UNIVERSIDAD

Blas Pascal

Programación Genérica y Eventos

Informe técnico del software analizado

Integrante:

-Sofia Julieta Coniglio

Profesor:

-Mónica Liliana Nano

Introducción

En el presente trabajo se analiza la plataforma Steam, uno de los softwares de distribución digital de videojuegos más utilizados a nivel mundial. Steam ofrece a los usuarios la posibilidad de comprar, descargar, instalar y gestionar videojuegos, además de brindar servicios complementarios como chat, comunidad, foros y soporte técnico.

Si bien es una herramienta ampliamente reconocida y consolidada, su interfaz actual presenta diversos problemas de usabilidad y sobrecarga visual, los cuales pueden dificultar la experiencia del usuario. Entre ellos se destacan la duplicación de menús, filtros desordenados, gestión poco clara de descargas, botones mal ubicados y exceso de elementos en pantalla.

El objetivo principal de este trabajo es:

- Identificar defectos de usabilidad presentes en la interfaz de Steam.
- Proponer y desarrollar un rediseño simplificado que optimice la navegación, organización y experiencia del usuario.
- Implementar un prototipo en C++ con Win32 y GDI+, que muestre cómo aplicar estos cambios en una interfaz funcional.

De esta forma, el trabajo busca demostrar cómo la aplicación de principios de diseño centrados en el usuario puede mejorar significativamente la eficiencia y la satisfacción en el uso de una plataforma tan masiva como Steam.

Análisis de la Interfaz de Steam

Problemas de Usabilidad Detectados

1.Duplicación y exceso de menús

Steam tiene múltiples menús con opciones repetidas (ejemplo: “Tienda” aparece en distintos lugares).

2. Filtros desordenados y extensos

Los géneros y categorías aparecen en listas demasiado largas y dispersas.

3. Gestión de descargas poco clara

La barra inferior de descargas es poco visible y no cuenta con alertas claras.

4. Botones mal ubicados

Algunas funciones como “Añadir producto” no están en lugares intuitivos.

5. Sobrecarga visual y desorden

Exceso de banners, publicidad y menús que distraen al usuario de lo principal: jugar o gestionar juegos.

Prototipo Simplificado en Win32 (Implementación)

Para demostrar las mejoras se desarrolló un **prototipo funcional en C++ utilizando la API Win32 y la librería GDI+**.

Las características principales de esta interfaz son:

1. **Menú superior unificado y claro**
    a. Cuatro botones principales: *Steam*, *Biblioteca*, *Tienda* y *Comunidad*.
    b. Un buscador fijo en la esquina superior derecha.
2. **Sección de destacados y recomendaciones**
    a. Un área central que muestra una imagen grande del juego seleccionado.
    b. Se incluyeron ejemplos con *Warframe* y *CS:GO*.
3. **Botones inferiores para cambiar de juego**
    a. Dos botones permiten alternar entre los juegos destacados.
    b. La imagen de fondo cambia dinámicamente según la selección.
4. **Ajuste dinámico de la interfaz**
    a. Todos los controles (botones, buscador, etiquetas, imágenes) se adaptan automáticamente al tamaño de la ventana.

b. Esto garantiza que la interfaz se vea correctamente tanto en ventana pequeña como maximizada.

Codigo:

#include <windows.h> #include <gdiplus.h> #include #include <windowsx.h> // Para macros de edición (GetWindowTextW, etc.)

using namespace Gdiplus;

#pragma comment (lib,"Gdiplus.lib")

// Prototipo de la ventana LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam);

// Variables globales HBRUSH hBrushFondo; HBRUSH hBrushBoton; ULONG_PTR gdiplusToken;

// Imágenes Image* imgFondo = nullptr; Image* imgWarframe = nullptr; Image* imgCSGO = nullptr;

