**注意编码习惯，善用常量而不是常数；适当注释**

绘图练习题：

1. 如右图效果，在窗口中绘制国旗，注意大、小五角星的位置、角度，国旗的宽高比例等。

#include <Windows.h>

#define WINDOW\_WIDTH 950

#define WINDOW\_HEIGHT 650

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

VOID PAINT(HWND hwnd, HDC hdc);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInst, LPSTR lpszCmdLine, int nCmdShow)

{

HWND hwnd;

MSG Msg;

WNDCLASS wndclass;

TCHAR lpszTitle[] = L"余庆祥设计";

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

wndclass.lpszMenuName = NULL;

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

wndclass.lpszClassName = L"xiang";

if (!RegisterClass(&wndclass))

{

MessageBox(NULL, L"注册窗口失败", L"注册窗口失败", MB\_ICONERROR);

return 0;

}

hwnd = CreateWindow(L"xiang", lpszTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT, WINDOW\_WIDTH, WINDOW\_HEIGHT,

NULL, NULL, hInstance, NULL);

if (!hwnd)

{

MessageBox(NULL, L"创建窗口失败", L"创建窗口失败", MB\_ICONERROR);

return 0;

}

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

while (GetMessage(&Msg, NULL, 0, 0))

{

TranslateMessage(&Msg);

DispatchMessage(&Msg);

}

return Msg.wParam;

}

HDC hdc;

PAINTSTRUCT ps;//

HBRUSH hBrush1, hOldBrush;//

HPEN hOldPen; // 初始时的画笔样式

HPEN hPen1 = CreatePen(PS\_NULL, 0, RGB(0, 0, 0)); // 画笔样式为：空线

#include <cmath>////

#define PI 3.14159265358979323846////

int nWidth = 900 / 2 / 15;

int nHeight = 600 / 2 / 10;

void SetFivePoints(POINT\* pts, int nCount, int r, int angle, POINT pOrg);////

void DrawFiveStart(HDC hdc);////

void DrawSmallFiveStart(HDC hdc, int angle, POINT ptOrg);////

LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_PAINT:

hdc = GetDC(hwnd); // 获取设备环境

PAINT(hwnd, hdc); // 绘制

ReleaseDC(hwnd, hdc); // 释放设备环境

break;

case WM\_DESTROY:

DeleteObject(hPen1); // 删除画笔样式hPen1

DeleteObject(hBrush1); // 删除笔刷样式hBrush1

SelectObject(hdc, hOldPen); // 恢复初始时的画笔样式

SelectObject(hdc, hOldBrush); // 恢复初始时的笔刷样式

PostQuitMessage(0);

return 0;

}

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

}

VOID PAINT(HWND hwnd, HDC hdc)

{

hBrush1 = (HBRUSH)CreateSolidBrush(RGB(255, 0, 0));//

hOldBrush = (HBRUSH)SelectObject(hdc, hBrush1);//

// 红色背景

Rectangle(hdc, 0, 0, 900, 600);//

// 画出五角星

hBrush1 = (HBRUSH)CreateSolidBrush(RGB(250, 244, 8));//

hOldBrush = (HBRUSH)SelectObject(hdc, hBrush1);//

hOldPen = (HPEN)SelectObject(hdc, hPen1);////

DrawFiveStart(hdc);////

// 画出小的五角星

DrawSmallFiveStart(hdc, 70, { nWidth \* 10, nHeight \* 2 });////

DrawSmallFiveStart(hdc, 120, { nWidth \* 12, nHeight \* 4 });////

DrawSmallFiveStart(hdc, 90, { nWidth \* 12, nHeight \* 7 });////

DrawSmallFiveStart(hdc, 70, { nWidth \* 10, nHeight \* 9 });////

}

void SetFivePoints(POINT\* pts, int nCount, int r, int angle, POINT pOrg)////

{

for (int i = 0; i < nCount; i++)////

{

pts[i].x = pOrg.x + (LONG)(r \* cos((angle + i \* 72) \* PI / 180));////

pts[i].y = pOrg.y - (LONG)(r \* sin((angle + i \* 72) \* PI / 180));////

}

}

void DrawFiveStart(HDC hdc)

{

// 画出五角星

POINT pts[5] = { 0 };////

POINT pOrg = { 0 };////

pOrg.x = 5 \* nWidth;////

pOrg.y = 5 \* nHeight;////

SetPolyFillMode(hdc, WINDING);

SetFivePoints(pts, 5, 35 \* 3, 90, pOrg);////

POINT pNewPts[5] = { pts[0], pts[2], pts[4], pts[1], pts[3] };////

Polygon(hdc, pNewPts, 5);////

}

void DrawSmallFiveStart(HDC hdc, int angle, POINT ptOrg)////

{

POINT pts[5] = { 0 };////

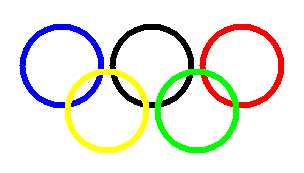
SetFivePoints(pts, 5, nHeight, angle, ptOrg);////

