

~\Desktop\gravity.cpp

```
1  //-----
2  // File: gravity.cpp
3  //
4  // Copyright (c) 1999 Microsoft Corporation. All rights reserved.
5  //-----
6
7  #include "d3dx.h"
8  #include "resource.h"
9  #include "stdio.h"
10
11 #define NAME_OF_THE_APP "D3DX - Gravity"
12
13 #define RELEASENULL(pObject) if (pObject) {pObject->Release(); pObject = NULL;}
14
15 #define NUM_PLANETS          300
16 #define NUM_PARTICLES        1500
17 #define NUM_MATERIALS        25
18 #define NUM_SPHERES          25 //reused by planets
19 #define FULLSCREEN_WIDTH     640
20 #define FULLSCREEN_HEIGHT    480
21
22 #define Y_OFFSET(distance)    distance*(Random(2.0f)-1.0f)
23                               -1.0 ~ +1.0
24 typedef struct _SUN
25 {
26     D3DXVECTOR4 pos;
27     float distance;
28     ID3DXSimpleShape* pSphere;
29     DWORD dwMaterial;
30 } SUN;
31
32 typedef struct _PLANET
33 {
34     D3DXVECTOR4 pos;
35     float distance;
36     DWORD dwSphere;
37     DWORD dwMaterial;
38     BOOL bAttract;
39 } PLANET;
40
41 typedef struct _PARTICLE
42 {
43     D3DXVECTOR4 pos;
44     float distance;
45     DWORD dwMaterial;
46     BOOL bAttract;
47 } PARTICLE;
48
49 class CGravity
50 {
51 public:
52     CGravity();
53     ~CGravity();
54     void          PauseDrawing();
55     void          RestartDrawing();
56     void          UpdateTime();
```

```
57     void UnInit();
58     HRESULT InitD3DX();
59     HRESULT InitRenderer();
60     HRESULT HandleModeChanges();
61     void DestroySpheres();
62     HRESULT GenerateSpheres();
63     HRESULT Draw();
64     void ApplyGravity( float* pfDistance,
65                       D3DXVECTOR4* pPos,
66                       BOOL* pbAttract);
67
68     BOOL m_bD3DXReady;
69     BOOL m_bActive;
70     BOOL m_bIsFullscreen;
71
72     HWND m_hwndMain;
73     RECT m_rWindowedRect;
74
75     LPDIRECT3DDEVICE7 m_pD3DDev;
76     LPDIRECT3D7 m_pD3D;
77     LPDIRECTDRAW7 m_pDD;
78
79     float m_fViewRot[3];
80     float m_fSunRot[3];
81
82     ID3DXContext* m_pD3DX;
83
84     SUN m_Sun;
85     PLANET m_Planets[NUM_PLANETS];
86     PARTICLE m_Particles[NUM_PARTICLES];
87
88     ID3DXMatrixStack* m_pWorldStack;
89     ID3DXMatrixStack* m_pViewStack;
90
91     D3DLIGHT7 m_LightOnSun;
92     D3DLIGHT7 m_LightFromSun;
93     D3DMATERIAL7 m_SunMaterial;
94     D3DMATERIAL7 m_PlanetMaterials[NUM_MATERIALS];
95     ID3DXSimpleShape* m_Spheres[NUM_SPHERES];
96
97     double m_dAbsoluteTime;
98     double m_dElapsedTime;
99     double m_dPeriod;
100    LARGE_INTEGER m_liLastTime;
101 };
102
103 CGravity* g_pGravity;
104
105 CGravity::CGravity()
106 {
107     m_bD3DXReady = FALSE;
108     m_bIsFullscreen = FALSE;
109     m_pD3DDev = NULL;
110     m_pD3D = NULL;
111     m_pDD = NULL;
112     m_pD3DX = NULL;
113     m_fViewRot[0] = 1.0f;
114     m_fViewRot[1] = -1.0f;
115     m_fViewRot[2] = 0.0f;
```

```

116     m_fSunRot[0]      = 0.0f;
117     m_fSunRot[1]      = 0.0f;
118     m_fSunRot[2]      = 0.0f;
119     m_pWorldStack     = NULL;
120     m_pViewStack      = NULL;
121     m_bActive         = !m_bIsFullscreen;
122     m_liLastTime.QuadPart = 0;
123     m_dAbsoluteTime    = 0;
124     m_dElapsedTime     = 0;
125     m_Sun.pSphere     = NULL;
126
127     for( int i = 0; i < NUM_SPHERES; i++ ) // Planets
128     {
129         m_Spheres[i] = NULL;
130     }
131
132     LARGE_INTEGER liFrequency;
133     if(!QueryPerformanceFrequency(&liFrequency)) Hz
134         liFrequency.QuadPart = 1;
135
136     m_dPeriod = (double)1/liFrequency.QuadPart; C SEC = C / Hz → Hz = C / SEC
137     if(!QueryPerformanceCounter(&m_liLastTime)) SEC = C / Hz
138         m_liLastTime.QuadPart = 0;
139 }
140
141 CGravity::~CGravity()
142 {
143     g_pGravity->UnInit();
144 }
145
146 void InterpretError(HRESULT hr)
147 {
148     char errStr[100];
149     D3DXGetErrorString(hr, 100, errStr );
150     MessageBox(NULL,errStr,"D3DX Error",MB_OK);
151 }
152
153 float Random(float fMax)
154 {
155     return fMax*rand()/RAND_MAX;
156 }
157
158 void CGravity::UpdateTime()
159 {
160     LARGE_INTEGER liCurrTime;
161     if(!QueryPerformanceCounter(&liCurrTime))
162         liCurrTime.QuadPart = m_liLastTime.QuadPart + 1;
163
164     m_dElapsedTime = (double)(liCurrTime.QuadPart - m_liLastTime.QuadPart)*
165                     m_dPeriod;
166
167     m_dAbsoluteTime += m_dElapsedTime;
168     m_liLastTime = liCurrTime;
169 }
170
171
172 void CGravity::PauseDrawing()
173 {
174     g_pGravity->m_bActive = FALSE;

