مختصات موانع برد

برای بدست آوردن مختصات موانع ، به علت آنکه تعداد موانع زیاد است ، نمی توان مختصات موانع را به صورت دستی تنظیم کرد پس باید به دنبال یک روش مفید وبهینه برای یافتن این مختصات باشیم ، اما روش بهینه چیست؟ چطور می توان المان های زیادی را دریافت کرد و سپس مختصات آنها را خروجی داد؟

ابتدا فایل مورد نظر که قرار است مختصات موانع در آن نگهداری کنیم را میسازیم :

    ofstream file("board\_\_pos\_\_1.text");

سپس یک آرایه دو بعدی برای نگهداری اولیه مقادیر تعریف می کنیم ، و بعد از آن سه متغیر int تعریف می کنیم و یک رشته که قرار است بردی که قبلا آن را ساخته ایم را نگهداری کند ( مقدار رشته همان برد بازی است ، امابه علت نا مرتب شدن صفحه آن را حذف کردم .)

    long XY[2068][4];

    long x = 66,y = 75,n = 0;

    string L;

    با استفاده از حلقه زیر مختصات موانع را مشخص می کنیم ، به هر مانع 4 مقدار تعلق می گیرد که برای کشیدن برد کافی است ( منطق این حلقه هم این چنین است که اگر مانع افقی بود مقدار X آن متفاوت است اما اگر عمودی بود مقدار y آن متفاوت است.)

    for (int i = 0 ; i < L.size() ; i++){

        if (L[i] == ','){

            //cout << endl;

            x = 66 , y += 25;

        }

        else {

            //cout << L[i];

            if (L[i] == '|'){

                x += 10;

                XY[n][0] = x,XY[n][1] = y;

                XY[n][2] = x, XY[n][3] = y - 25;

                n++;

            }

            else if (L[i] == '\_'){

                XY[n][0] = x,XY[n][1] = y;

                x += 10;

                XY[n][2] = x, XY[n][3] = y;

                n++;

            }

            else if (L[i] == ' ') x += 10;

        }

    }

برای آنکه از آن مختصات استفاده کنیم باید آن را به صورت یک آرایه دوبعدی ذخیر کنیم :

    file << "{";

    for (int i = 0 ; i < n ; i++){

    file << "{" << " " << XY[i][0] << " ," << XY[i][1] << " ," << XY[i][2] << " ," << XY[i][3] << " }" << ",";

    }

    file << "}";

برنامه که به پایان رسید با استفاده از دستور زیر بسته می شود.

    file.close();

مختصات فضای خالی برد

برای بدست آوردن مختصات فضای خالی ، به علت آنکه تعداد فضای خالی هم زیاد است ، نمی توان مختصات آن را به صورت دستی تنظیم کرد پس باید به دنبال یک روش مفید وبهینه برای یافتن این مختصات باشیم ، اما روش بهینه چیست؟ چطور می توان المان های زیادی را دریافت کرد و سپس مختصات آنها را خروجی داد؟

ابتدا فایل مورد نظر که قرار است مختصات را در آن نگهداری کنیم را میسازیم :

    ofstream file("S\_\_pos\_\_.text");

سپس یک آرایه دو بعدی برای نگهداری اولیه مقادیر تعریف می کنیم ، و بعد از آن سه متغیر int تعریف می کنیم و یک رشته که قرار است بردی که قبلا آن را ساخته ایم را نگهداری کند ( مقدار رشته همان برد بازی است ، اما به علت نامرتب شدن صفحه آن را حذف کردم .)

    long XY[2068][3];

    long x = 66,y = 75,n = 0;

    string L;

    for (int i = 0 ; i < L.size() ; i++){

        if (L[i] == ','){

            //cout << endl;

            x = 66 , y += 25;

        }

        else {

            //cout << L[i];

            if (L[i] == '|') x += 10;

            else if (L[i] == '\_'){

                XY[n][0] = x,XY[n][2] = y;

                x += 10;

                XY[n][1] = x;

                n++;

            }

            else if (L[i] == ' ') {

                XY[n][0] = x,XY[n][2] = y;

                x += 10;

                XY[n][1] = x;

                n++;

            }

        }

    }

برای آنکه از آن مختصات استفاده کنیم باید آن را به صورت یک آرایه دوبعدی ذخیر کنیم :

    file << "{";

    for (int i = 0 ; i < n ; i++){

    file << "{" << " " << XY[i][0] << " ," << XY[i][1] << " ," << XY[i][2] << " }" << ",";

    }

    file << "}";

برنامه که به پایان رسید با استفاده از دستور زیر بسته می شود.