POINT pNewPts[5] = { pts[0], pts[2], pts[4], pts[1], pts[3] };////

Polygon(hdc, pNewPts, 5);////

}



1. 如右图效果，绘制奥运五环，注意画刷应该使用空画刷NULL\_BRUSH，才能形成圆环（透明）的效果。

#include <Windows.h>

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

VOID PAINT(HWND hwnd, HDC hdc);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInst, LPSTR lpszCmdLine, int nCmdShow)

{

HWND hwnd;

MSG Msg;

WNDCLASS wndclass;

TCHAR lpszTitle[] = L"余庆祥设计";

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

wndclass.lpszMenuName = NULL;

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

wndclass.lpszClassName = L"xiang";

if (!RegisterClass(&wndclass))

{

MessageBox(NULL, L"注册窗口失败", L"注册窗口失败", MB\_ICONERROR);

return 0;

}

hwnd = CreateWindow(L"xiang", lpszTitle, WS\_OVERLAPPEDWINDOW,

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

NULL, NULL, hInstance, NULL);

if (!hwnd)

{

MessageBox(NULL, L"创建窗口失败", L"创建窗口失败", MB\_ICONERROR);

return 0;

}

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

while (GetMessage(&Msg, NULL, 0, 0))

{

TranslateMessage(&Msg);

DispatchMessage(&Msg);

}

return Msg.wParam;

}

#define diameter 200 // 对象直径

#define radius diameter / 2 // 对象半径

#define init\_x 50 // 对象初始的x轴位置

#define init\_y 50 // 对象初始的y轴位置

#define width init\_x + diameter // 对象宽度

#define height init\_y + diameter // 对象长度

#define spacing\_x 16 // 对象之间的x轴间距

#define line\_width 8 // 画笔样式的宽度

HDC hdc;

HPEN hOldPen; // 初始时的画笔样式

HPEN hPen1 = CreatePen(PS\_SOLID, line\_width, RGB(0, 0, 255)); // 画笔样式为：实线6宽度蓝色

HPEN hPen2 = CreatePen(PS\_SOLID, line\_width, RGB(0, 0, 0)); // 画笔样式为：实线6宽度黑色

HPEN hPen3 = CreatePen(PS\_SOLID, line\_width, RGB(255, 0, 0)); // 画笔样式为：实线6宽度红色

HPEN hPen4 = CreatePen(PS\_SOLID, line\_width, RGB(255, 255, 0)); // 画笔样式为：实线6宽度黄色

HPEN hPen5 = CreatePen(PS\_SOLID, line\_width, RGB(0, 255, 0)); // 画笔样式为：实线6宽度绿色

LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_PAINT:

hdc = GetDC(hwnd); // 获取设备环境

PAINT(hwnd, hdc); // 绘制

ReleaseDC(hwnd, hdc); // 释放设备环境

break;

case WM\_DESTROY:

DeleteObject(hPen1); // 删除画笔样式hPen1

DeleteObject(hPen2); // 删除画笔样式hPen2

DeleteObject(hPen3); // 删除画笔样式hPen3

DeleteObject(hPen4); // 删除画笔样式hPen4

DeleteObject(hPen5); // 删除画笔样式hPen5

SelectObject(hdc, hOldPen); // 恢复初始时的画笔样式

PostQuitMessage(0);

return 0;

}

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

}

VOID PAINT(HWND hwnd, HDC hdc)

