#include "MyDirectX.h"

const string APPTITLE = "Tank";

const int SCREENWIDTH = 640;

const int SCREENHEIGHT = 480;

LPDIRECT3DSURFACE9 back\_surf = NULL;

LPDIRECT3DTEXTURE9 tank\_img;

LPD3DXSPRITE sprite\_obj;

//bullet sprite

LPDIRECT3DTEXTURE9 bullet\_image;

int start = 0;

const int cellwidth = 24;

const int cellheight = 24;

const int BULLET\_NUMBERS= 100;

//explode sprite

LPDIRECT3DTEXTURE9 explode\_image;

//stone sprite

LPDIRECT3DTEXTURE9 stone\_image;

CSound \*sound\_explode = NULL;

CSound \*sound\_fire = NULL;

const int cellrows = SCREENHEIGHT / cellheight;

const int cellcolumns = SCREENWIDTH / cellwidth;

int cell[cellrows][cellcolumns];

int last\_bullet = -1;

int valid\_bullet[BULLET\_NUMBERS];

int last\_explode = -1;

int valid\_explode[BULLET\_NUMBERS];

int start\_bullet;

struct SPRITE // define structure SPRITE at the beginning of MyGame.cpp

{

int x, y, movex, movey;

int frame, columns;

int width, height;

int startframe, endframe;

int starttime, delay;

int direction;

SPRITE()

{

x = y = 0; movex = movey = 0;

frame = 0; columns = 1; width = height = 0;

startframe = endframe = 0; starttime = delay = 0;

direction = 1;

}

};

SPRITE tank;

SPRITE stonecell[cellrows][cellcolumns];

SPRITE bullet[BULLET\_NUMBERS];

SPRITE explode[BULLET\_NUMBERS];

int Collision(SPRITE sprite1, SPRITE sprite2)

{

RECT rect1;

rect1.left = sprite1.x + 1;

rect1.top = sprite1.y + 1;

rect1.right = sprite1.x + sprite1.width - 1;

rect1.bottom = sprite1.y + sprite1.height - 1;

RECT rect2;

rect2.left = sprite2.x + 1;

rect2.top = sprite2.y + 1;

rect2.right = sprite2.x + sprite2.width - 1;

rect2.bottom = sprite2.y + sprite2.height - 1;

RECT dest;

return IntersectRect(&dest, &rect1, &rect2); // if rect1 and rect2 has intersection

} // return true

bool LoadStones()

{

D3DXIMAGE\_INFO info;

HRESULT result;

//load the stone image

stone\_image = LoadTexture("stone.bmp", D3DCOLOR\_XRGB(255, 0, 255));

if (stone\_image == NULL)

return false;

//set the stone's properties

for (int i = 0; i< cellrows; i++)

for (int j = 0; j< cellcolumns; j++)

{

cell[i][j] = 0;

stonecell[i][j].width = cellwidth;

stonecell[i][j].height = cellheight;

stonecell[i][j].x = j \* cellwidth;

stonecell[i][j].y = i \* cellheight;

stonecell[i][j].movex = 0;

stonecell[i][j].movey = 0;

}

// randomly build 100 stone cells

for (int i = 0; i<100; i++)

cell[rand() % cellrows][rand() % cellcolumns] = 1;

return true;

}

bool LoadTank()

{

D3DXIMAGE\_INFO info;

HRESULT result;

//load the tank sprite

tank\_img = LoadTexture("tank.bmp", D3DCOLOR\_XRGB(255, 0, 255));

if (tank\_img == NULL)

return false;

//set the tank's properties

result = D3DXGetImageInfoFromFile("tank.bmp", &info);

if (result != D3D\_OK)

return false;

tank.width = info.Width;

tank.height = info.Height / 4;

tank.x = (SCREENWIDTH - tank.width) / 2;

tank.y = SCREENHEIGHT - tank.height;

tank.movex = 0; tank.movey = -1;

tank.endframe = 0;

return true;

}

void UpdateTank()

{

//check for right arrow

if (Key\_Down(DIK\_RIGHT))