طراحی پنجره

این پروژه تنها با استفاده از یک کتابخانه نوشته شده است ، که آن کتابخانه windows.h است ، کتابخانه windows.h در واقع یک کتابخانه گرافیکی است ، که ما با استفاده از این کتابخانه ، پروژه را به صورت گرافیکی پیاده سازی کرده ایم ، در ادامه در رابطه با نوع پیاده سازی پروژه به صورت گرافیکی مطالبی را ارائه خواهیم داد ، که کد اصلی کل پروژه همین کد است و کد های قبلی فقط برای بدست آوردن داده مورد نیاز پروژه زده شده بود.

موارد بالا از جمله داده ها ، پارامترها و ماکرو بود که خوب است قبل از آنکه به بررسی کد بپردازیم آن را مطالعه کرده تا به یا نحوه عملکرد آنها آشنا شوید ، و بدانید که هر کدام چه هستند و چه کاربردی در کد ما دارند ، پس قبل از بررسی کد به موارد زیر می پردازیم :

: LRESULT نوعی داده است ، برای برگرداندن مقدار در windows API

CALLBACK : یک ماکرو که این تابع برای پردازش رویدادها استفاده می شود.

HWND: نوعی داده که یک هندل به یک پنجره در windows API است.

UINT: نوعی داده بدون علامت صحیح که معمولا برای شناسایی نوع پیام ارسال شده به پنجره استفاده می شود.

WPARAM : نوعی داده برای نگهداری اطلاعات اضافی پیام

LPARAM: نوعی داده که نسبت به WPARAM اندازه بزرگتری از داده را نگهداری می کند ؛ اما کاربردشان مانند هم است.

WINAPI : یک نوع فراخوانی که مقادیر ورودی به تابع WinMain را بررسی می کند ، همچنین پارامتر های ارسالی و ...

HINSTANCE: پارامتر اول ، به شما اجازه دسترسی به منابع را می دهد.

HINSTANCE: پارامتر دوم نشانگر نمونه والد است که در اکثر برنامه ها نادیده گرفته می شود.

LPSTR: پارامتر سوم یک رشته است که معمولا برای آرگومان های خط فرمان استفاده می شود.

nShowCmd : پارامتر چهارم تعیین می کند که پنجره چگونه نمایش داده شود.

حال که با موارد بالا آشنا شده اید می توانیم در ادامه به بررسی کد پروژه بپردازیم.

تابع کشیدن شئ

در حین اجرای بازی ما نیاز داریم که یک شئ در پنجره تعریف شده باشد و آن شئ در حین اجرای بازی وقتی کاربر روی دکمه مورد نظرش می زند آن شی مکانش تغییر کند ؛ به همین دلیل تابع زیر که مربوط به کشیدن آن شئ است را تعریف کرده ایم.

تابع زیر برای کشیدن یک مستطیل طراحی شده است ، پارامتر HDC که برای این تابع مشخص شده

است در واقع در درون خود اطلاعات مربوط به دستگاه گرافیکی را نگه داری می کند.

void Draw(HDC hdc)}

Rectangle(hdc, pos\_O\_x , pos\_O\_y , pos\_O\_x + 10 , pos\_O\_y + 10);

{

ساختن لیبل

برای آنکه به پروژه یک هویت کلاسی بدهیم ( به این معنا که این پروژه برای یک درس مشخص طراحی شده است این لیبل را ایجاد کرده ایم.

case WM\_CREATE: }

HRGN hRgn = CreateRoundRectRgn(20, 20, 400, 50, 50, 50);