{

hOldPen = (HPEN)SelectObject(hdc, hPen1); // 保存初始时的画笔样式，并设置画笔为hPen1样式

Arc(hdc, init\_x, init\_y,

width, height, 0, 0, 0, 0); // 绘制圆弧

SelectObject(hdc, hPen2); // 设置画笔为hPen2样式

Arc(hdc, init\_x + diameter \* 1 + spacing\_x \* 1, init\_y,

width + diameter \* 1 + spacing\_x \* 1, height, 0, 0, 0, 0); // 绘制圆弧

SelectObject(hdc, hPen3); // 设置画笔为hPen3样式

Arc(hdc, init\_x + diameter \* 2 + spacing\_x \* 2, init\_y,

width + diameter \* 2 + spacing\_x \* 2, height, 0, 0, 0, 0); // 绘制圆弧

SelectObject(hdc, hPen4); // 设置画笔为hPen4样式

Arc(hdc, init\_x + radius, init\_y + radius,

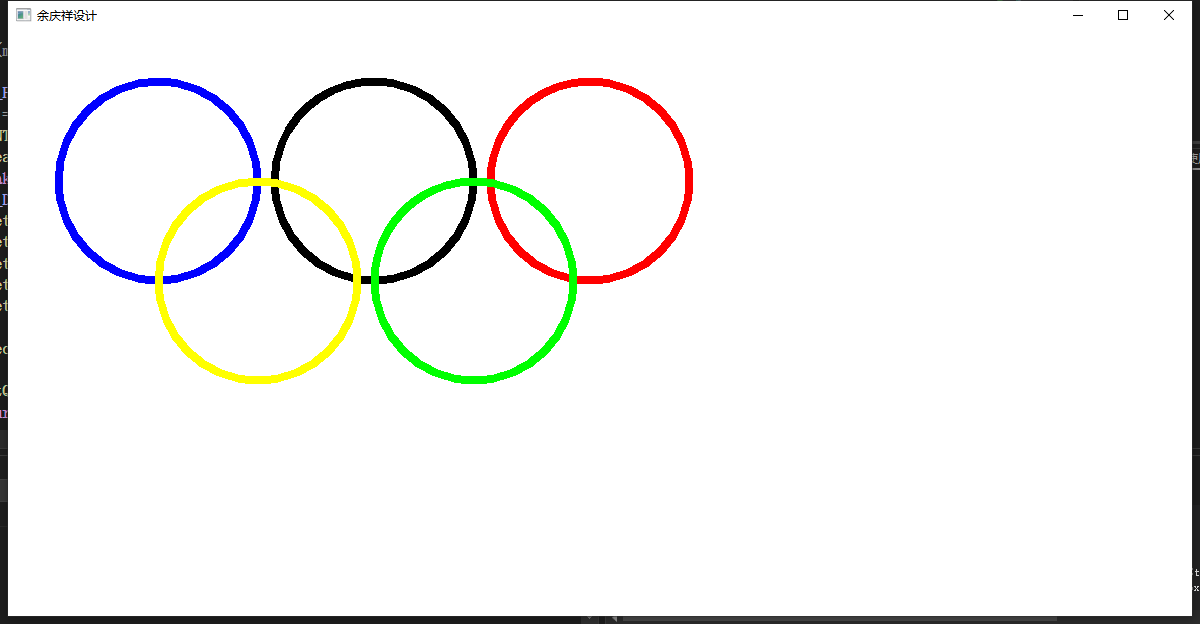
width + radius, height + radius, 0, 0, 0, 0); // 绘制圆弧

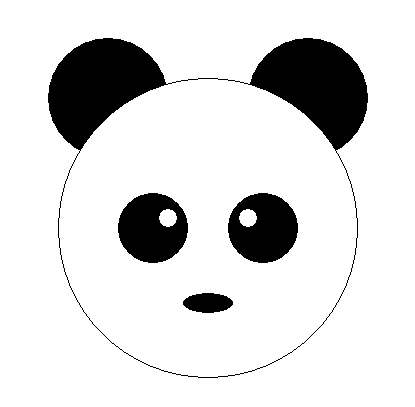
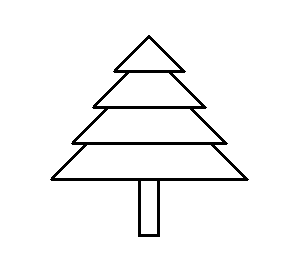
SelectObject(hdc, hPen5); // 设置画笔为hPen5样式

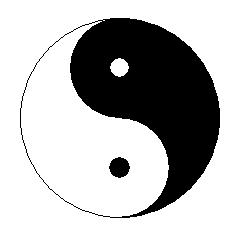
Arc(hdc, init\_x + radius + diameter \* 1 + spacing\_x \* 1, init\_y + radius,

width + radius + diameter \* 1 + spacing\_x \* 1, height + radius, 0, 0, 0, 0); // 绘制圆弧

}



3、绘制以下图形：



#include <Windows.h>

typedef struct {

float x;

float y;

} objCenterXY;

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

VOID PAINT(HWND hwnd, HDC hdc); // 声明绘画函数

void drawCircular(HDC hdc, objCenterXY obj\_center, float obj\_radius); // 声明绘制圆形函数

void drawCircularArc(HDC hdc, const objCenterXY obj\_center, float obj\_radius); // 声明绘制圆形边框函数

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInst, LPSTR lpszCmdLine, int nCmdShow)

{

HWND hwnd;

MSG Msg;

WNDCLASS wndclass;

TCHAR lpszTitle[] = L"余庆祥设计";

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

wndclass.lpszMenuName = NULL;

wndclass.hbrBackground = (HBRUSH)GetStockObject(GRAY\_BRUSH);

wndclass.lpszClassName = L"xiang";

if (!RegisterClass(&wndclass))

{

MessageBox(NULL, L"注册窗口失败", L"注册窗口失败", MB\_ICONERROR);

return 0;

}

hwnd = CreateWindow(L"xiang", lpszTitle, WS\_OVERLAPPEDWINDOW,

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

NULL, NULL, hInstance, NULL);

if (!hwnd)

{

MessageBox(NULL, L"创建窗口失败", L"创建窗口失败", MB\_ICONERROR);

return 0;

}

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

while (GetMessage(&Msg, NULL, 0, 0))

{

TranslateMessage(&Msg);

DispatchMessage(&Msg);

}

return Msg.wParam;

}

#define init\_x 50 // 全局的初始的x轴位置

#define init\_y 50 // 全局的初始的y轴位置

#define diameter 200 // 全局的一个直径参考

#define line\_width 1 // 画笔样式的默认宽度

#define object\_spacing 100 // 两个对象之间的间隙

HDC hdc;

