บทที่ 3
การทดสอบเฟรมเวิร์คที่พัฒนา

ในบทนี้เป็นเรื่องเกี่ยวกับการใช้งานและสิ่งที่ได้จากการศึกษา ตัวอย่างการใช้งานเกมเฟรมเวิร์คในด้านต่างๆ เช่น การรับอินพุตของผู้เล่น (input) การปรับตำแหน่งของกล้อง (update camera) การจัดการกับวัตถุในเกม (Game Object Management) การตรวจจับการชน (Collision Detection) ปัญญาประดิษฐ์ของเกม (Artificial Intelligence)

3.1 ตัวอย่างที่ 1 การสร้างพื้นผิว (Polygon)

ตัวอย่างนี้แสดงการเรียกใช้คลาส vector3D และ polygon ในการเก็บข้อมูลของเวอร์เท็กซ์ (Vertex) แล้วนำมาวาดรูปด้วยคลาส render โดยแสดงในตัวอักษรเข้ม ส่วนที่เป็นการเขียนโปรแกรมบนวินโดส์ใช้ตัวอักษรธรรมดา จะขออธิบายเฉพาะส่วนที่เป็นสีเข้ม และในตัวอย่างต่อไปจะขอแสดงเฉพาะส่วนที่เรียกใช้คลาสเท่านั้น

```
#include "classfile.h"

// Windows Globals
const char      *szWndClass         = "GameFramework";
const char      *szProgramName      = "GameFramework Example";
HINSTANCE g_hInstance;
HWND            g_hWnd;

// Function Prototypes
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow);
LRESULT WINAPI MsgProc( HWND hWnd, UINT msg, WPARAM wParam,
LPARAM lParam );
//HRESULT hrInit3D( HWND hWnd );
//void vCleanup(void);
void myInit();

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow)
{
        WNDCLASS  wc;
```

```
        HWND            hWnd;
        MSG             msg;

        // Set up and register window class
    wc.style = CS_HREDRAW | CS_VREDRAW;
    wc.lpfnWndProc = MsgProc;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hInstance = hInstance;
    wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = (HBRUSH )GetStockObject(BLACK_BRUSH);
    wc.lpszMenuName = NULL;
    wc.lpszClassName = szWndClass;
    RegisterClass(&wc);

    // Create a window
    hWnd = CreateWindowEx(WS_EX_TOPMOST,
                szWndClass,
                szProgramName,
                WS_OVERLAPPEDWINDOW,
                0,
                0,
                800,
                600,
                NULL,
                NULL,
                hInstance,
                NULL);

        // Setup global window handles
        g_hInstance = hInstance;
        g_hWnd = hWnd;

        ShowWindow(hWnd, nCmdShow);
        UpdateWindow(hWnd);

1       polygon floor1;
2       vector3D floor1v0(-50, 0, -200);
3       vector3D floor1v1(50, 0, -200);
4       vector3D floor1v2(-50, 0, 200);
5       floor1.setVector(floor1v0, floor1v1, floor1v2);
6       polygon floor2;
7       vector3D floor2v0(-50, 0, 200);
8       vector3D floor2v1(50, 0, -200);
9       vector3D floor2v2(50, 0, 200);
10      floor2.setVector(floor2v0, floor2v1, floor2v2);
```

```
11      vector3D v0(-2.5, 0, 0, 0xFFF00000);
12      vector3D v1(2.5, 0, 0, 0x000FFF00);
13      vector3D v2(0, 5, 0, 0x00000FFF);
14      polygon p(v0, v1, v2);

15      camera camera1(0.0f, 2.5f, -50.0f);
16      render render1(g_hWnd, camera1);
17      render1.clearScene();
18      render1.setCamera(camera1);
19      render1.setProjection();
20      render1.drawtoBackBuffer(p);
21      render1.drawtoBackBuffer(floor1);
22      render1.drawtoBackBuffer(floor2);
23      render1.present();

        while(1)
        {
                if (PeekMessage(&msg, NULL, 0, 0, PM_NOREMOVE))
                {
                        if ( !GetMessage ( &msg, NULL, 0, 0 ) )
                                return msg.wParam;
                        TranslateMessage( &msg );
                        DispatchMessage( &msg );
                }
                else
                {

                }
        }

        return(msg.wParam);
};

LRESULT WINAPI MsgProc( HWND hWnd, UINT msg, WPARAM wParam,
LPARAM lParam )
{
        switch( msg )
    {
        case WM_DESTROY:
        PostQuitMessage( 0 );
                        return 0;
    }

    return DefWindowProc( hWnd, msg, wParam, lParam );
}
```

ส่วนที่เป็นสีเข้มเป็นการเรียกใช้คลาสต่างๆของเกมเฟรมเวิร์ค

คำอธิบาย ซอร์สโค้ด (Source Code)

บรรทัดที่1 ประกาศตัวแปรpolygon ชื่อfloor1

บรรทัดที่2-4 ประกาศตัวแปรvector3D เพื่อเป็นเวอร์เท็กซ์ (Vertex) ของfloor1

บรรทัดที่5 ใส่ค่าvector3Dให้กับpolygon ชื่อfloor1

บรรทัดที่6-10 เป็นส่วนของ floor2

บรรทัดที่11-14 ประกาศตัวแปร vector3D และตั้งค่าเวอร์เท็กซ์ (Vertex) ให้ polygonแบบใช้ตัวสร้างของคลาสpolygon