HWND Label = CreateWindow("STATIC", "یک نوشته"

WS\_VISIBLE | WS\_CHILD, 15, 5, 1040, 20,

hwnd, nullptr, nullptr, nullptr);

SetWindowRgn(Label, hRgn, TRUE);

break;

{

تنظیم سایز پنجره

برای آنکه کاربر نتواند اندازه پنجره را تغییر دهد ما یک اندازه مشخص برای پنجره تعیین کرده ایم. و زمانیکه اندازه پنجره تغییر کند این پیام ( WM\_SIZE ) به برنامه ارسال می شود.

case WM\_SIZE:

SetWindowPos(hwnd, NULL, 0, 0, 1080, 720, SWP\_NOZORDER);

return 0;

تنظیم کشیدن خطوط درون پنجره

case WM\_PAINT:

}

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hwnd, &ps);

Draw(hdc);

HPEN Maze\_Board\_G = CreatePen(PS\_SOLID, 10, RGB(40, 152, 7));

for (int i = 0 ; i < 1161 ; i++)}

if ((XY[i][1] > 65) and (XY[i][0] > 66))}

SelectObject(hdc, Maze\_Board\_G);

MoveToEx(hdc,XY[i][0] ,XY[i][1] , NULL);

LineTo(hdc, XY[i][2] ,XY[i][3]);

{

{

EndPaint(hwnd, &ps);

تنظیم دکمه های ورودی و عملکرد آن

case WM\_KEYDOWN:

else if (wParam == VK\_RIGHT)}

if (pos\_O\_x < X\_1)}

if (num\_enter == 1)}

W\_P[pos][0] = pos\_O\_x,W\_P[pos][1] = pos\_O\_y,W\_P[pos][3] = pos\_O\_y;

pos\_O\_x += 15;

W\_P[pos][2] = pos\_O\_x;

num\_enter = 0;

pos++;

{

else }

int pos\_\_x = pos\_O\_x;

for (int i = 0 ; i < 1691 ; i++)}

W\_P[pos][0] = pos\_O\_x,W\_P[pos][1] = pos\_O\_y,W\_P[pos][3] = pos\_O\_y;

if ((pos\_O\_y >= S[i][2] - 25) and (pos\_O\_y <= S[i][2]))}

if ((pos\_O\_x >= S[i][0]) and (pos\_O\_x <= S[i][1]))}

pos\_O\_x += 5;

W\_P[pos][2] = pos\_O\_x;

pos++;

break;

{

{

{

if (pos\_\_x == pos\_O\_x)}

if(n == 0)}

W\_P[pos][3] = pos\_O\_y;

pos++;

MessageBox(hwnd, "Please press Enter\_key then press (Up\_Down\_Left\_Right)\_key", "Warning!!!", MB\_OK);

{

else }

W\_P[pos][3] = pos\_O\_y;

pos++;

MessageBox(hwnd, "Game Over!!!", "Noob", MB\_OK);

Sleep(2000);

ExitProcess(0);

{

{

{

{

}

تنظیم رنگ پس زمینه

case WM\_ERASEBKGND:

}

HDC hdc = (HDC)wParam;

RECT rect;

GetClientRect(hwnd, &rect);

HBRUSH brush = CreateSolidBrush(RGB(180, 170, 120));

FillRect(hdc, &rect, brush);

DeleteObject(brush);

return 1;

{

تنظیم تابع WINAPI

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE, LPSTR, int nShowCmd) {

const char CLASS\_NAME[] = "Sample Window Class";

WNDCLASS wc = {};

wc.lpfnWndProc = Window\_and\_Print\_Board;

wc.hInstance = hInstance;

wc.lpszClassName = CLASS\_NAME;

RegisterClass(&wc);

HWND hwnd = CreateWindowEx(

0,

CLASS\_NAME,

"Maze Project",

WS\_OVERLAPPEDWINDOW & ~WS\_MAXIMIZEBOX,

CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT,

NULL,

NULL,

hInstance,

NULL

);

ShowWindow(hwnd, nShowCmd);

MSG msg;

while (GetMessage(&msg, NULL, 0, 0)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}