```
#include <Windows.h>
#include <mmsystem.h>
#include <d3dx9tex.h>
#include <d3dx9.h>
#include <dinput.h>
#include "Camera.h"
#include <DShow.h>

#pragma comment (lib, "d3d9.lib")
#pragma comment (lib, "d3dx9.lib")
#pragma comment (lib, "dinput8.lib")
#pragma comment (lib, "dxguid.lib")
#pragma comment (lib, "quartz.lib")
#pragma comment (lib, "strmiids.lib")
#pragma comment (lib, "winmm.lib" )

#define   FVF_FLAGS         D3DFVF_XYZ | D3DFVF_TEX1  //ce contine custom vertex ul xyz, texture

//Acesta va fi trimis spre fereastra aplicației ori de câte ori interfața de evenimente necesită să fie interogată despre evenimentele din filtru.
#define   WM_GRAPHNOTIFY    WM_APP + 1

LPDIRECT3D9             g_pD3D = NULL;  // Used to create the D3DDevice
LPDIRECT3DDEVICE9       g_pd3dDevice = NULL; // Our rendering device

//Mesh - obiect 3D modelat poligonal
LPD3DXMESH              g_pMesh = NULL;

D3DMATERIAL9* g_pMeshMaterials = NULL; // Materials for our mesh
LPDIRECT3DTEXTURE9* g_pMeshTextures = NULL; //// Textures for our mesh
DWORD g_dwNumMaterials = 0; // // Number of mesh materials

IGraphBuilder* pGraphBuilder = NULL;
//Interfața pentru control media (IMediaControl)- un fel de media player responsabil cu pornirea și oprirea rulării.
IMediaControl* pMediaControl = NULL;
//We receie events in case something happened - during playing, stoping, errors etc..
IMediaEventEx* pMediaEvent = NULL;
////We can use to fast forward, revert etc..
IMediaSeeking* pMediaSeeking = NULL;

D3DXMATRIXA16 worldMatrix;

/// <summary>
/// Ce sunt matricile tipul lor si la ce le folosim?
/// def custoom vertex
/// </summary>
struct CUSTOMVERTEX
{
    float x, y, z; // Pozitia vertexilor
    float tu, tv; // Coordonatele de textura
};
//def coordonatele in spatiu si pe textura
CUSTOMVERTEX skyboxCoordonates[24] =
```

```
{
    { -10.0f, -10.0f,  10.0f,  0.0f, 1.0f },
    { -10.0f,  10.0f,  10.0f,  0.0f, 0.0f },
    { 10.0f, -10.0f,  10.0f,  1.0f, 1.0f },
    { 10.0f,  10.0f,  10.0f,  1.0f, 0.0f },

    { 10.0f, -10.0f, -10.0f,  0.0f, 1.0f },
    { 10.0f,  10.0f, -10.0f,  0.0f, 0.0f },
    { -10.0f, -10.0f, -10.0f,  1.0f, 1.0f },
    { -10.0f,  10.0f, -10.0f,  1.0f, 0.0f },

    { -10.0f, -10.0f, -10.0f,  0.0f, 1.0f },
    { -10.0f,  10.0f, -10.0f,  0.0f, 0.0f },
    { -10.0f, -10.0f,  10.0f,  1.0f, 1.0f },
    { -10.0f,  10.0f,  10.0f,  1.0f, 0.0f },

    { 10.0f, -10.0f,  10.0f,  0.0f, 1.0f },
    { 10.0f,  10.0f,  10.0f,  0.0f, 0.0f },
    { 10.0f, -10.0f, -10.0f,  1.0f, 1.0f },
    { 10.0f,  10.0f, -10.0f,  1.0f, 0.0f },

    { -10.0f,  10.0f,  10.0f,  0.0f, 1.0f },
    { -10.0f,  10.0f, -10.0f,  0.0f, 0.0f },
    { 10.0f,  10.0f,  10.0f,  1.0f, 1.0f },
    { 10.0f,  10.0f, -10.0f,  1.0f, 0.0f },

    { -10.0f, -10.0f, -10.0f,  0.0f, 1.0f },
    { -10.0f, -10.0f,  10.0f,  0.0f, 0.0f },
    { 10.0f, -10.0f, -10.0f,  1.0f, 1.0f },
    { 10.0f, -10.0f,  10.0f,  1.0f, 0.0f }
};

LPDIRECT3DVERTEXBUFFER9     SkyboxVertexBuffer = NULL;
LPDIRECT3DTEXTURE9          Textures[6];

CXCamera* camera;

float miscare_mouse_cam_y = 0.0f;