// Botones globales HWND btnSteam, btnBiblioteca, btnTienda, btnComunidad, txtBuscar; HWND btnWarframe, btnCSGO;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) { // Inicializar GDI+ GdiplusStartupInput gdiplusStartupInput; GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL);

```
// Pinceles globales
hBrushFondo = CreateSolidBrush(RGB(20, 40, 80));
hBrushBoton = CreateSolidBrush(RGB(30, 60, 120));

// Registrar ventana
WNDCLASS wc = { 0 };
wc.lpfnWndProc = WndProc;
wc.hInstance = hInstance;
wc.lpszClassName = L"SteamUI";
wc.hbrBackground = hBrushFondo;
wc.hCursor = LoadCursor(NULL, IDC_ARROW);

RegisterClass(&wc);
```

```
HWND hwnd = CreateWindowEx(
    0, L"SteamUI", L"Mi Steam Simplificado",
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, CW_USEDEFAULT, 1200, 700,
    NULL, NULL, hInstance, NULL
);

ShowWindow(hwnd, nCmdShow);
UpdateWindow(hwnd);

MSG msg;
while (GetMessage(&msg, NULL, 0, 0)) {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}

// Liberar recursos
if (hBrushFondo) DeleteObject(hBrushFondo);
if (hBrushBoton) DeleteObject(hBrushBoton);
GdiplusShutdown(gdiplusToken);

return (int)msg.wParam;


}
```

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) { switch (msg) { case WM_CREATE: // Cargar imágenes de juegos (usar rutas relativas) imgWarframe = Image::FromFile(L"Recursos\warframe.png"); imgCSGO = Image::FromFile(L"Recursos\csgo.png");

```
   // Mostrar Warframe por defecto si existe
    imgFondo = (imgWarframe) ? imgWarframe : nullptr;

    // Menú superior
    btnSteam = CreateWindow(L"BUTTON", L"Steam",
        WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
        20, 10, 120, 30, hwnd, (HMENU)1, NULL, NULL);

    btnBiblioteca = CreateWindow(L"BUTTON", L"Biblioteca",
        WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
        150, 10, 120, 30, hwnd, (HMENU)2, NULL, NULL);
```

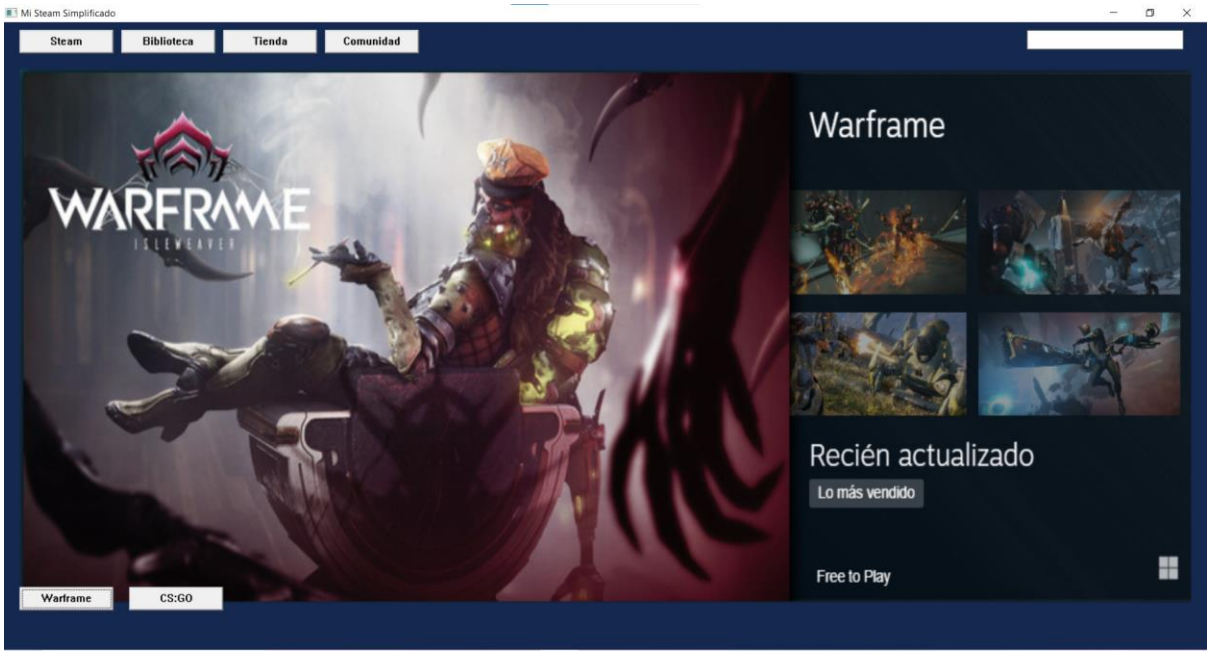
```
    btnTienda = CreateWindow(L"BUTTON", L"Tienda",
        WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
        280, 10, 120, 30, hwnd, (HMENU)3, NULL, NULL);

    btnComunidad = CreateWindow(L"BUTTON", L"Comunidad",
        WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
        410, 10, 120, 30, hwnd, (HMENU)4, NULL, NULL);

    // Cuadro de búsqueda
    txtBuscar = CreateWindow(L"EDIT", L"",
        WS_CHILD | WS_VISIBLE | WS_BORDER | ES_AUTOHSCROLL,
        950, 10, 200, 25, hwnd, (HMENU)10, NULL, NULL);



    // Botones de juegos
    btnWarframe = CreateWindow(L"BUTTON", L"Warframe",
        WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
        20, 600, 120, 30, hwnd, (HMENU)20, NULL, NULL);

    btnCSGO = CreateWindow(L"BUTTON", L"CS:GO",
        WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
        150, 600, 120, 30, hwnd, (HMENU)21, NULL, NULL);



case WM_PAINT: {
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hwnd, &ps);

    if (imgFondo) {
        Graphics graphics(hdc);

        RECT rect;
        GetClientRect(hwnd, &rect);

        // Espacio para botones arriba y abajo
        int x = 20;
        int y = 60;
        int ancho = rect.right - 40;
        int alto = rect.bottom - 120;
```