```

```

175     if( m_bIsFullscreen )
176         ShowCursor(TRUE);
177 }
178
179 void CGravity::RestartDrawing()
180 {
181     g_pGravity->m_bActive = TRUE;
182     if( m_bIsFullscreen )
183         ShowCursor(FALSE);
184 }
185
186 //*****
187 // Renderer Initialization Code
188 //*****
189
190 HRESULT CGravity::InitD3DX()
191 {
192     HRESULT hr;
193     DWORD i;
194     char buff[1024];
195
196     if( FAILED( hr = D3DXInitialize() ) )
197         return hr;
198
199
200
201     // D3DX Initialization
202     hr = D3DXCreateContextEx( D3DX_DEFAULT,           // D3DX handle
203                             m_bIsFullscreen ? D3DX_CONTEXT_FULLSCREEN:0, // flags
204                             m_hwndMain,
205                             NULL,                     // focusWnd
206                             D3DX_DEFAULT,             // colorbits
207                             D3DX_DEFAULT,             // alphabits
208                             D3DX_DEFAULT,             // numdepthbits
209                             0,                         // numstencilbits ← stencilは使わん
210                             D3DX_DEFAULT,             // numbackbuffers
211                             m_bIsFullscreen? FULLSCREEN_WIDTH:D3DX_DEFAULT, //
212                             m_bIsFullscreen? FULLSCREEN_HEIGHT:D3DX_DEFAULT, //
213                             D3DX_DEFAULT,             // refresh rate
214                             &m_pD3DX                 // returned D3DX interface
215     );
216     if( FAILED(hr) )
217         return hr;
218
219     m_pD3DDev = m_pD3DX->GetD3DDevice();
220     if( m_pD3DDev == NULL )
221         return E_FAIL;
222
223     m_pD3D = m_pD3DX->GetD3D();
224     if( m_pD3D == NULL )
225         return E_FAIL;
226
227     m_pDD = m_pD3DX->GetDD();
228     if( m_pDD == NULL )
229         return E_FAIL;
230
231     m_bD3DXReady = TRUE;

```

width
height

D3DPP.
Back Buffer Height 640
Width 640

stencilは使わん

```
232     return InitRenderer();
233 }
234
235 // *****
236 // Renderer Initialization Code
237 // *****
238
239 HRESULT CGravity::InitRenderer()
240 {
241     HRESULT hr;
242     int i;
243
244     if( !m_bD3DXReady )
245         return E_FAIL;
246
247     hr = m_pD3DX->SetClearColor(D3DRGBA(0,0,0,0));
248     if( FAILED(hr) )
249         return hr;
250
251     srand(4);
252     hr = m_pD3DDev->SetRenderState( D3DRENDERSTATE_DITHERENABLE, TRUE );
253     if ( FAILED(hr) )
254         return hr;
255
256     hr = m_pD3DDev->SetRenderState( D3DRENDERSTATE_SPECULARENABLE, TRUE );
257     if ( FAILED(hr) )
258         return hr;
259
260     hr = m_pD3DX->Clear(D3DCLEAR_TARGET|D3DCLEAR_ZBUFFER);
261     if ( FAILED(hr) )
262         return hr;
263
264     D3DVALUE dvR,dvG,dvB;
265     dvR = 1.0f;
266     dvG = 0.9f;
267     dvB = 0.6f;
268     memset(&m_LightFromSun,0,sizeof(D3DLIGHT7));
269
270     // Light which illuminates the "planets"
271     m_LightFromSun.dltType = D3DLIGHT_POINT;
272     m_LightFromSun.dvAttenuation0 = 0.5f;
273     m_LightFromSun.dvAttenuation1 = 0.008f;
274     m_LightFromSun.dvAttenuation2 = 0.0f;
275     m_LightFromSun.dcvDiffuse.dvR = dvR;
276     m_LightFromSun.dcvDiffuse.dvG = dvG;
277     m_LightFromSun.dcvDiffuse.dvB = dvB;
278     m_LightFromSun.dcvSpecular.dvR = dvR;
279     m_LightFromSun.dcvSpecular.dvG = dvG;
280     m_LightFromSun.dcvSpecular.dvB = dvB;
281     m_LightFromSun.dvRange = 5000.0f;
282
283     // Light which illuminates the "sun"
284     memcpy(&m_LightOnSun, &m_LightFromSun, sizeof(D3DLIGHT7) );
285     m_LightOnSun.dvAttenuation0 = 0.0f;
286     m_LightOnSun.dvAttenuation1 = 0.003f;
287     m_LightOnSun.dvAttenuation2 = 0.0f;
288     m_LightOnSun.dcvDiffuse.dvR = dvR;
289     m_LightOnSun.dcvDiffuse.dvG = dvG;
290     m_LightOnSun.dcvDiffuse.dvB = dvB;
```

```

291 m_LightOnSun.dcvSpecular.dvR = 1.0f;
292 m_LightOnSun.dcvSpecular.dvG = 1.0f;
293 m_LightOnSun.dcvSpecular.dvB = 1.0f;
294
295 hr = m_pD3DDev->LightEnable( 0, TRUE );
296 if ( FAILED(hr) )
297     return hr;
298
299 memset(&m_SunMaterial, 0, sizeof(D3DMATERIAL7));
300 m_SunMaterial.diffuse.r = dvR;
301 m_SunMaterial.diffuse.g = dvG;
302 m_SunMaterial.diffuse.b = dvB;
303 m_SunMaterial.specular.r = 1.0f;
304 m_SunMaterial.specular.g = 1.0f;
305 m_SunMaterial.specular.b = 1.0f;
306 m_SunMaterial.power = 3.0f;
307
308 for( i = 0; i < NUM_MATERIALS; i++ )
309 {
310     memcpy(&m_PlanetMaterials[i], &m_SunMaterial, sizeof(D3DMATERIAL7));
311     m_PlanetMaterials[i].diffuse.r = Random(1.0f);
312     m_PlanetMaterials[i].diffuse.g = Random(1.0f);
313     m_PlanetMaterials[i].diffuse.b = Random(1.0f);
314     m_PlanetMaterials[i].power = 1.0f;
315 }
316
317 if( FAILED( hr = GenerateSpheres() ) )
318     return hr;
319
320 float fPlanetRad;
321 for( i = 0; i < NUM_PARTICLES; i++ )
322 {
323     // Make the planets.
324     fPlanetRad = (float)Random(3.2f) + 0.7f;
325
326     D3DXMATRIX initRot;
327     D3DXMatrixRotationAxis(&initRot, &D3DXVECTOR3( 0.0f, 1.0f, 0.0f ),
Random(2*D3DX_PI) + 1 );
328     m_Particles[i].distance = Random(400.0f) + 50.0f;
329     D3DXVECTOR4 vUnit( 1.0f, Y_OFFSET(m_Particles[i].distance/5000.0f), 0.0f, 1.0f );
330     D3DXVec4Normalize(&vUnit, &vUnit);
331     D3DXVec4Transform( &vUnit, &vUnit, &initRot );
332     m_Particles[i].pos = vUnit;
333
334     m_Particles[i].dwMaterial = (DWORD)Random((FLOAT) NUM_MATERIALS);
335     m_Particles[i].bAttract = TRUE;
336 }
337
338 // Create Matrix Stack
339 D3DXCreateMatrixStack( 0, &m_pWorldStack );
340
341 // Create Matrix Stack
342 D3DXCreateMatrixStack( 0, &m_pViewStack );
343
344 return S_OK;
345 }
346
347 // *****