float miscare_mouse_cam_x = 0.0f;

float Mesh_x = 0; ////pozitie mesh
float Mesh_y = -2.5;
float Mesh_z = 0;

//rotatie mesha
float rotMesh_x = 0;
float rotMesh_y = 0;
float rotMesh_z = 0;

float cam_x = 0;
float cam_y = 0;
float cam_z = -50;

LPDIRECTINPUTDEVICE8    dinmouse; // the pointer to the mouse device
```

```
DIMOUSESTATE            mousestate; //// the storage for the mouse-information
BYTE                    keystate[256]; //// the storage for the key-information
LPDIRECTINPUTDEVICE8    dinkeyboard; //// the pointer to the keyboard device
LPDIRECTINPUT8          din; //// the pointer to our DirectInput interface

HWND hWnd; //Handle al ferestrei ce va fi asociată cu dispozitivul creat. IDirect3DDevice9 va utiliza această fereastră ca o planșă pentru desenare.Această valoare va fi fereastra creată în pasul 1 anterior.
HDC hdc; //?????

HRESULT SkyBox()
{
    HRESULT hRet;

    hRet = g_pd3dDevice->CreateVertexBuffer(sizeof(CUSTOMVERTEX) * 24 //memorie alocata 24 de custom vertex uri
        , 0 //pozitia de unde sa inceapa sa aloce
        , FVF_FLAGS, //cum arata custom vertexul meu din memoria mea - formatul
        D3DPOOL_MANAGED, //sistemului ii zice sa copieze resursele automat
        &SkyboxVertexBuffer, //referinta la vertexBuffer
        NULL); //shared intre mai multe device uri/ferestre

    if (FAILED(hRet))
    {
        ::MessageBox(NULL, "Failed to create the vertex buffer!", "Error in SkyBox()", MB_OK | MB_ICONSTOP);
        return false;
    }

    void* pVertices = NULL;
    //6 fete  *4 pct pe fiecare fata
    SkyboxVertexBuffer->Lock(0, sizeof(CUSTOMVERTEX) * 24, (void**)&pVertices, 0); //blocheaza zona de memeorie pentru restul , nimeni altcineva nu are acces la ea
    memcpy(pVertices, skyboxCoordonates, sizeof(CUSTOMVERTEX) * 24); //in vertex buffer copiez coordonatele definite mai sus
    SkyboxVertexBuffer->Unlock(); //unlock memorie

    hRet = D3DXCreateTextureFromFile(g_pd3dDevice, ("Skybox\\front.jpg"), &Textures[0]);
    hRet = D3DXCreateTextureFromFile(g_pd3dDevice, ("Skybox\\back.jpg"), &Textures[1]);
    hRet = D3DXCreateTextureFromFile(g_pd3dDevice, ("Skybox\\left.jpg"), &Textures[2]);
    hRet = D3DXCreateTextureFromFile(g_pd3dDevice, ("Skybox\\right.jpg"), &Textures[3]);
    hRet = D3DXCreateTextureFromFile(g_pd3dDevice, ("Skybox\\top.jpg"), &Textures[4]);
    hRet = D3DXCreateTextureFromFile(g_pd3dDevice, ("Skybox\\bottom.jpg"), &Textures[5]);

    //daca nu s-a incarcat vreuna se opreste apk
    if (FAILED(hRet))
    {
        ::MessageBox(NULL, "Failed to open 1 or more images files!", "Error Opening Texture Files", MB_OK | MB_ICONSTOP); //Buton e ok si iconita
        return false;
    }
}

//DirectShow poate rula fișiere cu derulare continuă cum ar fi cele video și de sunet
```

```
HRESULT InitDirectShow(HWND hWnd)
{
    //vreau sa il fac pe fereastra asta
    CoInitialize(NULL);
    CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC_SERVER, IID_IGraphBuilder,
(void**)&pGraphBuilder); //creare instanta pt graph builder ( interfata pentru controale media)
