del usuario y los mapea a manejadores apropiados.

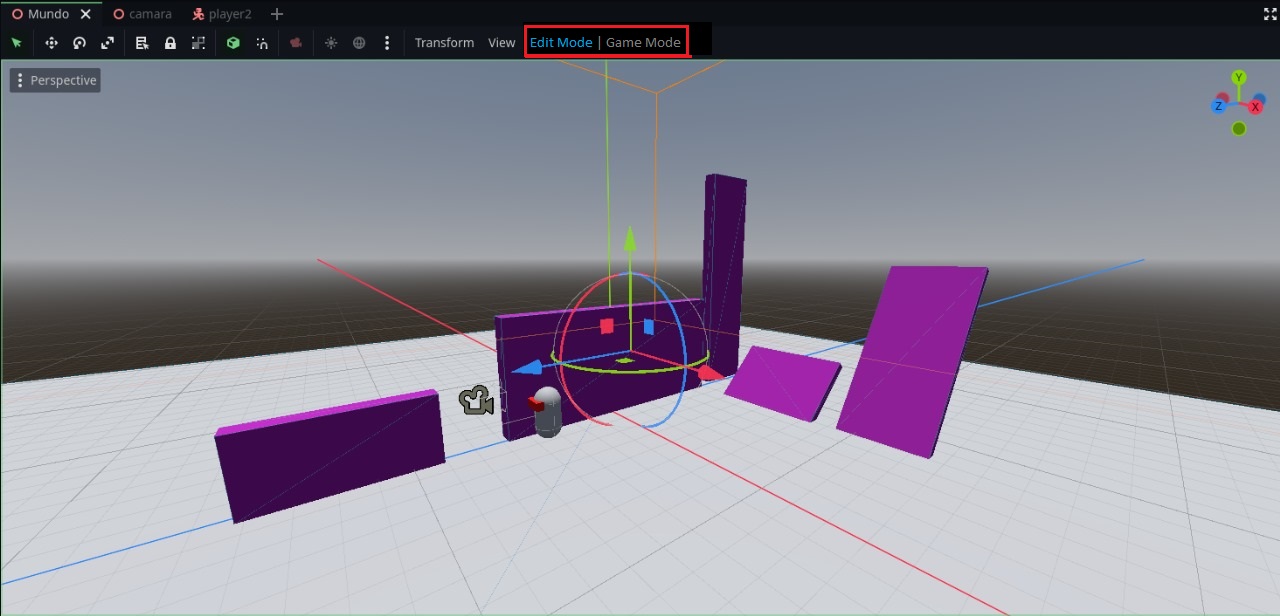
Callbacks: Las acciones sobre los nodos (selección y deselección) son ejecutadas como callbacks.

Conversión de eventos: Mapea eventos del usuario a manejadores de eventos específicos del nodo seleccionado.