```

反射

Light の核数 からの反射

反射 色の反射

反射 の強度

25 個 元々の

左手座標系

← 合、213

使われ ないの？

Y 軸 1 ~ 2π+1 回転

50 ~ 450

$(\frac{1}{100} \sim \frac{1}{10}) \times (-1.0 \sim 1.0)$

```

348 // GenerateSpheres
349 // *****
350
351 HRESULT CGravity::GenerateSpheres()
352 {
353     HRESULT hr;
354
355     float fPlanetRad;
356     for( int i = 0; i < NUM_SPHERES; i++ )
357     {
358         fPlanetRad = (float)Random(3.2f) + 0.7f;
359         hr = D3DXCreateSphere( m_pD3DDev,
360                               fPlanetRad,
361                               (int)max(4, fPlanetRad*2),
362                               (int)max(3, fPlanetRad*3),
363                               1,
364                               &m_Spheres[i] );
365
366         if( FAILED(hr) )
367             return hr;
368     }
369
370     // Make the sun.
371     hr = D3DXCreateSphere(m_pD3DDev, 20.0f, 50, 25, 1, &m_Sun.pSphere);
372     if( FAILED(hr) )
373         return hr;
374
375     for( i = 0; i < NUM_PLANETS; i++ )
376     {
377         // Make the planets.
378         fPlanetRad = (float)Random(3.2f) + 0.7f;
379
380         D3DXMATRIX initRot;
381         D3DXMatrixRotationAxis(&initRot, &D3DXVECTOR3( 0.0f, 1.0f, 0.0f ),
382                                Random(2*D3DX_PI) + 1 );
383
384         m_Planets[i].distance = Random(400.0f) + 50.0f;
385         D3DXVECTOR4 vUnit( 1.0f, Y_OFFSET((float) m_Planets[i].distance/5000.0f), 0.0f,
386                            0.0f );
387         D3DXVec4Normalize(&vUnit, &vUnit);
388         D3DXVec4Transform( &vUnit, &vUnit, &initRot );
389         m_Planets[i].pos = vUnit;
390
391         m_Planets[i].dwSphere = (DWORD)Random((FLOAT) NUM_SPHERES);
392
393         m_Planets[i].dwMaterial = (DWORD)Random((FLOAT) NUM_MATERIALS);
394         m_Planets[i].bAttract = TRUE;
395     }
396
397     return S_OK;
398 }
399
400 void CGravity::DestroySpheres()
401 {
402     for( int i = 0; i < NUM_SPHERES; i++ )
403     {
404         RELEASENULL(m_Spheres[i]);
405     }
406 }

```

Handwritten annotations:

- Planet 2~6cm**: Points to the `fPlanetRad` variable.
- 25個の NUM_SPHERES**: Points to the `NUM_SPHERES` constant.
- NUM_MATERIAL**: Points to the `NUM_MATERIALS` constant.
- (0.7 ~ 3.9)**: Points to the range of `fPlanetRad`.
- PLANETS**: Points to the `NUM_PLANETS` constant.
- 0.7 ~ 3.9**: Points to the range of `fPlanetRad` in the planet loop.
- Y軸 (1 ~ 2π + 1) 回転**: Points to the rotation axis and angle.
- Particles と 同く**: Points to the `Random(400.0f) + 50.0f` expression.
- 25個の NUM_SPHERES**: Points to the `NUM_SPHERES` constant.
- 15個の NUM_MATERIALS**: Points to the `NUM_MATERIALS` constant.

```
405 void CGravity::UnInit()
406 {
407     DestroySpheres();
408     RELEASENULL(m_Sun.pSphere);
409     RELEASENULL(m_pD3DDev);
410     RELEASENULL(m_pD3D);
411     RELEASENULL(m_pDD);
412     RELEASENULL(m_pWorldStack);
413     RELEASENULL(m_pViewStack);
414     RELEASENULL(m_pD3DX);
415     m_bD3DXReady = FALSE;
416     D3DXUninitialize();
417 }
418
419 void CGravity::ApplyGravity(float* pfDistance, D3DXVECTOR4* pPos, BOOL* pbAttract)
420 {
421     BOOL bFlippedState = FALSE;
422     if( *pbAttract == TRUE )
423     {
424         // Do some fake gravity stuff... (a little too much gravity... :) )
425         if( *pfDistance > 1 )
426         {
427             // Get sucked towards the sun:
428             *pfDistance *= (float)pow(0.999,m_dElapsed*10);
429
430             if( *pfDistance < 50 )
431             {
432                 *pfDistance -= (float)(m_dElapsed*15);
433             }
434             else if( *pfDistance < 100 )
435             {
436                 *pfDistance -= (float)(m_dElapsed*10);
437             }
438             else if( *pfDistance < 150 )
439             {
440                 *pfDistance -= (float)(m_dElapsed*5);
441             }
442             *pfDistance = max(0.1f,*pfDistance);
443
444             D3DXMATRIX rot;
445             D3DXMatrixRotationAxis(&rot, &D3DXVECTOR3( 0.0f, 1.0f, 0.0f ),
446                                     (float)(80.0f*pow(*pfDistance,-1.1f)
447                                     *m_dElapsed));
448             D3DXVec4Transform( pPos, pPos, &rot );
449             (*pPos).y *= (float)pow(0.999,m_dElapsed*30);
450         }
451         if( *pfDistance <= 1 )
452         {
453             // Teleport out of the sun.
454             *pbAttract = FALSE;
455             bFlippedState = TRUE;
456         }
457         if( *pbAttract == FALSE )
458         {
459             if( *pfDistance > 500 )
460             {
461                 // move to a new far away location, and
462                 // start getting sucked in again
```