    ////Create Media Control and Events
    //dupa ce am facut un filtru gol il pregatim adica obtinem cele doua interfete care fac parte din
filtru
    pGraphBuilder->QueryInterface(IID_IMediaControl, (void**)&pMediaControl); ////ca un media player e
responsabil cu pornirea si oprirea media file
    pGraphBuilder->QueryInterface(IID_IMediaEventEx, (void**)&pMediaEvent); ////pentru a notifica daca
apare vreo eroare sau daca deexemplu media file-ul a luat sfarsit
    pGraphBuilder->QueryInterface(IID_IMediaSeeking, (void**)&pMediaSeeking);///bara de cautare intr-un
fisier media
    //Set window for events  - basically we tell our event in case you raise an event use the following
event id.
    pMediaEvent->SetNotifyWindow((OAHWND)hWnd,
        WM_GRAPHNOTIFY, //innregistrez evenimentele din video in coada de mesaje cu codul VM
GRAPHNOTIFY, DACA VOIAM HANDLE IL FACEAM ACOLO UDNE AM SI DESTORY UL IN MSG
        NULL); //creare eveniment nou
    pGraphBuilder->RenderFile(L"Library.mp3", NULL); //deshide fisierul mp3

    return S_OK;
}

VOID ReplaySound() {
    long evCode;

    pMediaEvent->WaitForCompletion(0, &evCode);
    if (evCode == EC_COMPLETE) { //daca e terminat
        LONGLONG startPos = 0;
        pMediaControl->Stop(); // il opreste
        pMediaSeeking->SetPositions(&startPos, AM_SEEKING_AbsolutePositioning, NULL,
AM_SEEKING_NoPositioning); //seteaza pozitia la inceput
    }
}
//Initializes DirectX : Creates D3D objectand device
HRESULT InitD3D(HWND hWnd)
{
    //// Create the D3D object.
    if (NULL == (g_pD3D = Direct3DCreate9(D3D_SDK_VERSION)))
        return E_FAIL;
    ////structura utilizata pentru a crea deviceul D3DDevice.ea specifica cum va functiona dispozitivu
3d creat
    D3DPRESENT_PARAMETERS d3dpp;
    ZeroMemory(&d3dpp, sizeof(d3dpp));

    d3dpp.Windowed = true; ////e pe true deoarece vrem sa ruleze in fereastra
    d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD; ////descrie cum se comporta bufferul spate cand este
flipped(sa devina noul buffer spate se comuta bufferul spate cu bufferl fata)
    d3dpp.BackBufferFormat = D3DFMT_UNKNOWN; ////specifica culoarea back bufferului,cum sunt aranjate
componentele RGB.Punem D3DFMT_UNKNOWN daca formatul etse necunoscut
    d3dpp.EnableAutoDepthStencil = TRUE; //permite directx sa se ocupe el cu bufferele spate
```

```
197.     d3dpp.AutoDepthStencilFormat = D3DFMT_D16; //// 32-bit z-buffer bit depth using 24 bits for the
    depth channel.nu memoreaza pixeli ci o valoare in el se face o masca.
198. 
199.     //pentru a creea dispozitivul folosim CreateDevice()-o metoda a abiectului direct 3d obtinut inainte
200.     //aceasta metoda cere anumiti parametrii incluzand idul unic al deviceului creat, tipul deviceului
201.     //un handler de fereatra,si niste flaguri de comportament care specifica
202.     //cum trebuie sa opereze deviceul creat. dacac se creeaza cu succes aceasta functie
203.     //retunreaza un pointer valid spre interfata IDirect3DDevice9
204.     //odata creat dispozitivul putem cu ajutorul lui sa afisam in el orice folosind o metoda
205. 