บรรทัดที่15 ประกาศตัวแปร camera โดยให้กล้องอยู่ที่ x=0 y=2.5 z=-50

บรรทัดที่16 ประกาศตัวแปร render ส่งตัวแปร handle ของวินโดส์และตัวแปร camera

บรรทัดที่17 เรียกใช้ฟังก์ชัน clearSceneของคลาส render เพื่อลบหน้าจอ

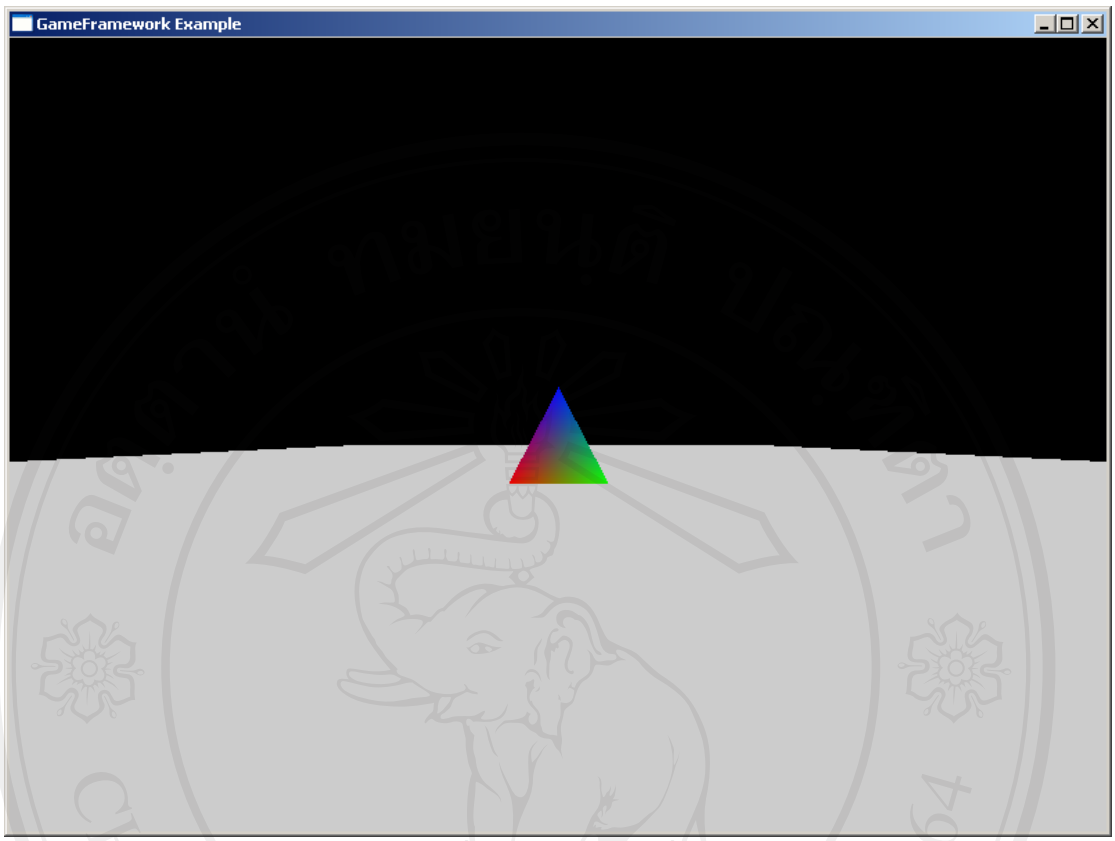
บรรทัดที่18 เรียกใช้ฟังก์ชัน setCameraของคลาส render เพื่อตั้งค่าของกล้อง

บรรทัดที่19 เรียกใช้ฟังก์ชัน setProjectionของคลาส render เพื่อตั้งค่าperspectiveการมองของกล้อง

บรรทัดที่20-22 เรียกใช้ฟังก์ชัน drawtoBackBufferของคลาส render เพื่อวาดรูปสู่ back buffer

บรรทัดที่23 เรียกใช้ฟังก์ชัน presentของคลาส render เพื่อวาดรูปสู่หน้าจอ

ผลลัพธ์เป็นดังรูปที่ 3.1



รูปที่ 3.1 รูปตัวอย่างที่ 1

3.2 ตัวอย่างที่ 2 การสร้าง gameObject

ตัวอย่างที่ 2 แสดงการสร้างวัตถุโดยใช้ข้อมูลจากคลาส vector3D, polygon ใส่ข้อมูลให้คลาส gameObject ในตัวอย่างนี้ตัดส่วนที่เกี่ยวกับการเขียนโปรแกรมวินโดส์ออก ส่วนที่คล้ายกับตัวอย่างที่1จะเป็นตัวอักษรธรรมดา ส่วนที่ต่างจะเป็นตัวอักษรเข้ม ส่วนที่เหมือนจะขออธิบายสั้นๆ

```
1       polygon floor1;
2       vector3D floor1v0(-50, 0, -200);
3       vector3D floor1v1(50, 0, -200);
4       vector3D floor1v2(-50, 0, 200);
5       floor1.setVector(floor1v0, floor1v1, floor1v2);
6       polygon floor2;
7       vector3D floor2v0(-50, 0, 200);
8       vector3D floor2v1(50, 0, -200);
9       vector3D floor2v2(50, 0, 200);
10      floor2.setVector(floor2v0, floor2v1, floor2v2);

11      gameObject gameObjectFloor1(floor1);
12      gameObject gameObjectFloor2;
13      gameObjectFloor2.setPolygon(floor2);

14      vector3D v0(-2.5, 0, 0, 0xFFF00000);
15      vector3D v1(2.5, 0, 0, 0x000FFF00);
16      vector3D v2(0, 5, 0, 0x00000FFF);
17      polygon p(v0, v1, v2);

18      camera c(0.0f, 2.5f, -30.0f);
19      render r(g_hWnd, c);

20      r.clearScene();
21      r.setCamera(c);
22      r.setProjection();
23      r.drawtoBackBuffer(p);
24      r.drawtoBackBuffer(gameObjectFloor1);
25      r.drawtoBackBuffer(gameObjectFloor2);
26      r.present();
```

คำอธิบาย ซอร์สโค้ด (Source Code)

บรรทัดที่1-10 เป็นการตั้งค่าเวอร์เท็กซ์ (Vertex) กับ พื้นผิว (Polygon)

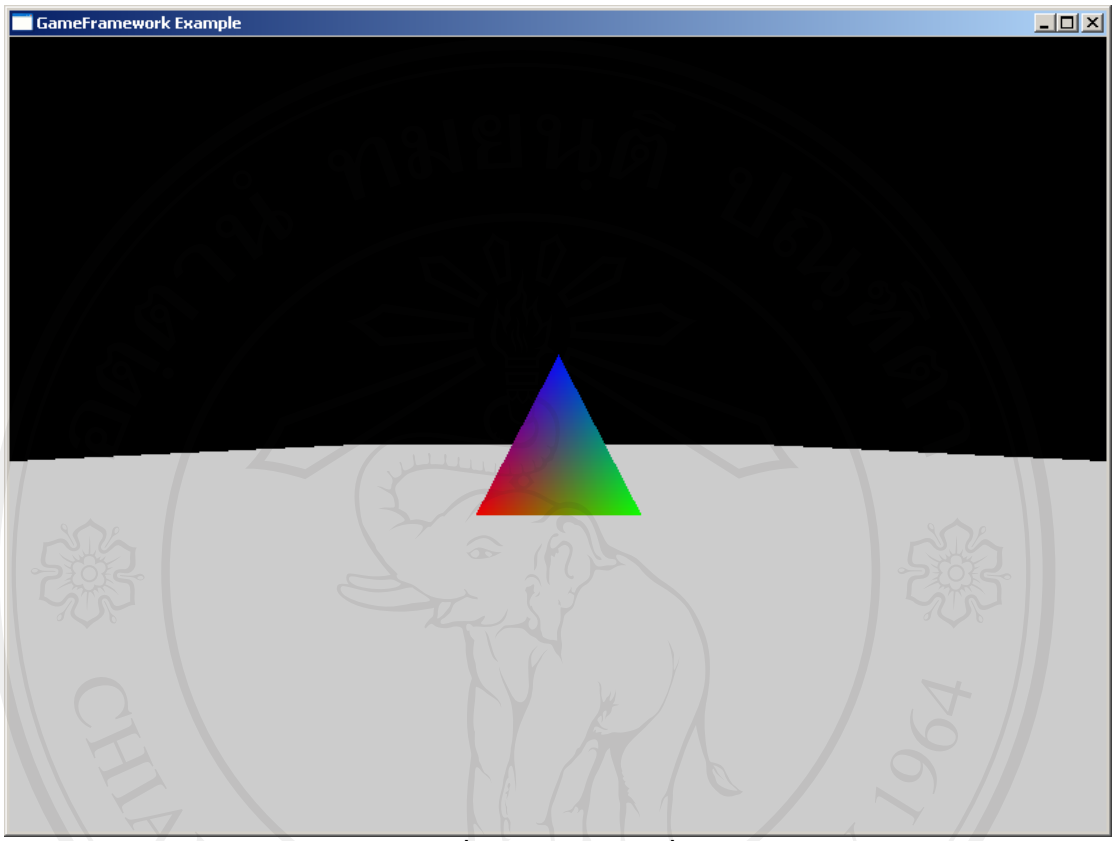
บรรทัดที่11 ประกาศตัวแปร gameObject โดยใช้ตัวสร้างในการset ค่า polygon