あとまわし

Distanceが
小さければ
50以下に
近づける

31力が強い

500より大きければ


```

463         *pfDistance = Random(100.0f) + 500.0f;    500 ~ 600 の位置
464         (*pPos).y = Y_OFFSET(*pfDistance/10000.0f);
465         D3DXVec4Normalize(pPos,pPos);
466         *pbAttract = TRUE;
467         return;
468     }
469     if( bFlippedState )
470     {
471         (*pPos).y = 20.0f;
472         D3DXVec4Normalize(pPos,pPos);
473         if( Random(1.0f) > 0.5f )
474         {
475             (*pPos).y = -(*pPos).y;
476         }
477         *pfDistance = 10.0f;
478     }
479     *pfDistance *= (float) pow(2.0, m_dElapsedTime);
480 }
481 }

```

```

482 // ***** Apply Gravity *****
483 // Rendering Code
484 // *****
485
486 HRESULT CGravity::Draw()
487 {
488     HRESULT hr;
489     int i;
490
491     if( !m_bD3DXReady )
492     {
493         return E_FAIL;
494     }
495     if( !m_bActive )
496     {
497         return S_OK;
498     }
499
500     hr = m_pD3DDev->BeginScene();
501     if ( SUCCEEDED(hr) )
502     {
503         hr = m_pD3DX->Clear(D3DCLEAR_TARGET|D3DCLEAR_ZBUFFER);
504         if ( FAILED(hr) )
505             return hr;
506
507         UpdateTime();
508
509         float fViewDist = 400 + 300 * (float) sin(m_dAbsoluteTime*0.2);    300 ~ 700
510
511         m_LightOnSun.dvPosition.dvX = 0.0f;
512         m_LightOnSun.dvPosition.dvY = 0.0f;
513         m_LightOnSun.dvPosition.dvZ = -100.0f + fViewDist;    0 ~ 600
514         m_pD3DDev->SetLight( 0, &m_LightOnSun );
515
516         // Set up state for drawing the sun.
517
518         m_fSunRot[0] = (float)(m_dAbsoluteTime/2);
519         m_fSunRot[1] = (float)(m_dAbsoluteTime/2);
520         m_fSunRot[2] = (float)(m_dAbsoluteTime);

```

```

522
523 m_pD3DDevice->SetMaterial(&m_SunMaterial);
524
525 D3DXMATRIX matSunWorld, matTemp;
526 D3DXMatrixRotationAxis(&matSunWorld, &D3DXVECTOR3( 1.0f, 0.0f, 0.0f ),
527 m_fSunRot[0] );
528 D3DXMatrixRotationAxis(&matTemp, &D3DXVECTOR3( 0.0f, 1.0f, 0.0f ),
529 m_fSunRot[1] );
530 D3DXMatrixMultiply(&matSunWorld,&matSunWorld,&matTemp);
531 D3DXMatrixRotationAxis(&matTemp, &D3DXVECTOR3( 0.0f, 0.0f, 1.0f ),
532 m_fSunRot[2] );
533 D3DXMatrixMultiply(&matSunWorld,&matSunWorld,&matTemp);
534 D3DXMatrixTranslation(&matTemp,0.0f,0.0f,fViewDist);
535 D3DXMatrixMultiply(&matSunWorld,&matSunWorld,&matTemp);
536
537 m_pD3DDevice->SetTransform( D3DTRANSFORMSTATE_WORLD,
538 (D3DMATRIX *)matSunWorld );

```

```

540 D3DXMATRIX matSunView;
541 D3DXMatrixIdentity(&matSunView);
542 m_pD3DDevice->SetTransform( D3DTRANSFORMSTATE_VIEW,
543 (D3DMATRIX *)matSunView );

```

```

544
545 m_Sun.pSphere->Draw();

```

```

547 m_pViewStack->LoadIdentity();
548 m_fViewRot[0] -= (float)(0.075*m_dElapsedTime);
549 m_fViewRot[1] += (float)(0.01*m_dElapsedTime);
550 m_fViewRot[2] += (float)(0.015*m_dElapsedTime);
551
552 m_pViewStack->RotateAxis( &D3DXVECTOR3( 1.0f, 0.0f, 0.0f ),
553 m_fViewRot[0] );
554 m_pViewStack->RotateAxis( &D3DXVECTOR3( 0.0f, 1.0f, 0.0f ),
555 m_fViewRot[1] );
556 m_pViewStack->RotateAxis( &D3DXVECTOR3( 0.0f, 0.0f, 1.0f ),
557 m_fViewRot[2] );
558 m_pViewStack->Translate(0.0f,0.0f,fViewDist);
559 m_pD3DDevice->SetTransform( D3DTRANSFORMSTATE_VIEW,
560 (D3DMATRIX*)m_pViewStack->GetTop() );
561
562
563 // Set up state for drawing the planets
564
565 m_pD3DDevice->SetLight( 0, &m_LightFromSun );
566
567 // Draw the planets
568 for( i = 0; i < NUM_PLANETS; i++ )
569 {
570 m_pD3DDevice->SetMaterial(&m_PlanetMaterials[m_Planets[i].dwMaterial]);
571 m_pWorldStack->LoadIdentity();
572
573 // Do some fake gravity stuff... (a little too much gravity... :) )
574 ApplyGravity(&m_Planets[i].distance,&m_Planets[i].pos, &m_Planets[i]
575 .bAttract);
576 m_pWorldStack->Translate( m_Planets[i].pos.x * m_Planets[i].distance,
577 m_Planets[i].pos.y * m_Planets[i].distance,
578 m_Planets[i].pos.z * m_Planets[i].distance );
579
580 m_pD3DDevice->SetTransform( D3DTRANSFORMSTATE_WORLD,