206.     if (FAILED(g_pD3D->CreateDevice(D3DADAPTER_DEFAULT, ////parametru de instrare,reprezinta deviceul
    folosit
207.         D3DDEVTYPE_HAL, hWnd,
208.         D3DCREATE_SOFTWARE_VERTEXPROCESSING, //altereaza focusul ferestrei
209.         &d3dpp, &g_pd3dDevice)))
210.     {
211.         if (FAILED(g_pD3D->CreateDevice(D3DADAPTER_DEFAULT, D3DDEVTYPE_REF, hWnd,
212.             D3DCREATE_SOFTWARE_VERTEXPROCESSING, ////procesarea vertexurilor
213.             &d3dpp, &g_pd3dDevice)))
214. 
215.             return E_FAIL;
216.     }
217. 
218.     camera = new CXCamera(g_pd3dDevice);
219. 
220.     return S_OK;
221. }
222. 
223. HRESULT InitGeometry()
224. { // Array of materials
225.     LPD3DXBUFFER pD3DXMtrlBuffer; //daca mesha are mai multe materiale / texturi le salvam aici
226. 
227.     //*******************************************************************************************
228.     // Functia incarca mesh-ul (geometria) brostei in ultimul parametru. Parametrul pMatBuffer
229.     // contine la sfarsitul apelului un buffer de texturi si materiale, iar parametrul nrMaterialeTurtle
230.     // contine numarul de elemente din acest buffer.
231.     // pMatBuffer va fi de fapt un vector de structuri D3DXMATERIAL. Aceasta structura contine
232.     // un D3DMATERIAL9 (coeficientii de reflexie difuza, speculara, ambientala etc) si un
    LPDIRECT3DTEXTURE9 (textura).
233.     //*******************************************************************************************
234. 
235.     if (FAILED(D3DXLoadMeshFromX("Book.x",
236.         D3DXMESH_SYSTEMMEM,
237.         g_pd3dDevice, ////pointer spre dispozitivul 3D uitilizat pentru crearea meshei
238.         NULL, ////Address to receive a buffer containing polygonal adjacency information
239.         &pD3DXMtrlBuffer, //Address to receive material information
240.         NULL, ///Address to receive a buffer of effect instances
241.         &g_dwNumMaterials, ////Address to receive the number of materials in the buffer
242.         &g_pMesh))) //pointer catre mesa
243.     {
244.         MessageBox(NULL, "Could not find Book.x", "Meshes.exe", MB_OK);
245.         return E_FAIL;
246.     }
247.     // We need to extract the material properties and texture names from the
```

```
    // pD3DXMtrlBuffer
    D3DXMATERIAL* d3dxMaterials = (D3DXMATERIAL*)pD3DXMtrlBuffer->GetBufferPointer();
    g_pMeshMaterials = new D3DMATERIAL9[g_dwNumMaterials];
    g_pMeshTextures = new LPDIRECT3DTEXTURE9[g_dwNumMaterials];

    //pt fiecare material
    for (DWORD i = 0; i < g_dwNumMaterials; i++)
    {
        //// Copy the material
        g_pMeshMaterials[i] = d3dxMaterials[i].MatD3D;

        //  // Set the ambient color for the material (D3DX does not do this)
        g_pMeshMaterials[i].Ambient = g_pMeshMaterials[i].Diffuse;

        //seteaza textura pe null
        g_pMeshTextures[i] = NULL;

        if (d3dxMaterials[i].pTextureFilename != NULL && lstrlen(d3dxMaterials[i].pTextureFilename) > 0)
        {
            // Create the texture
            if (FAILED(D3DXCreateTextureFromFile(g_pd3dDevice,
                d3dxMaterials[i].pTextureFilename,
                &g_pMeshTextures[i])))
            {