บรรทัดที่12 ประกาศตัวแปร gameObject ชื่อ gameObjectFloor2

บรรทัดที่13 set ค่า polygon ให้ gameObjectFloor2โดยใช้ฟังก์ชัน setPolygon

บรรทัดที่14-26 เป็นการตั้งค่า polygon camera และวาดรูปคล้ายกับตัวอย่างที่1

ผลลัพธ์เป็นดังรูปที่ 3.2 ซึ่งเหมือนกับตัวอย่างที่1แต่เปลี่ยนจาก polygon เป็น gameObject



รูปที่ 3.2 รูปตัวอย่างที่ 2

3.3 ตัวอย่างที่ 3 การสร้าง gridGame

ตัวอย่างที่ 3 สร้างตัวแปร vector3D, polygon แล้วใส่เข้าไปในคลาส gameObject เพื่อสร้างวัตถุแล้วใส่วัตถุเข้าไปในคลาส gridGame เพื่อบริหารจัดการวัตถุในเกมแล้วช่วยลดการวาดรูปเป็นการแสดงการใช้คลาส gridGame

```
1     polygon floor1;
2     vector3D floor1v0(-50, 0, -200);
3     vector3D floor1v1(50, 0, -200);
4     vector3D floor1v2(-50, 0, 200);
5     floor1.setVector(floor1v0, floor1v1, floor1v2);
6     polygon floor2;
7     vector3D floor2v0(-50, 0, 200);
8     vector3D floor2v1(50, 0, -200);
9     vector3D floor2v2(50, 0, 200);
10    floor2.setVector(floor2v0, floor2v1, floor2v2);

11    polygon p;
12    vector3D v0(-2.5, 0, 0, 0xFFF00000);
13    vector3D v1(2.5, 0, 0, 0x000FFF00);
14    vector3D v2(0, 5, 0, 0x00000FFF);
15    p.setVector(v0, v1, v2);

16    gameObject go(p);
17    gridGame gg(go);

18    camera c(0.0f, 2.5f , -50.0f);
19    render r(g_hWnd, c);

20    gg.updateVisible(c);
21    r.clearScene();
22    r.setCamera(c);
23    r.setProjection();
24    r.drawtoBackBuffer(gg);
25    r.drawtoBackBuffer(floor1);
26    r.drawtoBackBuffer(floor2);
27    r.present();
```

ในตัวอย่างนี้ก็เหมือนกับตัวอย่างก่อนหน้าแต่มีส่วนที่ต่างคือส่วนที่เป็นสีเข้ม

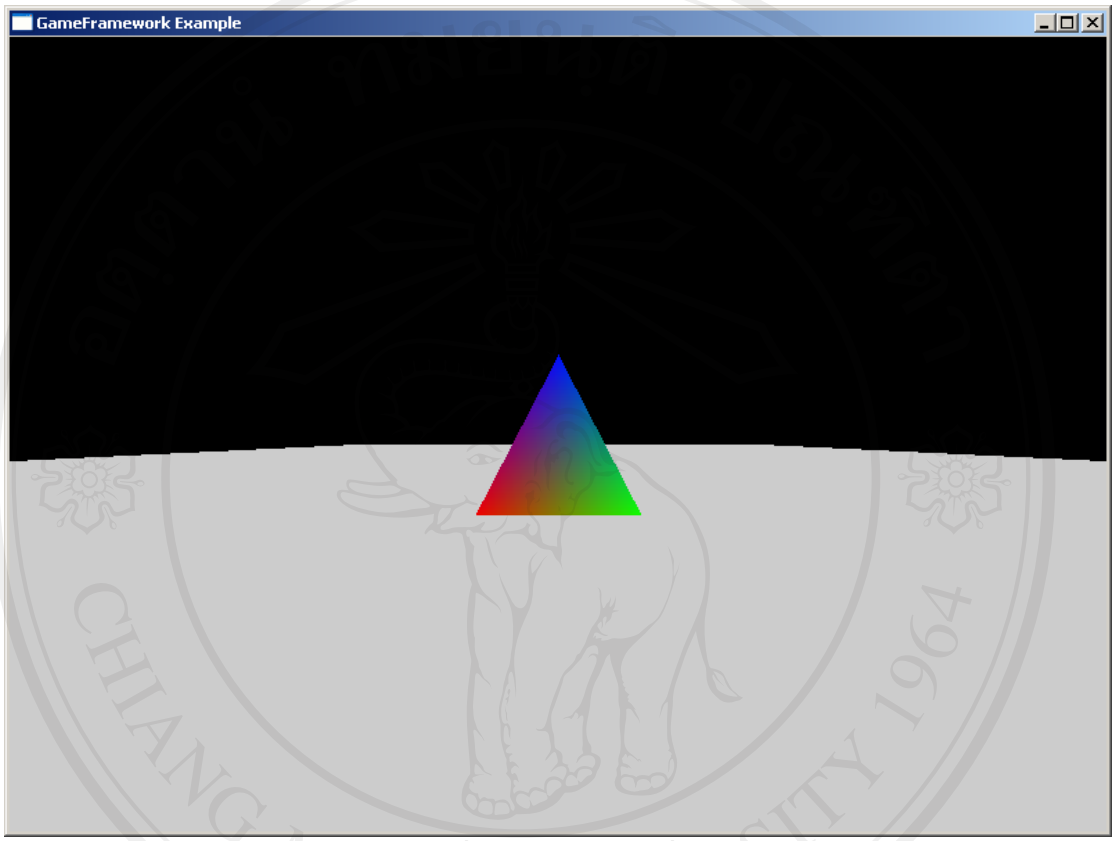
คำอธิบาย ซอร์สโค้ด (Source Code)

บรรทัดที่17 เป็นการประกาศตัวแปร gridGame ชื่อ gg รับค่า ganeObject เข้ามาสร้าง grid

บรรทัดที่20 เรียกให้ตัวแปร grigGame update ว่ากริดไหนบ้างที่visible

บรรทัดที่24 เรียกให้วาดรูปโดยส่งให้วาดรูปทั้งกริด ฟังก์ชันจะวาดเฉพาะที่ visible เท่านั้น

ผลลัพธ์เป็นดังรูปที่ 3.3 ซึ่งเหมือนกับตัวอย่างที่1และ2แต่เปลี่ยนเป็นวาดด้วยฟังก์ชันที่มีพารามิเตอร์ที่เป็น gridGame