```

VIEW(1)

Sun

2つと 300~500 程度

VIEW(2)

1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100. 101. 102. 103. 104. 105. 106. 107. 108. 109. 110. 111. 112. 113. 114. 115. 116. 117. 118. 119. 120. 121. 122. 123. 124. 125. 126. 127. 128. 129. 130. 131. 132. 133. 134. 135. 136. 137. 138. 139. 140. 141. 142. 143. 144. 145. 146. 147. 148. 149. 150. 151. 152. 153. 154. 155. 156. 157. 158. 159. 160. 161. 162. 163. 164. 165. 166. 167. 168. 169. 170. 171. 172. 173. 174. 175. 176. 177. 178. 179. 180. 181. 182. 183. 184. 185. 186. 187. 188. 189. 190. 191. 192. 193. 194. 195. 196. 197. 198. 199. 200. 201. 202. 203. 204. 205. 206. 207. 208. 209. 210. 211. 212. 213. 214. 215. 216. 217. 218. 219. 220. 221. 222. 223. 224. 225. 226. 227. 228. 229. 230. 231. 232. 233. 234. 235. 236. 237. 238. 239. 240. 241. 242. 243. 244. 245. 246. 247. 248. 249. 250. 251. 252. 253. 254. 255. 256. 257. 258. 259. 260. 261. 262. 263. 264. 265. 266. 267. 268. 269. 270. 271. 272. 273. 274. 275. 276. 277. 278. 279. 280. 281. 282. 283. 284. 285. 286. 287. 288. 289. 290. 291. 292. 293. 294. 295. 296. 297. 298. 299. 300. 301. 302. 303. 304. 305. 306. 307. 308. 309. 310. 311. 312. 313. 314. 315. 316. 317. 318. 319. 320. 321. 322. 323. 324. 325. 326. 327. 328. 329. 330. 331. 332. 333. 334. 335. 336. 337. 338. 339. 340. 341. 342. 343. 344. 345. 346. 347. 348. 349. 350. 351. 352. 353. 354. 355. 356. 357. 358. 359. 360. 361. 362. 363. 364. 365. 366. 367. 368. 369. 370. 371. 372. 373. 374. 375. 376. 377. 378. 379. 380. 381. 382. 383. 384. 385. 386. 387. 388. 389. 390. 391. 392. 393. 394. 395. 396. 397. 398. 399. 400. 401. 402. 403. 404. 405. 406. 407. 408. 409. 410. 411. 412. 413. 414. 415. 416. 417. 418. 419. 420. 421. 422. 423. 424. 425. 426. 427. 428. 429. 430. 431. 432. 433. 434. 435. 436. 437. 438. 439. 440. 441. 442. 443. 444. 445. 446. 447. 448. 449. 450. 451. 452. 453. 454. 455. 456. 457. 458. 459. 460. 461. 462. 463. 464. 465. 466. 467. 468. 469. 470. 471. 472. 473. 474. 475. 476. 477. 478. 479. 480. 481. 482. 483. 484. 485. 486. 487. 488. 489. 490. 491. 492. 493. 494. 495. 496. 497. 498. 499. 500. 501. 502. 503. 504. 505. 506. 507. 508. 509. 510. 511. 512. 513. 514. 515. 516. 517. 518. 519. 520. 521. 522. 523. 524. 525. 526. 527. 528. 529. 530. 531. 532. 533. 534. 535. 536. 537. 538. 539. 540. 541. 542. 543. 544. 545. 546. 547. 548. 549. 550. 551. 552. 553. 554. 555. 556. 557. 558. 559. 560. 561. 562. 563. 564. 565. 566. 567. 568. 569. 570. 571. 572. 573. 574. 575. 576. 577. 578. 579. 580. 581. 582. 583. 584. 585. 586. 587. 588. 589. 590. 591. 592. 593. 594. 595. 596. 597. 598. 599. 600. 601. 602. 603. 604. 605. 606. 607. 608. 609. 610. 611. 612. 613. 614. 615. 616. 617. 618. 619. 620. 621. 622. 623. 624. 625. 626. 627. 628. 629. 630. 631. 632. 633. 634. 635. 636. 637. 638. 639. 640. 641. 642. 643. 644. 645. 646. 647. 648. 649. 650. 651. 652. 653. 654. 655. 656. 657. 658. 659. 660. 661. 662. 663. 664. 665. 666. 667. 668. 669. 670. 671. 672. 673. 674. 675. 676. 677. 678. 679. 680. 681. 682. 683. 684. 685. 686. 687. 688. 689. 690. 691. 692. 693. 694. 695. 696. 697. 698. 699. 700. 701. 702. 703. 704. 705. 706. 707. 708. 709. 710. 711. 712. 713. 714. 715. 716. 717. 718. 719. 720. 721. 722. 723. 724. 725. 726. 727. 728. 729. 730. 731. 732. 733. 734. 735. 736. 737. 738. 739. 740. 741. 742. 743. 744. 745. 746. 747. 748. 749. 750. 751. 752. 753. 754. 755. 756. 757. 758. 759. 760. 761. 762. 763. 764. 765. 766. 767. 768. 769. 770. 771. 772. 773. 774. 775. 776. 777. 778. 779. 780. 781. 782. 783. 784. 785. 786. 787. 788. 789. 790. 791. 792. 793. 794. 795. 796. 797. 798. 799. 800. 801. 802. 803. 804. 805. 806. 807. 808. 809. 810. 811. 812. 813. 814. 815. 816. 817. 818. 819. 820. 821. 822. 823. 824. 825. 826. 827. 828. 829. 830. 831. 832. 833. 834. 835. 836. 837. 838. 839. 840. 841. 842. 843. 844. 845. 846. 847. 848. 849. 850. 851. 852. 853. 854. 855. 856. 857. 858. 859. 860. 861. 862. 863. 864. 865. 866. 867. 868. 869. 870. 871. 872. 873. 874. 875. 876. 877. 878. 879. 880. 881. 882. 883. 884. 885. 886. 887. 888. 889. 890. 891. 892. 893. 894. 895. 896. 897. 898. 899. 900. 901. 902. 903. 904. 905. 906. 907. 908. 909. 910. 911. 912. 913. 914. 915. 916. 917. 918. 919. 920. 921. 922. 923. 924. 925. 926. 927. 928. 929. 930. 931. 932. 933. 934. 935. 936. 937. 938. 939. 940. 941. 942. 943. 944. 945. 946. 947. 948. 949. 950. 951. 952. 953. 954. 955. 956. 957. 958. 959. 960. 961. 962. 963. 964. 965. 966. 967. 968. 969. 970. 971. 972. 973. 974. 975. 976. 977. 978. 979. 980. 981. 982. 983. 984. 985. 986. 987. 988. 989. 990. 991. 992. 993. 994. 995. 996. 997. 998. 999. 1000.