                MessageBox(NULL, "Could not find texture map", "Meshes.exe", MB_OK);
            }
        }
    }
    // Done with the material buffer
    pD3DXMtrlBuffer->Release();

    return S_OK;
}

////citeste datele de la dispozitive periferice cum ar fi tastatura,mouse
VOID InitDInput(HINSTANCE hInstance, HWND hWnd)
{
    DirectInput8Create(hInstance,   // the handle to the application
        DIRECTINPUT_VERSION,// the compatible version
        IID_IDirectInput8, // the DirectInput interface version
        (void**)&din,  // the pointer to the interface
        NULL); // COM stuff, so we'll set it to NULL

     // se creaza un dispozitiv pentur tastatura
    din->CreateDevice(GUID_SysKeyboard, // the default keyboard ID being used
        &dinkeyboard, // the pointer to the device interface
        NULL);  // COM stuff, so we'll set it to NULL

    // creaza un dispozitiv pentru mouse
    din->CreateDevice(GUID_SysMouse,  // the default keyboard ID being used
        &dinmouse,  // the pointer to the device interface
        NULL // COM stuff, so we'll set it to NULL
    );

```

```
302.     // set the data format to keyboard format
303.     dinkeyboard->SetDataFormat(&c_dfDIKeyboard);
304.     dinmouse->SetDataFormat(&c_dfDIMouse);
305.     // set the control we will have over the keyboard
306.     dinmouse->SetCooperativeLevel(hWnd, DISCL_EXCLUSIVE | DISCL_FOREGROUND); //exclusive - mouse ul e
    folosita doar de apk mea
307.     dinkeyboard->SetCooperativeLevel(hWnd, DISCL_NONEXCLUSIVE | DISCL_FOREGROUND); // tastatura non
    exclusiv
308. }
309.
310. VOID DetectInput()
311. {
312.     //captureaza inputul
313.     dinmouse->Acquire();
314.     dinkeyboard->Acquire();
315.     //citire tastatura
316.     dinmouse->GetDeviceState(sizeof(DIMOUSESTATE), (LPVOID)&mousestate);
317.
318.     dinkeyboard->GetDeviceState(256, (LPVOID)keystate);
319. }
320.
321. /// <summary>
322. /// te uiti la toata tastatura ce s-a apasat si ce se intampla
323. /// </summary>
324. void ProccesInput()
325. {
326.     //deplase mesa xOy
327.     if (keystate[DIK_UP] & 0x80)
328.     {
329.         Mesh_y += 0.1f;
330.     }
331.
332.     if (keystate[DIK_DOWN] & 0x80)
333.     {
334.         Mesh_y -= 0.1f;
335.     }
336.
337.     if (keystate[DIK_LEFT] & 0x80)
338.     {
339.         Mesh_x -= 0.1f;
340.     }
341.
342.     if (keystate[DIK_RIGHT] & 0x80)
343.     {
344.         Mesh_x += 0.1f;
345.     }
346.
347.     if (keystate[DIK_N] & 0x80)
348.     {
349.         Mesh_z += 0.1f;
350.     }
351.
352.     if (keystate[DIK_M] & 0x80)
353.     {
```

```
354.         Mesh_z -= 0.1f;
355.     }
356.
357.     //rotatie mesha
358.     if (keystate[DIK_1] & 0x80)
359.     {
360.         rotMesh_x += 0.5;
361.     }
362.     if (keystate[DIK_2] & 0x80)
363.     {
364.         rotMesh_x -= 0.5;
365.     }
366.     if (keystate[DIK_3] & 0x80)
367.     {
368.         rotMesh_y += 0.5;
369.     }
370.     if (keystate[DIK_4] & 0x80)
371.     {
372.         rotMesh_y -= 0.5;
373.     }
374.     if (keystate[DIK_5] & 0x80)
375.     {
376.         rotMesh_z += 0.5;
377.     }
378.     if (keystate[DIK_6] & 0x80)
379.     {
380.         rotMesh_z -= 0.5;
381.     }
382.