รูปที่ 3.3 รูปตัวอย่างที่ 3

3.4 ตัวอย่างที่ 4 การทดสอบการรับอินพุตและการชน

ตัวอย่างที่ 4 สร้างการรับอินพุตและปรับตำแหน่งของวัตถุในเกมไปตามการรับอินพุตแล้วมีการทดสอบการชนในแต่ละการเคลื่อนไหว เพื่อแสดงตัวอย่างสำหรับการใช้งานคลาส input, camera และฟังก์ชันเกี่ยวกับการตรวจจับการชนในคลาส camera

```
1    polygon floor1;
2    vector3D floor1v0(-50, 0, -200);
3    vector3D floor1v1(50, 0, -200);
4    vector3D floor1v2(-50, 0, 200);
5    floor1.setVector(floor1v0, floor1v1, floor1v2);
6    polygon floor2;
7    vector3D floor2v0(-50, 0, 200);
8    vector3D floor2v1(50, 0, -200);
9    vector3D floor2v2(50, 0, 200);
10   floor2.setVector(floor2v0, floor2v1, floor2v2);

11   polygon p;
12   vector3D v0(-2.5, 0, 0, 0xFFF00000);
13   vector3D v1(2.5, 0, 0, 0x000FFF00);
14   vector3D v2(0, 5, 0, 0x00000FFF);
15   p.setVector(v0, v1, v2);

16   gameObject go(p);
17   go.setName("triangle");
18   gridGame gg(go);

19   camera c(0.0f, 2.5f, -50.0f);
20   render r(g_hWnd, c);

21   gg.updateVisible(c);
22   r.clearScene();
23   r.setCamera(c);
24   r.setProjection();
25   r.drawtoBackBuffer(gg);
26   r.drawtoBackBuffer(floor1);
27   r.drawtoBackBuffer(floor2);
28   r.present();

29   int k;
30   input i(g_hInstance, g_hWnd);

31   // Process messages until the program is terminated
32   cell *tempcell;
33   int lp;
34   bool b=false;
```

```
35      bool bsee=false;
36      bool bhear=false;
37      char buffer[500];
38      while(1)
39      {
40              if (PeekMessage(&msg, NULL, 0, 0, PM_NOREMOVE))
41              {
42                      if ( !GetMessage ( &msg, NULL, 0, 0 ) )
43                              return msg.wParam;
44                      TranslateMessage( &msg );
45                      DispatchMessage( &msg );
46              }
47              else
48              {

49                       k=i.getKey();

50                      if(k==keyUP) {
51                      c.cameraForward();
52                      gg.updateVisible(c);
53                      tempcell=gg.getCell(c);
54                      c.clearHitObjects();
55                      c.hitObject(*tempcell);
56                      if(c.vhitObjects.size()!=0)
57                      {
58                      c.location=c.oldLocation; c.target=c.oldTarget;
59                      }
60                      r.clearScene();
61                      r.setCamera(c);
62                      r.setProjection();
63                      r.drawtoBackBuffer(gg);
64                      r.drawtoBackBuffer(floor1);
65                      r.drawtoBackBuffer(floor2);
66                      r.present();

67                      sprintf( buffer, "numobject %d", c.numHitObjects);

68                      r.drawText(hWnd, 0, 0, buffer, 4200000000, 0xFFFFFFFF,
false);
69                      for(int i=0;i<c.numHitObjects;i++)
70
        r.drawText(hWnd,0,80,c.vhitObjects.at(i).name,4200000000,0xFFFFFFF
F,false);
71                      }

72                      if(k==keyDOWN) {
73                      c.cameraBackward();
```

```
74                  gg.updateVisible(c);
75                  tempcell=gg.getCell(c);
76                  for(lp=0;lp<tempcell->numObjects;lp++){
77                  b=c.hitObject(tempcell->objects[lp]);
78                  if(b==true)
79                  { c.location=c.oldLocation; c.target=c.oldTarget;}
80                  }
81                  r.clearScene();
82                  r.setCamera(c);
83                  r.setProjection();
84                  r.drawtoBackBuffer(go);
85                  r.drawtoBackBuffer(floor1);
86                  r.drawtoBackBuffer(floor2);
87                  r.present();
88                  }

89                  if(k==keyLEFT) {
90                  c.cameraTurnLeft();
91                  r.clearScene();
92                  r.setCamera(c);
93                  r.setProjection();
94                  r.drawtoBackBuffer(go);
95                  r.drawtoBackBuffer(floor1);
96                  r.drawtoBackBuffer(floor2);
97                  r.present();
98                  }
99                  if(k==keyRIGHT){
100                 c.cameraTurnRight();
101                 r.clearScene();
102                 r.setCamera(c);
103                 r.setProjection();
104                 r.drawtoBackBuffer(go);
105                 r.drawtoBackBuffer(floor1);
106                 r.drawtoBackBuffer(floor2);
107                 r.present();
108                 }
109         }
110 }
```

คำอธิบาย ซอร์สโค้ด (Source Code)

บรรทัดที่1-30 เป็นการสร้างวัตถุในเกม

บรรทัดที่30 เป็นการประกาศตัวแปรประเภท inputทำลูปเพื่อรอรับค่าจากการกดคีย์บอร์ดของผู้เล่น

บรรทัดที่38 เป็นการวนลูปevent

บรรทัดที่49 เป็นการรับค่าการกดคีย์บอร์ดของผู้เล่นจากตัวปร input

บรรทัดที่50-71 เป็นกรณีการกดคีย์บอร์ด ลูกศรขึ้น

บรรทัดที่51 เรียกฟังก์ชัน cameraForward ฟังก์ชันจะทำการเลื่อนตำแหน่งกล้องขึ้นไปข้างหน้า และทำการเปลี่ยนตำแหน่งการมองของกล้องด้วย

บรรทัดที่52 เรียกฟังก์ชัน updateVisible ฟังก์ชันจะทำการupdate กริดว่ากริดไหน visible โดยรับค่า camera มาปรับการ visibleของกริด

บรรทัดที่53 เรียกรับค่าจากฟังก์ชัน getCell เพื่อรับค่ากริดที่ camera อยู่

บรรทัดที่54 เรียกฟังก์ชัน clearHitObjects ฟังก์ชันจะลบข้อมูลที่เก็บว่ากล้องชนวัตถุอะไรบ้าง

บรรทัดที่55 เรียกฟังก์ชัน hitObject ฟังก์ชันจะทดสอบการชนโดยทดสอบเฉพาะกริดที่ส่งเข้าไป

บรรทัดที่56-59 ข้อมูลที่เก็บว่ามีการชนอะไรบ้างจะมีขนาดไม่เท่ากับ 0 จะให้ค่าตำแหน่งกล้องให้เป็นตำแหน่งเดิมเหมือนการเดินชนแล้วหยุด