HPEN hOldPen; // 初始时的画笔样式

HPEN hPen1 = CreatePen(PS\_NULL, line\_width, RGB(255, 255, 255)); // 画笔样式为：无边

HPEN hPen2 = CreatePen(PS\_SOLID, line\_width, RGB(0, 0, 0)); // 画笔样式为：实线1宽度黑色

HBRUSH hOldBrush; // 初始时的笔刷样式

HBRUSH hBrush1 = CreateSolidBrush(RGB(255, 255, 255)); // 笔刷样式为：白色实心笔刷

HBRUSH hBrush2 = CreateSolidBrush(RGB(0, 0, 0)); // 笔刷样式为：黑色实心笔刷

LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_PAINT:

hdc = GetDC(hwnd); // 获取设备环境

PAINT(hwnd, hdc); // 绘制

ReleaseDC(hwnd, hdc); // 释放设备环境

break;

case WM\_DESTROY:

DeleteObject(hPen1); // 删除画笔样式hPen1

DeleteObject(hPen2); // 删除画笔样式hPen2

DeleteObject(hBrush1); // 删除笔刷样式hBrush1

DeleteObject(hBrush2); // 删除笔刷样式hBrush2

SelectObject(hdc, hOldPen); // 恢复初始时的画笔样式

SelectObject(hdc, hOldBrush); // 恢复初始时的笔刷样式

PostQuitMessage(0);

return 0;

}

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

}

VOID PAINT(HWND hwnd, HDC hdc)

{

// 绘制两仪

#define Liangyi\_init\_x init\_x // 两仪初始的x轴位置

#define Liangyi\_init\_y init\_y // 两仪初始的y轴位置

#define Liangyi\_radius diameter / 2 // 两仪半径

#define Liangyi\_small\_radius Liangyi\_radius / 2 // 小两仪的半径

#define Liangyi\_eye\_radius 10 // 两仪的小圆半径

objCenterXY liangyi = {

Liangyi\_init\_x + Liangyi\_radius,

Liangyi\_init\_y + Liangyi\_radius,

}; // 定义两仪的中心位置

objCenterXY liangyi\_up\_cir = {

liangyi.x,

Liangyi\_init\_y + Liangyi\_radius / 2,

}; // 定义两仪的上半圆中心位置

objCenterXY liangyi\_down\_cir = {

liangyi.x,

Liangyi\_init\_y + Liangyi\_radius + Liangyi\_radius / 2,

}; // 定义两仪的下半圆中心位置

hOldPen = (HPEN)SelectObject(hdc, hPen2); // 保存初始时的画笔样式，并设置画笔为hPen2样式

hOldBrush = (HBRUSH)SelectObject(hdc, hBrush1); // 保存初始时的笔刷样式，并设置笔刷为hBrush1样式

Pie(hdc, liangyi.x - Liangyi\_radius, liangyi.y - Liangyi\_radius,

liangyi.x + Liangyi\_radius, liangyi.y + Liangyi\_radius,

liangyi.x, liangyi.y - Liangyi\_radius,

liangyi.x, liangyi.y + Liangyi\_radius); // 绘制两仪的左半圆

SelectObject(hdc, hBrush2); // 设置笔刷为hBrush2样式

Pie(hdc, liangyi\_up\_cir.x - Liangyi\_small\_radius, liangyi\_up\_cir.y - Liangyi\_small\_radius,

liangyi\_up\_cir.x + Liangyi\_small\_radius, liangyi\_up\_cir.y + Liangyi\_small\_radius,

liangyi\_up\_cir.x, liangyi\_up\_cir.y - Liangyi\_small\_radius,

liangyi\_up\_cir.x, liangyi\_up\_cir.y + Liangyi\_small\_radius); // 绘制两仪的上左半圆

Pie(hdc, liangyi.x - Liangyi\_radius, liangyi.y - Liangyi\_radius,

liangyi.x + Liangyi\_radius, liangyi.y + Liangyi\_radius,

liangyi.x, liangyi.y + Liangyi\_radius,

liangyi.x, liangyi.y - Liangyi\_radius); // 绘制两仪的右半圆

SelectObject(hdc, hPen1); // 设置画笔为hPen1样式

SelectObject(hdc, hBrush1); // 设置笔刷为hBrush1样式

Pie(hdc, liangyi\_down\_cir.x - Liangyi\_small\_radius, liangyi\_down\_cir.y - Liangyi\_small\_radius,

liangyi\_down\_cir.x + Liangyi\_small\_radius, liangyi\_down\_cir.y + Liangyi\_small\_radius,

liangyi\_down\_cir.x, liangyi\_down\_cir.y + Liangyi\_small\_radius,

liangyi\_down\_cir.x, liangyi\_down\_cir.y - Liangyi\_small\_radius); // 绘制两仪的下右半圆

drawCircular(hdc, liangyi\_up\_cir, Liangyi\_eye\_radius); // 绘制上半圆的白色小圆

SelectObject(hdc, hBrush2); // 设置笔刷为hBrush2样式

drawCircular(hdc, liangyi\_down\_cir, Liangyi\_eye\_radius); // 绘制下半圆的黑色小圆

// 绘制熊猫

#define panda\_init\_x Liangyi\_init\_x + diameter + object\_spacing // 熊猫初始的x轴位置