383.     //miscare camera
384.     if (keystate[DIK_W] & 0x80)
385.     {
386.         cam_y += 1;
387.     }
388.     if (keystate[DIK_S] & 0x80)
389.     {
390.         cam_y -= 1;
391.     }
392.     if (keystate[DIK_D] & 0x80)
393.     {
394.         cam_x += 1;
395.     }
396.     if (keystate[DIK_A] & 0x80)
397.     {
398.         cam_x -= 1;
399.     }
400.     if (keystate[DIK_Q] & 0x80)
401.     {
402.         cam_z += 1;
403.     }
404.     if (keystate[DIK_E] & 0x80)
405.     {
406.         cam_z -= 1;
407.     }
```

```
408.
409.     //Play + stop la sound
410.     if (keystate[DIK_P] & 0x80)
411.     {
412.         pMediaControl->Run();
413.     }
414.     if (keystate[DIK_O] & 0x80)
415.     {
416.         pMediaControl->Stop();
417.     }
418.
419.     //se pune pe coada de mesaje, mesajul de destroy  aka inchizi programul
420.     if (keystate[DIK_ESCAPE] & 0x80)
421.     {
422.         PostQuitMessage(0);
423.     }
424. }
425. // Releases all previously initialized objects
426. VOID Cleanup()
427. {
428.     if (g_pMeshMaterials != NULL)
429.         delete[] g_pMeshMaterials;
430.
431.     if (g_pMeshTextures)
432.     {
433.         for (DWORD i = 0; i < g_dwNumMaterials; i++)
434.         {
435.             if (g_pMeshTextures[i])
436.                 g_pMeshTextures[i]->Release();
437.         }
438.         delete[] g_pMeshTextures;
439.     }
440.     if (g_pMesh != NULL)
441.         g_pMesh->Release();
442.
443.     if (g_pd3dDevice != NULL)
444.         g_pd3dDevice->Release();
445.
446.     if (g_pD3D != NULL)
447.         g_pD3D->Release();
448. }
449.
450. VOID SetupMatrices()
451. {
452.     g_pd3dDevice->SetTransform(D3DTS_WORLD, &worldMatrix);
453.
454.     // For the projection matrix, we set up a perspective transform (which
455.     // transforms geometry from 3D view space to 2D viewport space, with
456.     // a perspective divide making objects smaller in the distance). To build
457.     // a perpsective transform, we need the field of view (1/4 pi is common),
458.     // the aspect ratio, and the near and far clipping planes (which define at
459.     // what distances geometry should be no longer be rendered).
460.
461.     D3DXMATRIXA16 projMatrix;
```

```
462.     D3DXMatrixPerspectiveFovLH(&projMatrix, D3DX_PI / 4, 1.0f, 1.0f, 500.0f);
463.     g_pd3dDevice->SetTransform(D3DTS_PROJECTION, &projMatrix);
464. }
465. 
466. VOID SetupLights()
467. {
468.     //se defineste materialul din care sunt formate toate obiectele
469.     //culoarea alb si opac
470.     D3DMATERIAL9 mtrl;
471.     ZeroMemory(&mtrl, sizeof(D3DMATERIAL9));
472.     mtrl.Diffuse.r = mtrl.Ambient.r = 1.0f;
473.     mtrl.Diffuse.g = mtrl.Ambient.g = 1.0f;
474.     mtrl.Diffuse.b = mtrl.Ambient.b = 1.0f;
475.     mtrl.Diffuse.a = mtrl.Ambient.a = 1.0f;
476.     g_pd3dDevice->SetMaterial(&mtrl);
477. 
478.     //directia luminii
479.     //lumina difuza : punct, directionala, spotlight
480.     D3DXVECTOR3 vecDir = D3DXVECTOR3(1.0f, 1.0f, 1.0f);
481.     D3DLIGHT9 light;
482. 
483.     ZeroMemory(&light, sizeof(D3DLIGHT9)); //aloc memorie
484.     light.Type = D3DLIGHT_POINT; //lumina de tip punct
485.     //culoarea luminii
486.     light.Diffuse.r = 1.0f;
487.     light.Diffuse.g = 1.0f;
488.     light.Diffuse.b = 1.0f;
489.     //pozitia punctului
490.     light.Position.x = 5.0f;
491.     light.Position.y = 9.0f;
492.     light.Position.z = 2.0f;
493. 