บรรทัดที่60-66 วาดรูป

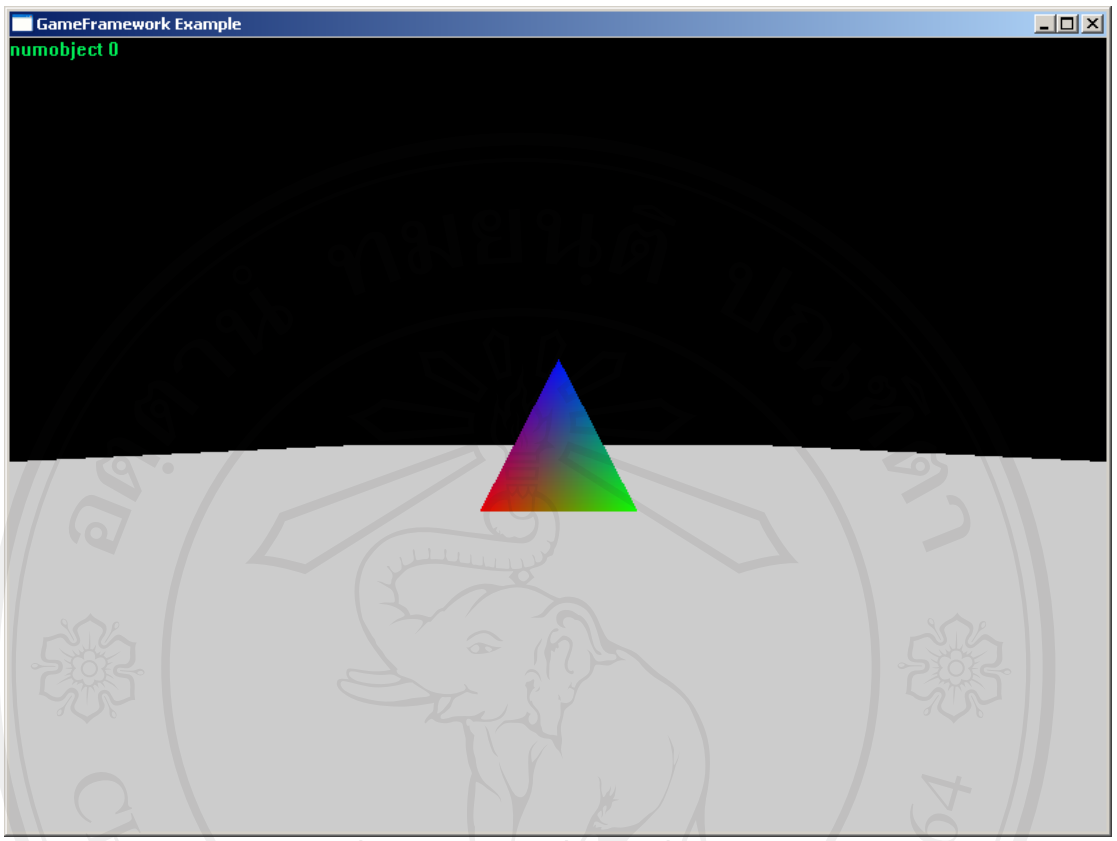
บรรทัดที่67 สร้างข้อความว่ามีกี่วัตถุที่ชน

บรรทัดที่68 วาดข้อความที่สร้างไว้ในบรรทัดที่67

บรรทัดที่69-71 เป็นการวนลูปวาดข้อความที่ชื่อของวัตถุที่ชน

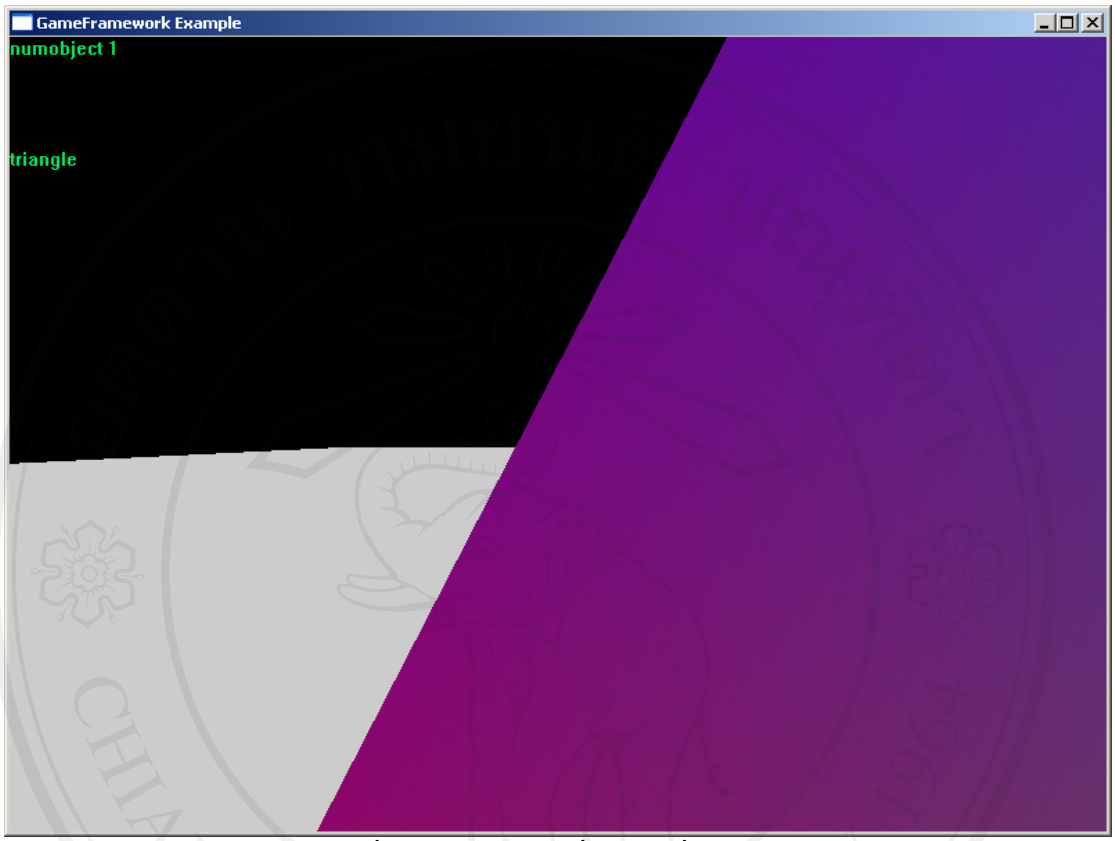
บรรทัดที่72-108 เป็นการกดคีย์บอร์ด ลูกศรลง เลี้ยวซ้าย เลี้ยวขวา

กรณีที่ยังไม่มีการชน ดังรูปที่ 3.4



รูปที่ 3.4 รูปตัวอย่างที่ 4 กรณีที่ไม่มีการชน

กรณีที่มีการชน ดังรูปที่ 3.5



รูปที่ 3.5 รูปตัวอย่างที่ 4 กรณีที่มีการชน

3.5 ตัวอย่างที่ 5 การมองเห็น

ตัวอย่างที่ 5 สร้างการรับอินพุตและปรับตำแหน่งของวัตถุในเกมไปตามการรับอินพุตแล้ว มีการทดสอบการมองเห็นของวัตถุซึ่งเป็นฟังก์ชันที่อยู่ในคลาส gameObject เพื่อแสดงตัวอย่างสำหรับการใช้งานฟังก์ชันเกี่ยวกับการตรวจจับการมองเห็นที่อยู่ในคลาส gameObject

```
1       k=i.getKey();

2       if(k==keyUP) {
3       c.cameraForward();
4       r.clearScene();
5       r.setCamera(c);
6       r.setProjection();
7       r.drawtoBackBuffer(go);
8       r.drawtoBackBuffer(floor1);
9       r.drawtoBackBuffer(floor2);
10      r.present();

11      bool1=go.canSee(c.location);
12      if(bool1==true){
13              tempstr="";
14              tempstr+=go.name;
15              tempstr+=" see";
16              r.drawText(hWnd, 0, 80,  tempstr, 4200000000, 0xFFFFFFFF, false);
17      }
18      else{
19              tempstr="";
20              tempstr+=go.name;
21              tempstr+=" Not see";
22              r.drawText(hWnd, 0, 80, tempstr, 4200000000, 0xFFFFFFFF, false);
23      }
24      }//END KEYUP
```