2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100. 101. 102. 103. 104. 105. 106. 107. 108. 109. 110. 111. 112. 113. 114. 115. 116. 117. 118. 119. 120. 121. 122. 123. 124. 125. 126. 127. 128. 129. 130. 131. 132. 133. 134. 135. 136. 137. 138. 139. 140. 141. 142. 143. 144. 145. 146. 147. 148. 149. 150. 151. 152. 153. 154. 155. 156. 157. 158. 159. 160. 161. 162. 163. 164. 165. 166. 167. 168. 169. 170. 171. 172. 173. 174. 175. 176. 177. 178. 179. 180. 181. 182. 183. 184. 185. 186. 187. 188. 189. 190. 191. 192. 193. 194. 195. 196. 197. 198. 199. 200. 201. 202. 203. 204. 205. 206. 207. 208. 209. 210. 211. 212. 213. 214. 215. 216. 217. 218. 219. 220. 221. 222. 223. 224. 225. 226. 227. 228. 229. 230. 231. 232. 233. 234. 235. 236. 237. 238. 239. 240. 241. 242. 243. 244. 245. 246. 247. 248. 249. 250. 251. 252. 253. 254. 255. 256. 257. 258. 259. 260. 261. 262. 263. 264. 265. 266. 267. 268. 269. 270. 271. 272. 273. 274. 275. 276. 277. 278. 279. 280. 281. 282. 283. 284. 285. 286. 287. 288. 289. 290. 291. 292. 293. 294. 295. 296. 297. 298. 299. 300. 301. 302. 303. 304. 305. 306. 307. 308. 309. 310. 311. 312. 313. 314. 315. 316. 317. 318. 319. 320. 321. 322. 323. 324. 325. 326. 327. 328. 329. 330. 331. 332. 333. 334. 335. 336. 337. 338. 339. 340. 341. 342. 343. 344. 345. 346. 347. 348. 349. 350. 351. 352. 353. 354. 355. 356. 357. 358. 359. 360. 361. 362. 363. 364. 365. 366. 367. 368. 369. 370. 371. 372. 373. 374. 375. 376. 377. 378. 379. 380. 381. 382. 383. 384. 385. 386. 387. 388. 389. 390. 391. 392. 393. 394. 395. 396. 397. 398. 399. 400. 401. 402. 403. 404. 405. 406. 407. 408. 409. 410. 411. 412. 413. 414. 415. 416. 417. 418. 419. 420. 421. 422. 423. 424. 425. 426. 427. 428. 429. 430. 431. 432. 433. 434. 435. 436. 437. 438. 439. 440. 441. 442. 443. 444. 445. 446. 447. 448. 449. 450. 451. 452. 453. 454. 455. 456. 457. 458. 459. 460. 461. 462. 463. 464. 465. 466. 467. 468. 469. 470. 471. 472. 473. 474. 475. 476. 477. 478. 479. 480. 481. 482. 483. 484. 485. 486. 487. 488. 489. 490. 491. 492. 493. 494. 495. 496. 497. 498. 499. 500. 501. 502. 503. 504. 505. 506. 507. 508. 509. 510. 511. 512. 513. 514. 515. 516. 517. 518. 519. 520. 521. 522. 523. 524. 525. 526. 527. 528. 529. 530. 531. 532. 533. 534. 535. 536. 537. 538. 539. 540. 541. 542. 543. 544. 545. 546. 547. 548. 549. 550. 551. 552. 553. 554. 555. 556. 557. 558. 559. 560. 561. 562. 563. 564. 565. 566. 567. 568. 569. 570. 571. 572. 573. 574. 575. 576. 577. 578. 579. 580. 581. 582. 583. 584. 585. 586. 587. 588. 589. 590. 591. 592. 593. 594. 595. 596. 597. 598. 599. 600. 601. 602. 603. 604. 605. 606. 607. 608. 609. 610. 611. 612. 613. 614. 615. 616. 617. 618. 619. 620. 621. 622. 623. 624. 625. 626. 627. 628. 629. 630. 631. 632. 633. 634. 635. 636. 637. 638. 639. 640. 641. 642. 643. 644. 645. 646. 647. 648. 649. 650. 651. 652. 653. 654. 655. 656. 657. 658. 659. 660. 661. 662. 663. 664. 665. 666. 667. 668. 669. 670. 671. 672. 673. 674. 675. 676. 677. 678. 679. 680. 681. 682. 683. 684. 685. 686. 687. 688. 689. 690. 691. 692. 693. 694. 695. 696. 697. 698. 699. 700. 701. 702. 703. 704. 705. 706. 707. 708. 709. 710. 711. 712. 713. 714. 715. 716. 717. 718. 719. 720. 721. 722. 723. 724. 725. 726. 727. 728. 729. 730. 731. 732. 733. 734. 735. 736. 737. 738. 739. 740. 741. 742. 743. 744. 745. 746. 747. 748. 749. 750. 751. 752. 753. 754. 755. 756. 757. 758. 759. 760. 761. 762. 763. 764. 765. 766. 767. 768. 769. 770. 771. 772. 773. 774. 775. 776. 777. 778. 779. 780. 781. 782. 783. 784. 785. 786. 787. 788. 789. 790. 791. 792. 793. 794. 795. 796. 797. 798. 799. 800. 801. 802. 803. 804. 805. 806. 807. 808. 809. 810. 811. 812. 813. 814. 815. 816. 817. 818. 819. 820. 821. 822. 823. 824. 825. 826. 827. 828. 829. 830. 831. 832. 833. 834. 835. 836. 837. 838. 839. 840. 841. 842. 843. 844. 845. 846. 847. 848. 849. 850. 851. 852. 853. 854. 855. 856. 857. 858. 859. 860. 861. 862. 863. 864. 865. 866. 867. 868. 869. 870. 871. 872. 873. 874. 875. 876. 877. 878. 879. 880. 881. 882. 883. 884. 885. 886. 887. 888. 889. 890. 891. 892. 893. 894. 895. 896. 897. 898. 899. 900. 901. 902. 903. 904. 905. 906. 907. 908. 909. 910. 911. 912. 913. 914. 915. 916. 917. 918. 919. 920. 921. 922. 923. 924. 925. 926. 927. 928. 929. 930. 931. 932. 933. 934. 935. 936. 937. 938. 939. 940. 941. 942. 943. 944. 945. 946. 947. 948. 949. 950. 951. 952. 953. 954. 955. 956. 957. 958. 959. 960. 961. 962. 963. 964. 965. 966. 967. 968. 969. 970. 971. 972. 973. 974. 975. 976. 977. 978. 979. 980. 981. 982. 983. 984. 985. 986. 987. 988. 989. 990. 991. 992. 993. 994. 995. 996. 997. 998. 999. 1000.