494.     //normalizeez ectorul de directie, din intervalul pe care l-am facut eu in intervalul 0,1
495.     D3DXVec3Normalize((D3DXVECTOR3*)&light.Direction, &vecDir);
496.     light.Range = 10000.0f;
497.     //setez lumina
498.     g_pd3dDevice->SetLight(0, &light);
499.     //o aprind
500.     g_pd3dDevice->LightEnable(0, TRUE);
501.     //aplicaia are iluminare
502.     g_pd3dDevice->SetRenderState(D3DRS_LIGHTING, TRUE);
503.     //lumina ambientala la mn e
504.     g_pd3dDevice->SetRenderState(D3DRS_AMBIENT, 0x00202020);
505. }
506. 
507. VOID Render()  //se executa tot timpul afisaza obiectele
508. {
509.     // Clear the backbuffer to a blue color
510.     g_pd3dDevice->Clear(0, NULL, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER,
511.         D3DCOLOR_XRGB(0, 0, 255), 1.0f, 0);
512.     // Begin the scene   ...tot ce vrem sa desenam punem intreb begin scene si end scene
513.     if (SUCCEEDED(g_pd3dDevice->BeginScene())) //buferul spate e pregatiti sa primeasca infomartii
    pentru redarea scenei
514.     {
```

```
        SetupMatrices();
        SetupLights();

        miscare_mouse_cam_y -= mousestate.lY * 0.1f;
        miscare_mouse_cam_x -= mousestate.lX * 0.1f;

        //unde ma aflu
        D3DXVECTOR3 vEyePt(10 * cosf(miscare_mouse_cam_x * D3DX_PI / 180), 10 * cosf(miscare_mouse_cam_y * D3DX_PI / 180) + sinf(miscare_mouse_cam_y * D3DX_PI / 180), 10 * sinf(miscare_mouse_cam_x * D3DX_PI / 180));
        //directia unde ma uit
        D3DXVECTOR3 vLookatPt(0.0f, 0.0f, 1.0f);
        //unde privesc in sus
        D3DXVECTOR3 vUpVec(0.0f, 1.0f, 0.0f);

        //camera pe care setez toate astea
        camera->LookAtPos(&vEyePt, &vLookatPt, &vUpVec);
        camera->SetPosition(cam_x, cam_y, cam_z);
        camera->Update();

        D3DXMATRIX translation_matrix;
        D3DXMATRIX scale_matrix;
        D3DXMATRIX rotation_matrix;

        D3DXMatrixScaling(&scale_matrix, 1.5, 1.5, 1.5);
        D3DXMatrixTranslation(&translation_matrix, Mesh_x, Mesh_y, Mesh_z);
        D3DXMatrixRotationYawPitchRoll(&rotation_matrix, rotMesh_y, rotMesh_x, rotMesh_z);
        worldMatrix = scale_matrix * rotation_matrix * translation_matrix;
        //D3DXMatrixMultiply(&worldMatrix, &translation_matrix, &scale_matrix);
        //seteaza matricea de word
        g_pd3dDevice->SetTransform(D3DTS_WORLD, &worldMatrix);

        //parcurge toate texturile si materialele din mesa si le deseneaza
        for (DWORD i = 0; i < g_dwNumMaterials; i++)
        {
            g_pd3dDevice->SetMaterial(&g_pMeshMaterials[i]);
            g_pd3dDevice->SetTexture(0, g_pMeshTextures[i]);

            g_pMesh->DrawSubset(i);
        }

        //aici desenam skybox ul
        D3DXMATRIXA16 matTranslateskybox;
        D3DXMATRIXA16 matScaleSkybox;
        D3DXMatrixScaling(&matScaleSkybox, 15.0, 15.0, 15.0);
        D3DXMatrixTranslation(&matTranslateskybox, 0, 0, 0);
        D3DXMatrixMultiply(&matTranslateskybox, &matScaleSkybox, &matTranslateskybox);
        g_pd3dDevice->SetTransform(D3DTS_WORLD, &matTranslateskybox);

        //seteaza flexible vertex formatul
        g_pd3dDevice->SetFVF(FVF_FLAGS);
        //sursa de unde sa imi ia vertex-urile
        g_pd3dDevice->SetStreamSource(0, SkyboxVertexBuffer, 0, sizeof(CUSTOMVERTEX));