ตัวอย่างนี้ตัดมาเฉพาะส่วนที่รับinputลูกศรขึ้น ตัวอย่างนี้คล้ายกับตัวอย่างที่แล้วขออธิบายในส่วนที่ต่างคือส่วนที่เป็นสีเข้ม

คำอธิบาย ซอร์สโค้ด (Source Code)

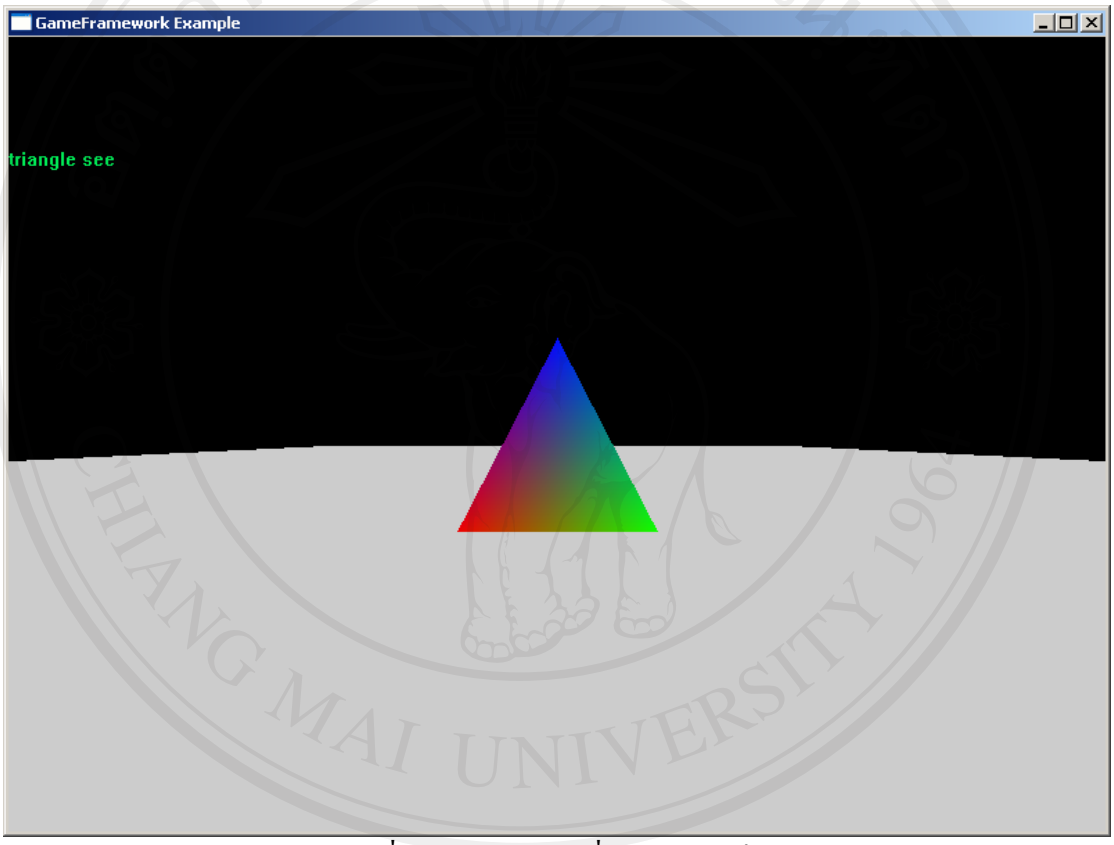
บรรทัดที่11 เรียกฟังก์ชัน canSee ของคลาส gameObject โดยส่งตัวแปร camera เพื่อบอกตำแหน่งของกล้อง ฟังก์ชันจะทดสอบว่า gameObject นี้มองเห็นกล้องหรือไม่

บรรทัดที่12 ถ้าการเรียกฟังก์ชันรีเทิร์น true มาแสดงว่ามีการเห็นกล้องของวัตถุ

บรรทัดที่13-17 สร้าง string ขึ้นมาเพื่อวาดข้อความ see

บรรทัดที่18-23 สร้าง string ขึ้นมาเพื่อวาดข้อความ Not see

ผลลัพธ์เป็นดังรูปที่ 3.6



รูปที่ 3.6 รูปตัวอย่างที่ 5 การมองเห็น

3.6 ตัวอย่างที่ 6 การได้ยิน

ตัวอย่างที่ 6 สร้างการรับอินพุตและปรับตำแหน่งของวัตถุในเกมไปตามการรับอินพุตแล้วมีการทดสอบการได้ยินของวัตถุซึ่งเป็นฟังก์ชันที่อยู่ในคลาส gameObject เพื่อแสดงตัวอย่างสำหรับการใช้งานฟังก์ชันเกี่ยวกับการตรวจจับการได้ยินที่อยู่ในคลาส gameObject

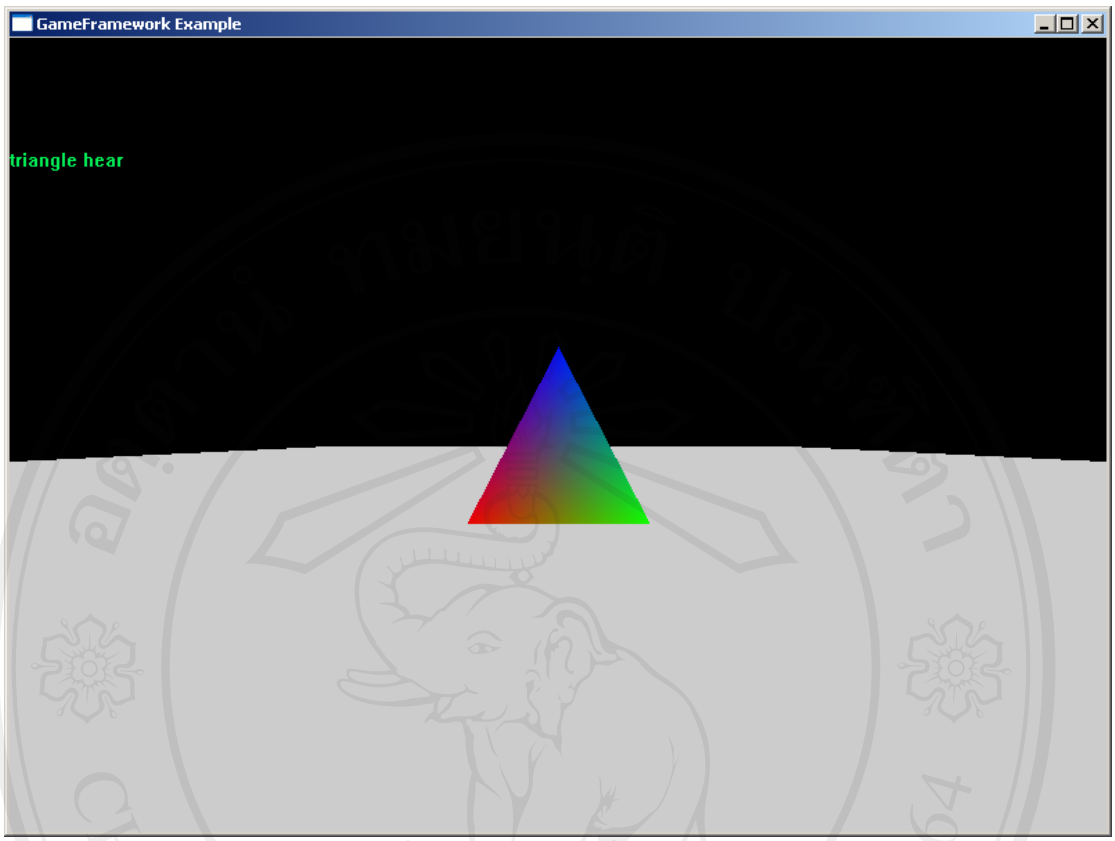
```
1       k=i.getKey();

2       if(k==keyUP) {
3               c.cameraForward();
4               r.clearScene();
5               r.setCamera(c);
6               r.setProjection();
7               r.drawtoBackBuffer(go);
8               r.drawtoBackBuffer(floor1);
9               r.drawtoBackBuffer(floor2);
10              r.present();

11              bool1=go.canHear(c.location);
12              if(bool1==true){
13                      tempstr="";
14                      tempstr+=go.name;
15                      tempstr+=" hear";
16                      r.drawText(hWnd, 0, 80, tempstr, 4200000000, 0xFFFFFFFF,
false);
17                              }
18              else{
19                      tempstr="";
20                      tempstr+=go.name;
21                      tempstr+=" Not hear";
22                      r.drawText(hWnd, 0, 80, tempstr, 4200000000, 0xFFFFFFFF,
false);
23              }
24      }//END KEYUP
```