Projection

```
580         (D3DMATRIX *)m_pWorldStack->GetTop() );
581
582     m_Spheres[m_Planets[i].dwSphere]->Draw();
583 }
584
585 m_pWorldStack->LoadIdentity();
586 m_pD3DDev->SetTransform( D3DTRANSFORMSTATE_WORLD,
587                         (D3DMATRIX *)m_pWorldStack->GetTop() );
588
589 // Draw the particles
590 D3DVERTEX vParticle;
591 for( i = 0; i < NUM_PARTICLES; i++ )
592 {
593     m_pD3DDev->SetMaterial(&m_PlanetMaterials[m_Particles[i].dwMaterial]);
594
595     // Do some fake gravity stuff... (a little too much gravity... :) )
596     ApplyGravity(&m_Particles[i].distance,&m_Particles[i].pos,
597                 &m_Particles[i].bAttract);
598
599     vParticle.dvX = m_Particles[i].pos.x* m_Particles[i].distance;
600     vParticle.dvY = m_Particles[i].pos.y* m_Particles[i].distance;
601     vParticle.dvZ = m_Particles[i].pos.z* m_Particles[i].distance;
602     vParticle.dvNX = -m_Particles[i].pos.x;
603     vParticle.dvNY = -m_Particles[i].pos.y;
604     vParticle.dvNZ = -m_Particles[i].pos.z;
605     m_pD3DDev->DrawPrimitive(D3DPT_POINTLIST,
606                             D3DFVF_VERTEX,
607                             &vParticle,
608                             1,
609                             D3DDP_WAIT );
610 }
611
612
613
614 m_pD3DDev->EndScene();
615 }
616
617 hr = m_pD3DX->UpdateFrame( 0 );
618 if ( hr == DDERR_SURFACELOST || hr == DDERR_SURFACEBUSY )
619     hr = HandleModeChanges();
620
621 return hr;
622 }
623
624 HRESULT CGravity::HandleModeChanges()
625 {
626     HRESULT hr;
627     hr = m_pDD->TestCooperativeLevel();
628
629     if( SUCCEEDED( hr ) || hr == DDERR_WRONGMODE )
630     {
631         UnInit();
632
633         if(FAILED(hr = InitD3DX()))
634             return hr;
635     }
636     else if( hr != DDERR_EXCLUSIVEMODEALREADYSET &&
637             hr != DDERR_NOEXCLUSIVEMODE )
638     {
```

1107

```
639     // Busted!!
640     return hr;
641 }
642 return S_OK;
643 }
644
645 // *****
646 // Windowing Code...
647 // *****
648
649 LRESULT CALLBACK WndProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
650 {
651     switch(uMsg)
652     {
653     case WM_ACTIVATEAPP:
654     {
655         if( !g_pGravity )
656             break;
657
658         if( g_pGravity->m_bIsFullscreen )
659         {
660             if( (BOOL)wParam )
661                 g_pGravity->RestartDrawing();
662             else
663                 g_pGravity->PauseDrawing();
664         }
665     }
666     break;
667 case WM_CREATE:
668     break;
669 case WM_CLOSE:
670     PostQuitMessage(0);
671     break;
672 case WM_SIZE:
673     if( g_pGravity
674         && g_pGravity->m_bD3DXReady
675         && !g_pGravity->m_bIsFullscreen
676     )
677     {
678         HRESULT hr;
679
680         if( wParam == SIZE_MINIMIZED )
681         {
682             g_pGravity->m_bActive = FALSE;
683             break;
684         }
685         else if( LOWORD(lParam)>0 && HIWORD(lParam)>0 )
686         {
687             if( FAILED(hr = g_pGravity->m_pD3DX->Resize(LOWORD(lParam),
688 HIWORD(lParam))))
689             {
690                 InterpretError(hr);
691                 g_pGravity->m_bD3DXReady = FALSE;
692                 PostQuitMessage(0);
693             }
694             g_pGravity->m_bActive = TRUE;
695         }
696     }
```