{

tank.movex = 1;

tank.movey = 0;

tank.frame = 1;

tank.x += tank.movex; // only move when key pressed

tank.y += tank.movey;

}

//check for up arrow

else if (Key\_Down(DIK\_UP))

{

tank.movex = 0;

tank.movey = -1;

tank.frame = 0;

tank.x += tank.movex;

tank.y += tank.movey;

}

//check for down arrow

else if (Key\_Down(DIK\_DOWN))

{

tank.movex = 0;

tank.movey = 1;

tank.frame = 2;

tank.x += tank.movex;

tank.y += tank.movey;

}

//check for left arrow

else if (Key\_Down(DIK\_LEFT))

{

tank.movex = -1;

tank.movey = 0;

tank.frame = 3;

tank.x += tank.movex;

tank.y += tank.movey;

}

if (tank.x < 0)

tank.x = 0;

if (tank.x > SCREENWIDTH - tank.width)

tank.x = SCREENWIDTH - tank.width;

if (tank.y < 0)

tank.y = 0;

if (tank.y > SCREENHEIGHT - tank.height)

tank.y = SCREENHEIGHT - tank.height;

for (int i = 0; i< cellrows; i++)

for (int j = 0; j< cellcolumns; j++)

{

// if hit stone, tank backward to its previous position

if (cell[i][j] == 1 && Collision(tank, stonecell[i][j]))

{

tank.x -= tank.movex;

tank.y -= tank.movey;

}

}

}

void UpdateBullets()

{

for (int i = 0; i<BULLET\_NUMBERS; i++)

{

// update bullets positions

bullet[i].x += bullet[i].movex;

bullet[i].y += bullet[i].movey;

// if bullet move out of screen, that bullet turns to be invalid

if (bullet[i].x < 0 || bullet[i].x > SCREENWIDTH - bullet[i].width)

valid\_bullet[i] = 0;

if (bullet[i].y < 0 || bullet[i].y > SCREENHEIGHT - bullet[i].height)

valid\_bullet[i] = 0;

}

if (Key\_Down(DIK\_SPACE) && GetTickCount() - start\_bullet > 500)

{

start\_bullet = GetTickCount();

last\_bullet++; // fire a new bullet

if (last\_bullet >= BULLET\_NUMBERS)

last\_bullet = 0; // use a circular array

valid\_bullet[last\_bullet] = 1;

bullet[last\_bullet].x = tank.x + tank.width / 2;

bullet[last\_bullet].y = tank.y + tank.height / 2;

bullet[last\_bullet].movex = tank.movex \* 5;

bullet[last\_bullet].movey = tank.movey \* 5;

bullet[last\_bullet].x += tank.movex \* tank.width;

bullet[last\_bullet].y += tank.movey \* tank.height;

PlaySound(sound\_fire);

}

for (int i = 0; i<cellrows; i++)

for (int j = 0; j<cellcolumns; j++)

for (int k = 0; k<BULLET\_NUMBERS; k++)

{

if (cell[i][j] == 1 && valid\_bullet[k] &&

Collision(stonecell[i][j], bullet[k]))

{

// bullet hits stone and both turns invalid

cell[i][j] = 0;

valid\_bullet[k] = 0;

last\_explode++;

if (last\_explode >= BULLET\_NUMBERS)

last\_explode = 0; // use a circular array

valid\_explode[last\_explode] = 1; explode[last\_explode].x = stonecell[i][j].x;

explode[last\_explode].y = stonecell[i][j].y;

PlaySound(sound\_explode);

}

}

}

void UpdateExplosions()

{

for (int i = 0; i<BULLET\_NUMBERS; i++)

{

// update explision frames

if (valid\_explode[i])

{

explode[i].frame++;

if (explode[i].frame > explode[i].endframe)

valid\_explode[i] = 0;

}

}

}

bool LoadBullets()

{

D3DXIMAGE\_INFO info;

HRESULT result;

//load the bullet sprite

bullet\_image = LoadTexture("bullet.bmp",

D3DCOLOR\_XRGB(255, 0, 255));

if (bullet\_image == NULL) return false;