ตัวอย่างนี้ก็เหมือนตัวอย่างการมองเห็นแต่เปลี่ยนเป็นเรียกฟังก์ชันทดสอบการได้ยินแทนในบรรทัดที่11

ผลลัพธ์เป็นดังรูปที่ 3.7



รูปที่ 3.7 รูปตัวอย่างที่ 6 การได้ยิน

3.7 ตัวอย่างที่ 7 การติดต่อสื่อสาร

ตัวอย่างที่ 7 สร้างการติดต่อสื่อสารโดยใช้คลาส socketObject มีปุ่ม 3 ปุ่มคือ 1. start 2. connect และ3. send ทดสอบโปรแกรมตามรูปที่ 3.8 โดยเปิดโปรแกรมขึ้นมา 2 โปรแกรม โปรแกรมแรกให้แทน server โปรแกรมที่ 2 แทน client ตามรูปที่ 3.9 ให้โปรแกรมทางด้านซ้ายเป็น server โปรแกรมทางด้านขวาเป็น client

ตัวอย่างซอร์สโค้ด (Source Code) ในตัวอย่างที่ 7 ตัดส่วนที่เกี่ยวกับการสร้างวินโดส์ออกนำมาเฉพาะส่วนที่เกี่ยวกับการใช้งาน

```
1. char serverstring[]="127.0.0.1";

2. socketObject        ServerSocketObject;
3. socketObject        ClientSocketObject;
4. char DataPacket[128];              // Data Packet to Receive
5.int iBytesReceived = 0;             // # of Bytes Received

6.socketObjectClientSocketObject2;
7.char DataPacket2[128];              // Data Packet to Transmit
8.int iBytesSent = 0;                 // # of Bytes Sent

9.LRESULT CALLBACK WndProc( HWND hwnd, UINT msg, WPARAM wParam,
10.LPARAM lParam )
11.{

12.switch(msg)
13.  {
14.  case WM_CREATE:
15.        {
16.     button1 =  CreateWindow(TEXT("button"), TEXT("1 START"),
17.                    WS_VISIBLE | WS_CHILD | BS_PUSHBUTTON,
18.                   80, 10, 100, 50,
19.                    hwnd, (HMENU) 1, NULL, NULL);
20.     button2 =  CreateWindow(TEXT("button"), TEXT("2 CONNECT"),
21.                    WS_VISIBLE | WS_CHILD | BS_PUSHBUTTON,
22.                   180, 10, 100, 50,
23.                    hwnd, (HMENU) 2, NULL, NULL);
24.              button3 =  CreateWindow(TEXT("button"), TEXT("3 SEND"),
25.                    WS_VISIBLE | WS_CHILD | BS_PUSHBUTTON,
```

```
                    180, 60, 100, 50,
                     hwnd, (HMENU) 3, NULL, NULL);
        break;
    }
case WM_COMMAND:
    {
    switch(LOWORD(wParam))
      {
       case 1:
      button1start(6000);
      break;
       case 2:
       button2connect( serverstring, 6000 );
      break;
       case 3:
       button3send();
      break;
      }
      break;
    }
case 3000:
      switch (WSAGETSELECTEVENT(lParam))
    {
      case FD_ACCEPT: //Connection request
      onAccept();
      break;
      case FD_CLOSE: //Lost connection
      MessageBox(hwnd, "FD_CLOSE", "FD_CLOSE", MB_OK);
      break;
      case FD_READ: //Incoming data to receive
      onReceive();
      break;
       case FD_WRITE: //Incoming data to receive
      break;
      }
  break;
}
```

```
63. return DefWindowProc(hwnd, msg, wParam, lParam);
64. }

65. void button1start( int iListenPort )
66. {
67.  int i;
68.      if ( i=ServerSocketObject.Bind( iListenPort ) )
69.      {
70.              ServerSocketObject.Listen();
71.      }
72.    if(i!=0)
73.    MessageBox(NULL, "bind true", "bind true", MB_OK);
74.    else
75.    MessageBox(hwnd, "bind false", "bind false", MB_OK);
76.}

77. void button2connect( char szServerIP[], int iServerListenPort )
78. {
79. bool b;
80.      if( b=ClientSocketObject2.Connect( szServerIP, iServerListenPort ) )
81.      {
82.              strcpy(DataPacket2, "TestData from Client");
83.      }
84.      if(b==true)
85.      MessageBox(hwnd, "connect true", "Connect true", MB_OK);
86.      else
87.      MessageBox(hwnd, "connect false", "Connect false", MB_OK);
88. }

89. void button3send()
90. {
91.      iBytesSent = ClientSocketObject2.Send(DataPacket2, 128, 0);
92. }
93. void onAccept()
94. {
95.      ServerSocketObject.Accept(ClientSocketObject);
96.      char szIP[16];
97.      sprintf(szIP, "%d.%d.%d.%d", ClientSocketObject.getIPb1(),
98.              ClientSocketObject.getIPb2(),
99.              ClientSocketObject.getIPb3(),
```

```
100.            ClientSocketObject.getIPb4());
101.    szIP[15]='\0';
102.    MessageBox(hwnd, szIP, szIP, MB_OK);
103. }

104. void onReceive()
105. {
106.    ClientSocketObject.Recv(DataPacket, 128, 0);
107.    MessageBox(hwnd, DataPacket, "On receive", MB_OK);
108. }
```

คำอธิบาย ซอร์สโค้ด (Source Code)

บรรทัดที่1-8 เป็นการสร้างตัวแปร global

บรรทัดที่9 เป็นฟังก์ชันที่เกี่ยวกับeventของวินโดส์ ฟังก์ชันจบที่บรรทัด67

บรรทัดที่14-29 เป็นการสร้างปุ่มกด

บรรทัดที่30-45 เป็นeventที่เกิดเมื่อมีการกดปุ่ม

เมื่อกดปุ่มที่1เรียกฟังก์ชันbutton1start(6000);

เมื่อกดปุ่มที่2เรียกฟังก์ชัน button2connect(serverstring, 6000);