#define panda\_init\_y init\_y // 熊猫初始的y轴位置

#define panda\_ear\_radius 30 // 熊猫的耳朵的半径

#define panda\_Scale\_factor 0.6 // 熊猫的脸与耳朵的比例系数

#define panda\_face\_radius 80 // 熊猫的脸的半径

#define panda\_eye\_distance\_hafe 10 // 熊猫的一半眼距

#define panda\_eye\_radius 20 // 熊猫的眼睛的半径

#define panda\_mouth\_width 15 // 熊猫的嘴巴宽度

#define panda\_mouth\_height 6 // 熊猫的嘴巴高度

#define panda\_eyeball\_radius 6 // 熊猫眼球的半径

#define panda\_eyeball\_to\_eyeball\_distance 7 // 熊猫眼球中心与眼睛中心的位置

objCenterXY panda\_left\_ear = {

panda\_init\_x + panda\_ear\_radius,

panda\_init\_y + panda\_ear\_radius,

}; // 定义熊猫的左耳中心位置

objCenterXY panda\_right\_ear = {

panda\_init\_x + (panda\_face\_radius) \* 2 - panda\_ear\_radius,

panda\_init\_y + panda\_ear\_radius,

}; // 定义熊猫的右耳中心位置

objCenterXY pandan\_face = {

panda\_init\_x + panda\_face\_radius,

panda\_left\_ear.y + panda\_face\_radius - 10,

}; // 定义熊猫的脸中心位置

objCenterXY panda\_left\_eye = {

pandan\_face.x - panda\_eye\_distance\_hafe - panda\_eye\_radius,

pandan\_face.y - panda\_eye\_radius + 14,

}; // 定义熊猫的左眼中心位置

objCenterXY panda\_right\_eye = {

pandan\_face.x + panda\_eye\_distance\_hafe + panda\_eye\_radius,

pandan\_face.y - panda\_eye\_radius + 14,

}; // 定义熊猫的右眼中心位置

objCenterXY panda\_mouth = {

pandan\_face.x,

pandan\_face.y + 40,

}; // 定义熊猫的嘴巴中心位置

objCenterXY panda\_left\_eyeball = {

panda\_left\_eye.x + panda\_eyeball\_to\_eyeball\_distance,

panda\_left\_eye.y - panda\_eyeball\_to\_eyeball\_distance,

}; // 定义熊猫的左眼球中心位置

objCenterXY panda\_right\_eyeball = {

panda\_right\_eye.x - panda\_eyeball\_to\_eyeball\_distance,

panda\_right\_eye.y - panda\_eyeball\_to\_eyeball\_distance,

}; // 定义熊猫的右眼球中心位置

drawCircular(hdc, panda\_left\_ear, panda\_ear\_radius); // 绘制熊猫的左耳

drawCircular(hdc, panda\_right\_ear, panda\_ear\_radius); // 绘制熊猫的右耳

SelectObject(hdc, hBrush1); // 设置笔刷为hBrush1样式

drawCircular(hdc, pandan\_face, panda\_face\_radius); // 绘制熊猫脸

SelectObject(hdc, hPen2); // 设置画笔为hPen2样式

drawCircularArc(hdc, pandan\_face, panda\_face\_radius); // 绘制熊猫脸的边缘

SelectObject(hdc, hBrush2); // 设置笔刷为hBrush2样式

drawCircular(hdc, panda\_left\_eye, panda\_eye\_radius); // 绘制熊猫的左眼

drawCircular(hdc, panda\_right\_eye, panda\_eye\_radius); // 绘制熊猫的右眼

Pie(hdc, panda\_mouth.x - panda\_mouth\_width,

panda\_mouth.y - panda\_mouth\_height,

panda\_mouth.x + panda\_mouth\_width,

panda\_mouth.y + panda\_mouth\_height,

0, 0, 0, 0); // 绘制熊猫的嘴巴

SelectObject(hdc, hPen1); // 设置画笔为hPen1样式

SelectObject(hdc, hBrush1); // 设置笔刷为hBrush1样式

drawCircular(hdc, panda\_left\_eyeball, panda\_eyeball\_radius); // 绘制熊猫的左眼球

drawCircular(hdc, panda\_right\_eyeball, panda\_eyeball\_radius); // 绘制熊猫的右眼球

// 绘制树

#define height\_disparity 15 // 树之间的高度差

#define Tree1\_init\_y init\_y + height\_disparity \* 3 // 树1初始的y轴位置

#define Tree1\_height 80 // 树1的高度

#define Tree2\_init\_y Tree1\_init\_y - height\_disparity // 树2初始的y轴位置

#define Tree2\_height Tree1\_height - height\_disparity // 树2的高度

#define Tree3\_init\_y Tree2\_init\_y - height\_disparity // 树3初始的y轴位置

#define Tree3\_height Tree2\_height - height\_disparity // 树3的高度

#define Tree4\_init\_y Tree3\_init\_y - height\_disparity // 树4初始的y轴位置

#define Tree4\_height Tree3\_height - height\_disparity // 树4的高度

#define Tree\_init\_x panda\_init\_x + diameter + object\_spacing + Tree1\_height / 2 // 树初始的x轴位置

#define Trunk\_init\_y Tree1\_init\_y + Tree1\_height // 树干初始的y轴位置

#define Trunk\_width\_half 10 // 树干的一半宽度

#define Trunk\_height 40 // 树干的高度

SelectObject(hdc, hPen2); // 设置画笔为hPen2样式

POINT lpPt1[3] = { Tree\_init\_x - (Tree1\_height), Tree1\_init\_y + Tree1\_height,