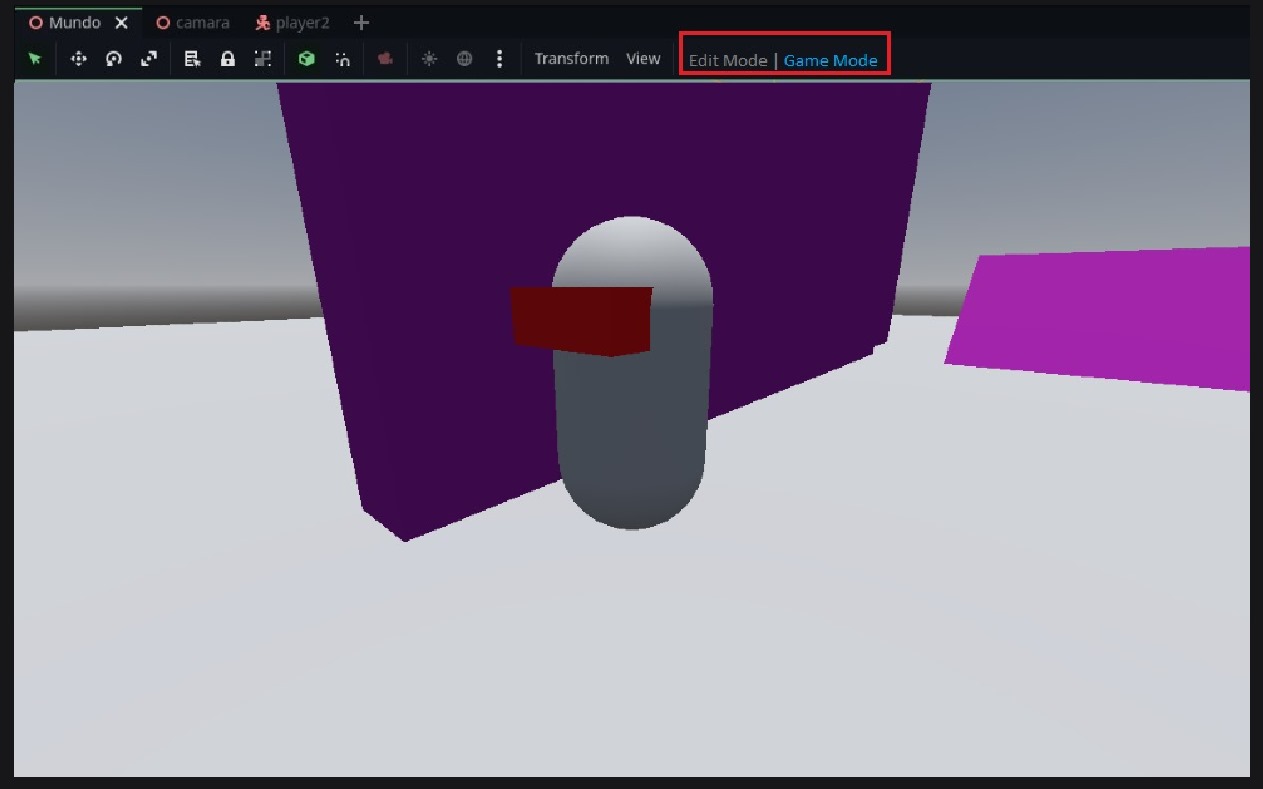
Mapeo de nodos: Los nodos se gestionan mediante un mapa que asocia los nombres de los nodos con sus instancias.

1. **Edición de juego en tiempo real:**

Como en otros motores gráficos, tales como Unity, sería una buena implementación para los usuarios que, al momento de ejecutar el juego, también poder modificarlo en tiempo real. Con esto los usuarios de Godot podrías modificar sus errores en tiempo real y ver como estos cambios afectan a la escena.  
Con el fin de ayudar a los desarrolladores, se puede implementar la creación de dos ventanas las cuales funcionen en conjunto una haciéndose cargo de los objetos de la escena y otra ejecutando el juego, sin necesidad de una ventana emergente como en la versión actual de Godot. Mientras el juego se ejecuta, el usuario puede modificar los elementos de la escena a su gusto, con el fin de evitar errores arreglándolos en el acto, por ejemplo, el comportamiento de un NPC u objetos colisionando y generando errores graves, también se puede aprovechar para modificar estéticamente el juego. Este cambio ayudaría al usuario en el desarrollo de su juego permitiéndole adaptar su juego a lo que más desee.



**Ilustración 2 – Modo Edición**



**Ilustración 3 – Modo Juego**

Como se puede ver en la **ilustración 2 y 3**, el usuario podría cambiar del Modo Juego al Modo Edicion solo haciendo click en los botones que se encuentran en la parte superior de la pantalla.

**Conclusión:**

Como conclusión a este prototipo de implementación a desarrollar en Godot se puede tener en cuenta que este cambio puede ayudar mucho a los usuarios o desarrolladores de videojuegos con este software a tener una mejor noción de los objetos que son parte de un Nodo en la escena principal, con el fin de evitar errores o problemas a largo plazo con la utilización de varios objetos y ayudar a la organización del proyecto de los usuarios.

**Codigo en Github:** <https://github.com/ManuNicolais/PGE/tree/main/Parcial%201>

<https://github.com/gonzarem29/PGE/tree/main/Parcial%201>