//set the bullet's properties

result = D3DXGetImageInfoFromFile("bullet.bmp", &info);

if (result != D3D\_OK) return false;;

for (int i = 0; i<BULLET\_NUMBERS; i++)

{

bullet[i].width = info.Width;

bullet[i].height = info.Height;

}

// initialize all bullet status to be invalid

for (int i = 0; i<BULLET\_NUMBERS; i++)

valid\_bullet[i] = 0;

return true;

}

bool LoadExplosions()

{

D3DXIMAGE\_INFO info;

HRESULT result;

//load the explode sprite

explode\_image = LoadTexture("explosion2.bmp",

D3DCOLOR\_XRGB(255, 0, 255));

if (explode\_image == NULL) return false;

//set the explode's properties

result = D3DXGetImageInfoFromFile("explosion2.bmp", &info);

if (result != D3D\_OK) return false;

for (int i = 0; i<BULLET\_NUMBERS; i++)

{

explode[i].width = info.Width / 8;

explode[i].height = info.Height;

explode[i].endframe = 7;

explode[i].columns = 8;

}

// initialize all explode status to be invalid

for (int i = 0; i<BULLET\_NUMBERS; i++)

valid\_explode[i] = 0;

return true;

}

bool Game\_Init(HWND hwnd)

{

Direct3D\_Init(hwnd, SCREENWIDTH, SCREENHEIGHT, false);

DirectInput\_Init(hwnd);

LoadTank();

LoadStones();

if (!LoadBullets())

{

MessageBox(hwnd, "Error loading Bullets", "Error", 0);

return false;

}

if (!LoadExplosions())

{

MessageBox(hwnd, "Error loading Explosion", "Error", 0);

return false;

}

DirectSound\_Init(hwnd);

//load explode wave file

sound\_explode = LoadSound("explode.wav");

if (sound\_explode == NULL) return 0;

//load hit wave file

sound\_fire = LoadSound("gunfire.wav");

if (sound\_fire == NULL) return 0;

back\_surf = LoadSurface("background.bmp");

DrawSurface(backbuffer, 0, 0, back\_surf);

return true;

}

void Game\_Run(HWND hwnd)

{

if (!d3ddev) return;

DirectInput\_Update();

if (GetTickCount() - start >= 30)

{

//reset timing

start = GetTickCount();

UpdateTank();

UpdateBullets();

UpdateExplosions();

}

d3ddev->ColorFill(backbuffer, NULL, D3DCOLOR\_XRGB(0, 0, 0));

if (d3ddev->BeginScene())

{

DrawSurface(backbuffer, 0, 0, back\_surf);

sprite\_obj->Begin(D3DXSPRITE\_ALPHABLEND);

Sprite\_Draw\_Frame(tank\_img, tank.x, tank.y, tank.frame,

tank.width, tank.height, tank.columns);

for (int i = 0; i< cellrows; i++)

{

for (int j = 0; j< cellcolumns; j++)

{

if (cell[i][j] == 1)

Sprite\_Draw\_Frame(stone\_image,

stonecell[i][j].x, stonecell[i][j].y,

stonecell[i][j].frame, stonecell[i][j].width,

stonecell[i][j].height, stonecell[i][j].columns);

}

}

for (int i = 0; i < BULLET\_NUMBERS; i++)

{

if (valid\_bullet[i])

Sprite\_Draw\_Frame(bullet\_image, bullet[i].x, bullet[i].y,

bullet[i].frame, bullet[i].width, bullet[i].height,

bullet[i].columns);

}

for (int i = 0; i < BULLET\_NUMBERS; i++)

{

if (valid\_explode[i])

Sprite\_Draw\_Frame(explode\_image, explode[i].x,

explode[i].y, explode[i].frame, explode[i].width,

explode[i].height, explode[i].columns);

}

sprite\_obj->End();

d3ddev->EndScene();

}

d3ddev->Present(NULL, NULL, NULL, NULL);

if (Key\_Down(DIK\_ESCAPE)) //escape key exits

gameover = true;

}

void Game\_End()

{

if (back\_surf)back\_surf->Release();

if (tank\_img)tank\_img->Release();

DirectInput\_Shutdown();

Direct3D\_Shutdown();

}