Tree\_init\_x , Tree1\_init\_y,

Tree\_init\_x + Tree1\_height, Tree1\_init\_y + Tree1\_height }; // 定义第一丛树的三个点坐标

Polygon(hdc, lpPt1, 3); // 绘制第一丛树，从下往上数

POINT lpPt2[3] = { Tree\_init\_x - (Tree2\_height), Tree2\_init\_y + Tree2\_height,

Tree\_init\_x , Tree2\_init\_y,

Tree\_init\_x + Tree2\_height, Tree2\_init\_y + Tree2\_height }; // 定义第二丛树的三个点坐标

Polygon(hdc, lpPt2, 3); // 绘制第二丛树

POINT lpPt3[3] = { Tree\_init\_x - (Tree3\_height), Tree3\_init\_y + Tree3\_height,

Tree\_init\_x , Tree3\_init\_y,

Tree\_init\_x + Tree3\_height, Tree3\_init\_y + Tree3\_height }; // 定义第三丛树的三个点坐标

Polygon(hdc, lpPt3, 3); // 绘制第三丛树

POINT lpPt4[3] = { Tree\_init\_x - (Tree4\_height), Tree4\_init\_y + Tree4\_height,

Tree\_init\_x , Tree4\_init\_y,

Tree\_init\_x + Tree4\_height, Tree4\_init\_y + Tree4\_height }; // 定义第四丛树的三个点坐标

Polygon(hdc, lpPt4, 3); // 绘制第四丛树

Rectangle(hdc, Tree\_init\_x - Trunk\_width\_half, Trunk\_init\_y,

Tree\_init\_x + Trunk\_width\_half, Trunk\_init\_y + Trunk\_height); // 绘制树干

}

void drawCircular(HDC hdc, const objCenterXY obj\_center, float obj\_radius)

{

Pie(hdc, obj\_center.x - obj\_radius,

obj\_center.y - obj\_radius,

obj\_center.x + obj\_radius,

obj\_center.y + obj\_radius,

0, 0, 0, 0);

}

void drawCircularArc(HDC hdc, const objCenterXY obj\_center, float obj\_radius)

{

Arc(hdc, obj\_center.x - obj\_radius,

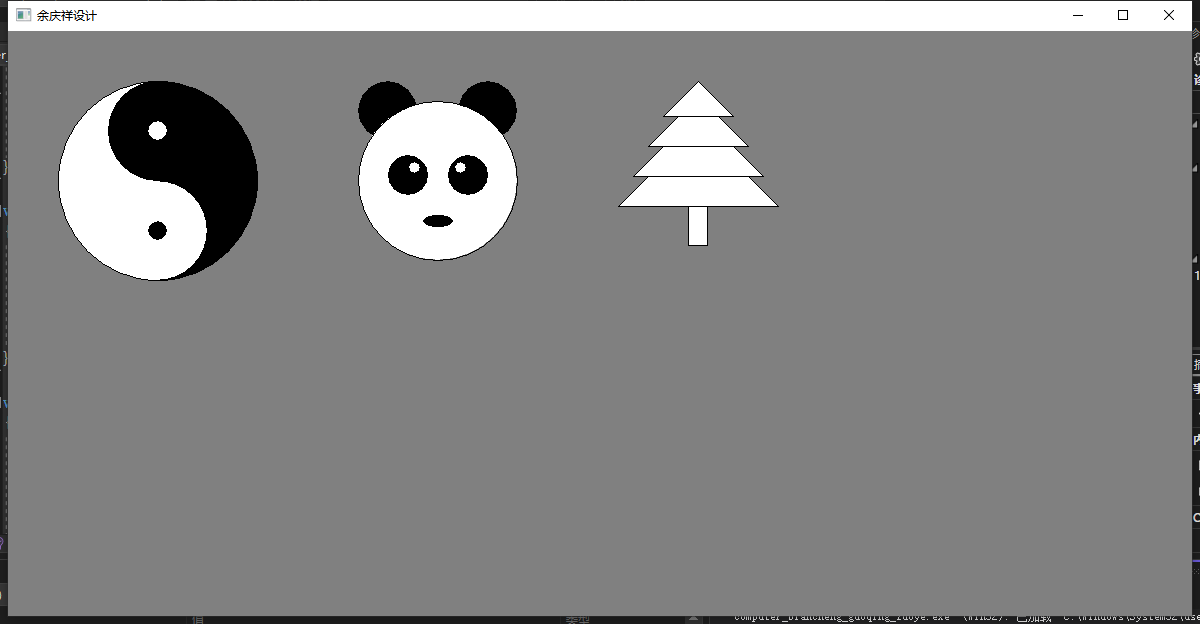
obj\_center.y - obj\_radius,

obj\_center.x + obj\_radius,

obj\_center.y + obj\_radius,

0, 0, 0, 0);

}



1. 使用SetPixcel函数画sin曲线、抛物线等等。

#include <Windows.h>

#include <cmath>

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

VOID PAINT(HWND hwnd, HDC hdc);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInst, LPSTR lpszCmdLine, int nCmdShow)