無しのバグ!
要調査

```

697     break;
698 case WM_KEYDOWN:
699     switch( wParam )
700     {
701     case VK_ESCAPE:
702     {
703         PostQuitMessage(0);
704         break;
705     }
706     }
707     break;
708 case WM_COMMAND:
709     if ( 1 == HIWORD(wParam) )
710     {
711         switch ( LOWORD(wParam) )
712         {
713         case IDM_FULLSCREEN:
714             if( g_pGravity && g_pGravity->m_bD3DReady )
715             {
716                 HRESULT hr;
717                 g_pGravity->m_bIsFullscreen = ! g_pGravity->m_bIsFullscreen;
718                 g_pGravity->m_bD3DReady = FALSE;
719
720                 if ( g_pGravity->m_bIsFullscreen )
721                 {
722                     // going to fullscreen
723                     GetWindowRect( hwnd, &g_pGravity->m_rWindowedRect );
724                 }
725                 ShowCursor(!(g_pGravity->m_bIsFullscreen));
726                 hr = g_pGravity->m_pD3D->Clear(D3DCLEAR_TARGET|D3DCLEAR_ZBUFFER);
727                 if ( FAILED(hr) )
728                 {
729                     InterpretError(hr);
730                     g_pGravity->PauseDrawing();
731                     PostQuitMessage(-1);
732                     break;
733                 }
734                 g_pGravity->UnInit();
735
736                 if ( !g_pGravity->m_bIsFullscreen )
737                 {
738                     RECT& r = g_pGravity->m_rWindowedRect;
739                     SetWindowPos(hwnd, HWND_NOTOPMOST,
740                                 r.left,
741                                 r.top,
742                                 r.right-r.left,
743                                 r.bottom-r.top,
744                                 SWP_NOACTIVATE );
745                 }
746
747                 hr = g_pGravity->InitD3D();
748                 if ( FAILED(hr) )
749                 {
750                     InterpretError(hr);
751                     g_pGravity->PauseDrawing();
752                     PostQuitMessage(-1);
753                     break;
754                 }
755             }

```

ALT + Return

何で?

何で?

有るのか?

一度 Gravity の初期化を消去

フルスクリーンに切り替える

これは?

ウィンドウを作る

フルスクリーンの初期化が失敗した

```
756         break;
757     }
758 }
759 break;
760 default:
761     break;
762 }
763
764 return DefWindowProc(hwnd, uMsg, wParam, lParam);
765
766 }
767
768 int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
769 LPSTR lpszCmdLine, int nCmdShow)
770 {
771     HRESULT hr;
772     MSG msg;
773     WNDCLASS wc;
774     HACCEL hAccelApp;
775     HCURSOR hcur = NULL;
776     int ret = 0;
777
778     g_pGravity = new CGravity; // set up our data AFTER starting up d3dx!
779     if( !g_pGravity )
780     {
781         ret = -1;
782         goto Exit;
783     }
784
785     // Register the window class for the main window.
786
787     if (!hPrevInstance)
788     {
789         hcur = CopyCursor(LoadCursor(NULL, IDC_ARROW));
790
791         wc.style = 0;
792         wc.lpfnWndProc = (WNDPROC) WndProc;
793         wc.cbClsExtra = 0;
794         wc.cbWndExtra = 0;
795         wc.hInstance = hInstance;
796         wc.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_APP_ICON));
797         wc.hCursor = hcur;
798         wc.hbrBackground = (HBRUSH)GetStockObject(BLACK_BRUSH);
799         wc.lpszMenuName = NULL;
800         wc.lpszClassName = NAME_OF_THE_APP;
801
802         if (!RegisterClass(&wc))
803         {
804             ret = -1;
805             goto Exit;
806         }
807     }
808
809     // Create the window
810
811     g_pGravity->m_hwndMain = CreateWindow( NAME_OF_THE_APP,
812     NAME_OF_THE_APP,
813     WS_OVERLAPPEDWINDOW,
814     CW_USEDEFAULT,
```

712~

```
815     CW_USEDEFAULT,  
816     400,  
817     400,  
818     (HWND) NULL,  
819     (HMENU) NULL,  
820     hInstance,  
821     (LPVOID) NULL);  
822  
823     if (!g_pGravity->m_hwndMain)  
824     {  
825         ret = -1;  
826         goto Exit;  
827     }  
828  
829  
830     // Hide the cursor if necessary  
831     if( g_pGravity->m_bIsFullscreen )  
832     {  
833         ShowCursor(FALSE);  
834     }  
835  
836     // Show the window  
837     ShowWindow(g_pGravity->m_hwndMain, nCmdShow);  
838     UpdateWindow(g_pGravity->m_hwndMain);  
839  
840     hAccelApp = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDR_APP_ACCELERATOR));  
841     if ( !hAccelApp )  
842     {  
843         ret = -1;  
844         goto Exit;  
845     }  
846  
847     // Initialize D3DX  
848     hr = g_pGravity->InitD3DX();  
849     if ( FAILED(hr) )  
850     {  
851         InterpretError(hr);  
852         ret = -1;  
853         goto Exit;  
854     }  
855  
856     BOOL bGotMsg;  
857     PeekMessage( &msg, NULL, 0U, 0U, PM_NOREMOVE );  
858     while( WM_QUIT != msg.message )  
859     {  
860         bGotMsg = PeekMessage( &msg, NULL, 0U, 0U, PM_REMOVE );  
861  
862         if( bGotMsg )  
863         {  
864             if ( !TranslateAccelerator( g_pGravity->m_hwndMain, hAccelApp, &msg ) )  
865             {  
866                 TranslateMessage( &msg );  
867                 DispatchMessage( &msg );  
868             }  
869         }  
870         else  
871         {  
872             if( g_pGravity && g_pGravity->m_bActive )  
873             {
```

このところ

```
874         hr = g_pGravity->Draw();
875         if( FAILED(hr) )
876         {
877             InterpretError(hr);
878             g_pGravity->m_bD3DXReady = FALSE;
879             PostQuitMessage(-1);
880         }
881     }
882     else
883     {
884         WaitMessage();
885     }
886 }
887 }
888 delete g_pGravity; // clean up our data BEFORE shutting down d3dx!
889
890 Exit:
891     if(hcur)
892         DestroyCursor(hcur);
893
894     return ret;
895 }
```