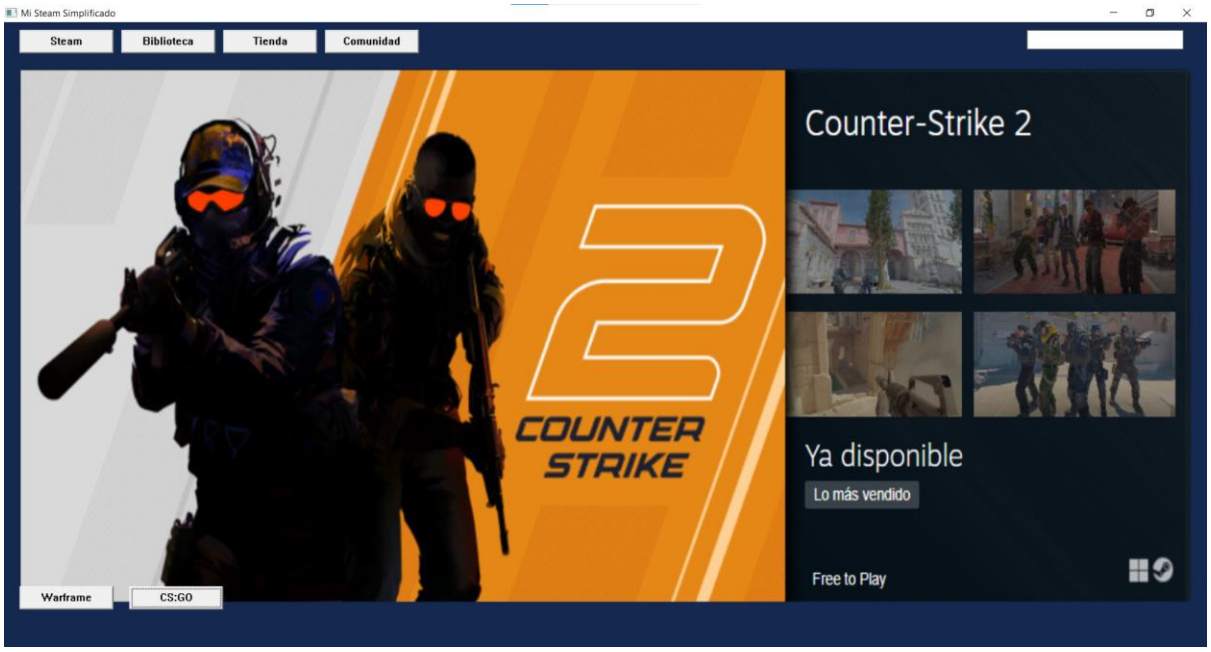
```
        graphics.DrawImage(imgFondo, x, y, ancho, alto);
    }

    EndPaint(hwnd, &ps);
} break;

case WM_COMMAND:
    switch (LOWORD(wParam)) {
    case 1:
        MessageBox(hwnd, L"Abriste Steam (configuracion, cambiar
cuenta, soporte, productos.)", L"Steam", MB_OK);
        break;
    case 2:
        MessageBox(hwnd, L"Biblioteca de juegos (juegos instalados,
incorporar producto)", L"Biblioteca", MB_OK);
        break;
    case 3:
        MessageBox(hwnd, L"Tienda (recomendaciones, categorias,
ofertas.)", L"Tienda", MB_OK);
        break;
    case 4:
        MessageBox(hwnd, L"Comunidad (foros, workshop, noticias,
amigos.)", L"Comunidad", MB_OK);
        break;
    case 20: // Botón Warframe
        if (imgWarframe) imgFondo = imgWarframe;
        InvalidateRect(hwnd, NULL, TRUE);
        break;
    case 21: // Botón CS:GO
        if (imgCSGO) imgFondo = imgCSGO;
        InvalidateRect(hwnd, NULL, TRUE);
        break;

    }

    // Filtro de búsqueda
    if (HIWORD(wParam) == EN_CHANGE && LOWORD(wParam) == 10) {
        wchar_t buffer[256];
        GetWindowTextW(txtBuscar, buffer, 256);

        if (wcsstr(buffer, L"gratis")) {
            MessageBox(hwnd, L"Mostrando juegos Free to Play...",
```

```
L"Filtro", MB_OK);
        }
        else if (wcsstr(buffer, L"Warframe")) {
            MessageBox(hwnd, L"Warframe encontrado. Estado:
Instalado.", L"Resultado", MB_OK);
        }
        else if (wcsstr(buffer, L"CSGO")) {
            MessageBox(hwnd, L"Counter Strike: GO encontrado.
Estado: No instalado.", L"Resultado", MB_OK);
        }
    }
    break;

case WM_CTLCOLORBTN: {
    HDC hdc = (HDC)wParam;
    SetBkColor(hdc, RGB(30, 60, 120));
    SetTextColor(hdc, RGB(255, 255, 255));