{

HWND hwnd;

MSG Msg;

WNDCLASS wndclass;

TCHAR lpszTitle[] = L"余庆祥设计";

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

wndclass.lpszMenuName = NULL;

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

wndclass.lpszClassName = L"xiang";

if (!RegisterClass(&wndclass))

{

MessageBox(NULL, L"注册窗口失败", L"注册窗口失败", MB\_ICONERROR);

return 0;

}

hwnd = CreateWindow(L"xiang", lpszTitle, WS\_OVERLAPPEDWINDOW,

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

NULL, NULL, hInstance, NULL);

if (!hwnd)

{

MessageBox(NULL, L"创建窗口失败", L"创建窗口失败", MB\_ICONERROR);

return 0;

}

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

while (GetMessage(&Msg, NULL, 0, 0))

{

TranslateMessage(&Msg);

DispatchMessage(&Msg);

}

return Msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

HDC hdc;

switch (message)

{

case WM\_PAINT:

hdc = GetDC(hwnd); // 获取设备环境

PAINT(hwnd, hdc); // 绘制

ReleaseDC(hwnd, hdc); // 释放设备环境

break;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

}

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

}

VOID PAINT(HWND hwnd, HDC hdc)

{

// sin曲线

#define PI 3.14159265358979323846

#define init\_x 300 // 原点的x轴位置

#define init\_y 150 // 原点的y轴位置

#define origin\_width init\_x // 原点的x轴宽度度

#define origin\_height 100 // 原点的y轴高度

// 绘制x坐标轴

for (int i = init\_x - origin\_width; i < init\_x + origin\_width; i++)

{

SetPixel(hdc, i, init\_y, RGB(0, 0, 0));

}

// 绘制y坐标轴

for (int i = init\_y - origin\_height; i < init\_y + origin\_height; i++)

{

SetPixel(hdc, init\_x, i, RGB(0, 0, 0));

}

// 绘制sin曲线

// 绘制x轴正方向的曲线

for (int i = init\_x, xita = 0; i < init\_x + origin\_width; i++, xita++)

{

xita %= 360;

SetPixel(hdc, i, sin(xita \* PI / 180) \* origin\_height + init\_y, RGB(255, 0, 0));

}

// 绘制x轴反方向的曲线

for (int i = init\_x, xita = 0; i > init\_x - origin\_width; i--, xita--)

{

xita %= 360;

SetPixel(hdc, i, sin(xita \* PI / 180) \* origin\_height + init\_y, RGB(255, 0, 0));

}

// 抛物线

#define distance\_y 200 // 两个对象之间y轴的间距

#define parabola\_init\_x init\_x // 原点的x轴位置

#define parabola\_init\_y init\_y + origin\_height + distance\_y // 原点的y轴位置

#define parabola\_origin\_width parabola\_init\_x - 100 // 原点的x轴宽度度

#define parabola\_origin\_height 150 // 原点的y轴高度

int p = 250; // 确定一个焦点位置

// 绘制x坐标轴

for (int i = parabola\_init\_x - (parabola\_origin\_width); i < parabola\_init\_x + parabola\_origin\_width; i++)

{

SetPixel(hdc, i, parabola\_init\_y, RGB(0, 0, 0));

}

// 绘制y坐标轴

for (int i = parabola\_init\_y - (parabola\_origin\_height); i < parabola\_init\_y + parabola\_origin\_height; i++)

{

SetPixel(hdc, parabola\_init\_x, i, RGB(0, 0, 0));

}

// 绘制抛物线

for (int x = -(parabola\_origin\_width / 2), y = 0; x < (parabola\_origin\_width / 2); x++)