```

```
        //dezactivez iluminarea pt skybox
        g_pd3dDevice->SetRenderState(D3DRS_LIGHTING, FALSE);

        //deseneaza fiecare fata pe rand si aplica textura
        for (ULONG i = 0; i < 6; ++i)
        {
            g_pd3dDevice->SetTexture(0, Textures[i]);
            g_pd3dDevice->DrawPrimitive(D3DPT_TRIANGLESTRIP, i * 4, 2);
        }
        //activeaza dinou lumina
        g_pd3dDevice->SetRenderState(D3DRS_LIGHTING, TRUE);
        g_pd3dDevice->EndScene();
    }

    g_pd3dDevice->Present(NULL, NULL, NULL, NULL);
}

////procesare de mesaje de ex cand dau pe inchidere aplicatie
LRESULT WINAPI MsgProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    switch (msg)
    {
    case WM_DESTROY:
        Cleanup();
        PostQuitMessage(0);
        return 0;
    }
    //chestiile default minimizeaza fereastra
    return DefWindowProc(hWnd, msg, wParam, lParam);
}

INT WINAPI WinMain(HINSTANCE hInst, HINSTANCE, LPSTR, INT)
{
    //clasa care defineste window-ul
    WNDCLASSEX wc = { sizeof(WNDCLASSEX),
        CS_CLASSDC,
        MsgProc,
        0L,
        0L,
        GetModuleHandle(NULL), NULL, NULL, NULL, NULL,
        "D3D Tutorial", NULL };

    // register the window class
    RegisterClassEx(&wc);

    // Crearea ferestrei aplicatiei pe ea se vor desena toate datele
    //creeaza o fereastra de 1024X768 pixeli
    //in hWnd se pastreaza handlerul ferestrei
    HWND hWnd = CreateWindow("D3D Tutorial",
        "Book Project", //titlu fereastra
        WS_OVERLAPPEDWINDOW,  //stilul ferestrei
        100, //pozitia pe x
        100, // pozitia pe y
        1024, //lungime
```

```
        768, //latime
        GetDesktopWindow(),
        NULL, //we aren't using any menus => null
        wc.hInstance, //application handle
        NULL); //multiple windows

    // Initialize Direct3D creeaza obictul 3d
    if (SUCCEEDED(InitD3D(hWnd)))
    {
        InitDInput(hInst, hWnd);

        if (FAILED(InitDirectShow(hWnd)))
            return 0;

        SkyBox();

        if (SUCCEEDED(InitGeometry()))
        {
            // Show the window una ce view-punctul de vizualizare si una world
            ShowWindow(hWnd, SW_SHOWDEFAULT);
            UpdateWindow(hWnd);

            // this struct holds Windows event messages
            MSG msg;
            ZeroMemory(&msg, sizeof(msg));
            while (msg.message != WM_QUIT) //cand timp nu se inchide aplicatia de catre user
            {
                // is there a message to process?
                if (PeekMessage(&msg, NULL, 0U, 0U, PM_REMOVE))
                {
                    // dispatch the message
                    TranslateMessage(&msg);
                    // send the message to the WindowProc function
                    DispatchMessage(&msg);
                }
                //daca nu apare nici un mesaj
                //face in continuu asta
                else
                {
                    DetectInput(); // update the input data before rendering
                    ReplaySound();
                    ProccesInput();
                    Render();
                }
            }
        }
    }

    UnregisterClass("D3D Tutorial", wc.hInstance);
    return 0;
}
```