    return (LRESULT)hBrushBoton;
}
case WM_SIZE: {
    int width = LOWORD(lParam);
    int height = HIWORD(lParam);

    // Menú superior
    MoveWindow(btnSteam, 20, 10, 120, 30, TRUE);
    MoveWindow(btnBiblioteca, 150, 10, 120, 30, TRUE);
    MoveWindow(btnTienda, 280, 10, 120, 30, TRUE);
    MoveWindow(btnComunidad, 410, 10, 120, 30, TRUE);
    MoveWindow(txtBuscar, width - 230, 10, 200, 25, TRUE);



    // Botones de juegos (abajo de la ventana)
    MoveWindow(btnWarframe, 20, height - 80, 120, 30, TRUE);
    MoveWindow(btnCSGO, 160, height - 80, 120, 30, TRUE);


    InvalidateRect(hwnd, NULL, TRUE);
    break;
}

case WM_DESTROY:
```

```
    if (imgWarframe) delete imgWarframe;
    if (imgCSGO) delete imgCSGO;
    imgFondo = nullptr;
    PostQuitMessage(0);
    break;
}

return DefWindowProc(hwnd, msg, wParam, lParam);


}
```

InterfazUI





Conclusión

El rediseño simplificado en Win32 demuestra cómo, eliminando redundancias y priorizando lo esencial, se puede lograr una interfaz mucho más clara y usable que la versión original de Steam.

- El **menú superior único** reduce la sobrecarga cognitiva.
- El **buscador visible** facilita encontrar juegos rápidamente.
- La **sección central de destacados** mantiene el foco en lo importante: los juegos.
- Los **botones de selección de juegos** muestran cómo se puede navegar de manera simple entre contenidos.
- El **ajuste dinámico al tamaño de ventana** garantiza flexibilidad y una buena experiencia en diferentes resoluciones.

En conclusión, este prototipo demuestra que aplicar principios de diseño centrado en el usuario permite no solo mejorar la estética, sino también la **eficiencia y la experiencia general** en plataformas digitales de gran escala.