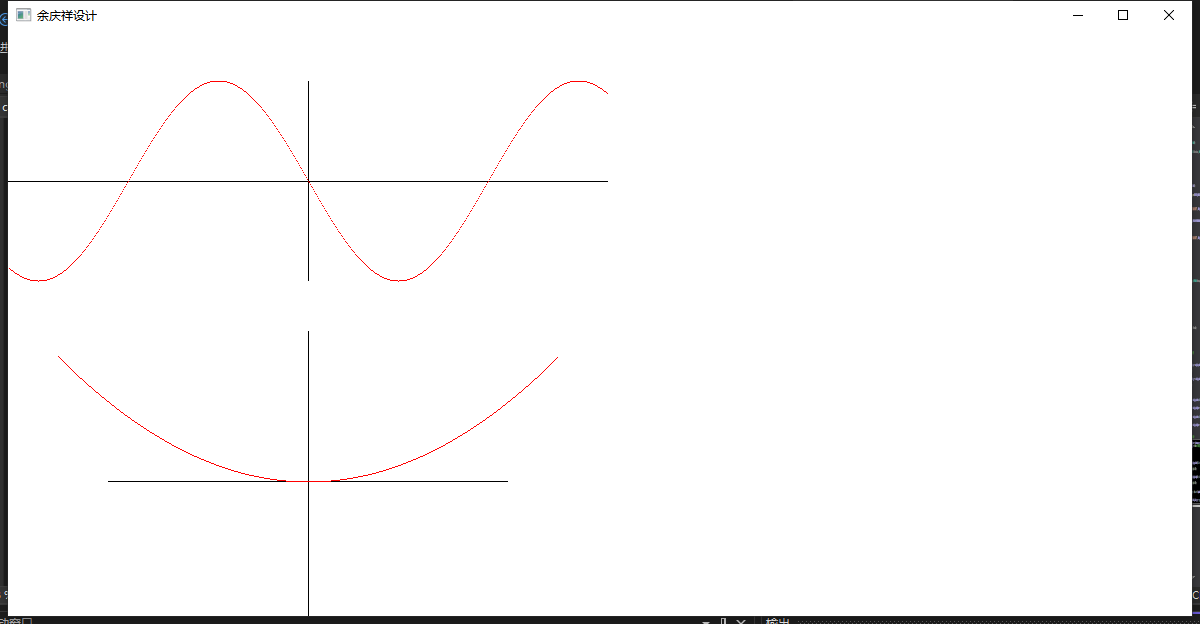
{

y = x \* x / (2 \* p);

SetPixel(hdc, x + parabola\_init\_x, parabola\_init\_y - y, RGB(255, 0, 0));

}

}



1. 如右图效果，绘制文本，程序中可控制字体、居中、换行等效果。

#include <Windows.h>

#include <cmath>

#define windows\_width 900

#define windows\_height 600

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

VOID PAINT(HWND hwnd, HDC hdc);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInst, LPSTR lpszCmdLine, int nCmdShow)

{

HWND hwnd;

MSG Msg;

WNDCLASS wndclass;

TCHAR lpszTitle[] = L"余庆祥设计";

RECT wr = { 0, 0, windows\_width, windows\_height };

AdjustWindowRect(&wr, WS\_OVERLAPPEDWINDOW, FALSE);

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

wndclass.lpszMenuName = NULL;

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

wndclass.lpszClassName = L"xiang";

if (!RegisterClass(&wndclass))

{

MessageBox(NULL, L"注册窗口失败", L"注册窗口失败", MB\_ICONERROR);

return 0;

}

hwnd = CreateWindow(L"xiang", lpszTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT, wr.right - wr.left, wr.bottom - wr.top,

NULL, NULL, hInstance, NULL);

if (!hwnd)

{

MessageBox(NULL, L"创建窗口失败", L"创建窗口失败", MB\_ICONERROR);

return 0;

}

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

while (GetMessage(&Msg, NULL, 0, 0))

{

TranslateMessage(&Msg);

DispatchMessage(&Msg);

}

return Msg.wParam;

}

HDC hdc;

HBRUSH hOldBrush; // 初始时的笔刷样式

HBRUSH hBrush1 = CreateSolidBrush(RGB(255, 0, 0)); // 笔刷样式为：红色实心笔刷

HFONT hOldFont; // 初始时的文字样式

HFONT hFont1 = CreateFont(30, 15, 0, 0,

1300, 0, 0, 0, 0,

0, 0, 10, 0, L"黑体"); // 文字样式为：30号黑体

static TCHAR str1[] = L"祝我们亲爱的祖国";

static TCHAR str2[] = L"生日快乐！";

LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_PAINT:

hdc = GetDC(hwnd); // 获取设备环境

PAINT(hwnd, hdc); // 绘制

ReleaseDC(hwnd, hdc); // 释放设备环境

break;

case WM\_DESTROY:

DeleteObject(hBrush1); // 删除笔刷样式hBrush1

DeleteObject(hFont1); // 删除文字样式hFont1

SelectObject(hdc, hOldBrush); // 恢复初始时的笔刷样式

SelectObject(hdc, hOldFont); // 恢复初始时的文字样式

PostQuitMessage(0);

return 0;

}

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

}

VOID PAINT(HWND hwnd, HDC hdc)

{

hOldBrush = (HBRUSH)SelectObject(hdc, hBrush1);

hOldFont = (HFONT)SelectObject(hdc, hFont1);

Rectangle(hdc, 0, 0, windows\_width, windows\_height); // 绘制整个客户区的尺寸的矩形

SetBkMode(hdc, TRANSPARENT); // 设置文字背景为透明模式

SetTextColor(hdc, RGB(255, 255, 0)); // 设置字体颜色

SetGraphicsMode(hdc, GM\_ADVANCED); // 设置允许世界转换的高级图形模式。

TEXTMETRIC tm;

GetTextMetrics(hdc, &tm); // 使用当前所选字体的指标填充指定的缓冲区

int text\_spacing = tm.tmExternalLeading + tm.tmHeight; // 两行字体之间的间距

// 分别移动间距使上下两行不拥挤

RECT rect1 = { 0, 0, windows\_width, windows\_height - text\_spacing };

RECT rect2 = { 0, 0, windows\_width, windows\_height + text\_spacing };

// 两组文字分别根据位置信息，并垂直居中文本、使文本对齐到矩形底部、在矩形中水平居中居中文本

DrawText(hdc, str1, -1, &rect1, DT\_VCENTER | DT\_SINGLELINE | DT\_CENTER);

DrawText(hdc, str2, -1, &rect2, DT\_VCENTER | DT\_SINGLELINE | DT\_CENTER);

}