เมื่อกดปุ่มที่3เรียกฟังก์ชัน button3send();

บรรทัดที่46-64 เป็นevent เกิดขึ้นจากโปรแกรมทำงานทางเน็ตเวิร์ค

event FD_ACCEPT เรียกฟังก์ชัน onAccept(); eventนี้เกิดจากการมีการมี connect จากclient แล้ว server พร้อมที่จะทำการ accept connect นี้

event FD_CLOSE ให้แสดง dialog ว่า FD_CLOSE eventนี้เกิดเมื่อมีการclose socket

event FD_READ เรียกฟังก์ชัน onReceive(); eventนี้เกิดเมื่อsocket พร้อมที่จะรับข้อมูล

event FD_WRITE eventนี้เกิดเมื่อsocket พร้อมที่จะส่งข้อมูล

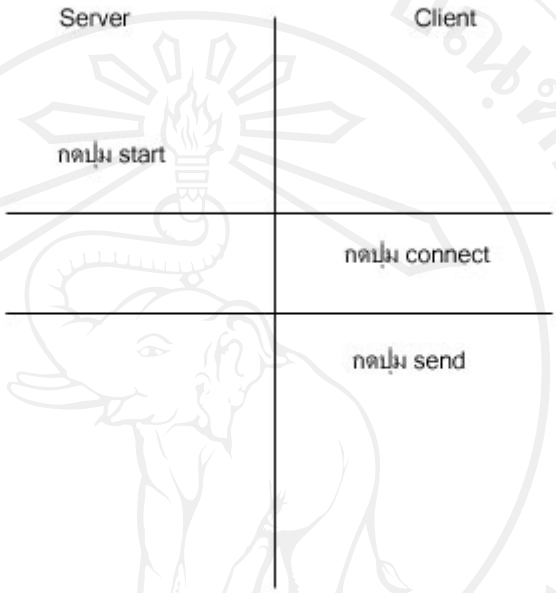
บรรทัดที่65-76 ฟังก์ชัน button1start(); ฟังก์ชันเรียกฟังก์ชัน Bind และ Listen ของคลาส socketObject เพื่อทำการ bind port และรอฟังการ connect ของ client

บรรทัดที่77-88 ฟังก์ชัน button2connect() ฟังก์ชันนี้ทำการ connect ไปที่ server แล้ว copy คำว่า TestData from Client ถ้าconnectสำเร็จจะแสดงdialogว่า connect true ไม่สำเร็จแสดงคำว่า connect false ฟังก์ชันนี้ทำงานเมื่อกดปุ่ม connect ในฝั่ง client

บรรทัดที่89-92 ฟังก์ชัน button3send() ฟังก์ชันนี้ทำการส่งข้อความโดยการใช้ ฟังก์ชัน Send() ของคลาส socketObject

บรรทัดที่93-103 ฟังก์ชันonAccept()ฟังก์ชันนี้ถูกเรียกเมื่อมี event FD_ACCEPT

บรรทัดที่104-108 ฟังก์ชันonReceive()ฟังก์ชันนี้ถูกเรียกเมื่อมี event FD_READ



รูปที่ 3.8 ขั้นตอนการกดปุ่ม

ลำดับการทำงานของโปรแกรมทั้งสองฝั่ง ตามรูปที่ 3.8 มีขั้นตอนการทดสอบคือ

1.ด้าน server ให้กดปุ่มstart ปุ่มstartนี้ ทำให้ฟังก์ชัน button1start() ในบรรทัดที่68-79 ทำงาน ฟังก์ชันจะทำการ bind port และ listen แล้วจะแสดง dialog ว่า bind true ถ้าbindสำเร็จ ตามรูปที่ 3.10

2.ด้าน client กดปุ่มconnect ปุ่มconnectนี้ ทำให้ฟังก์ชัน button2connect() ในบรรทัดที่80-91 ทำงาน ฟังก์ชันจะทำการ connect ตอนนี้จะเกิดการเชื่อมต่อกันระหว่าง server กับ client ฟังก์ชันจะแสดง dialog ว่า connect true ถ้า connect สำเร็จ

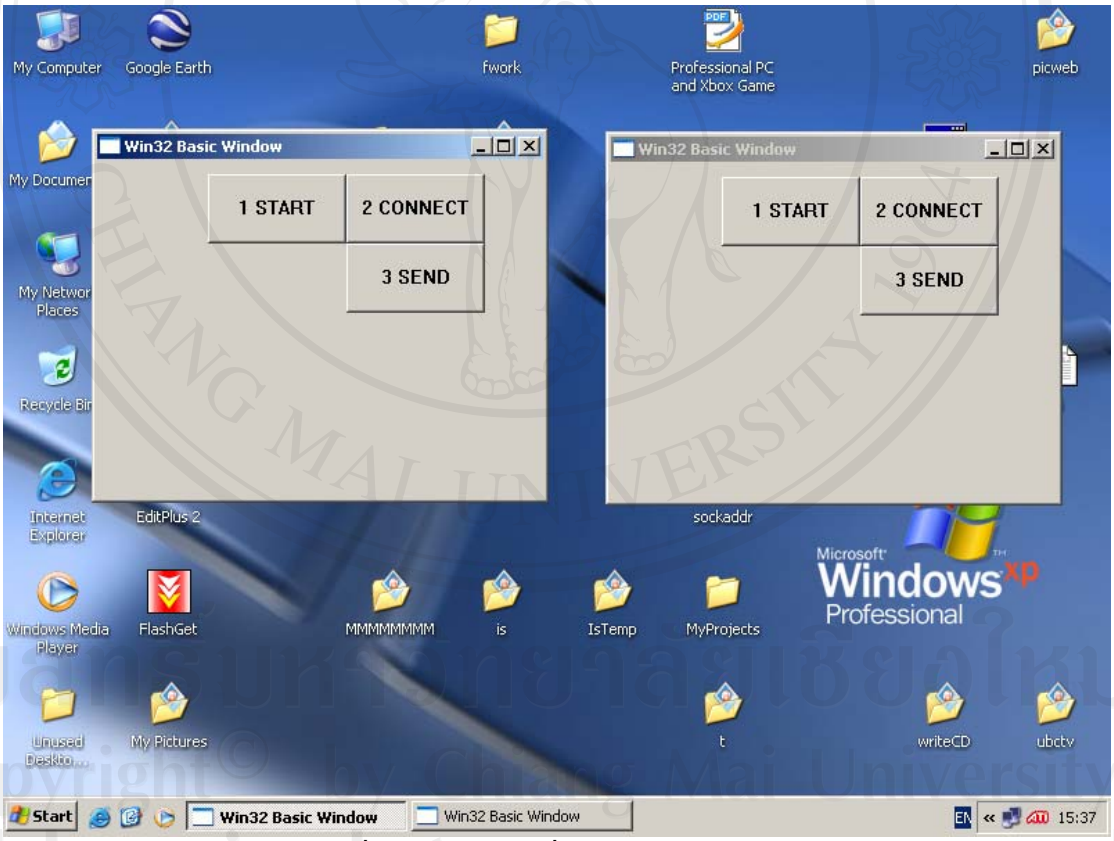
เมื่อเกิดการ connect โปรแกรมทางฝั่ง server จะเกิด event FD_ACCEPT ทำให้ฟังก์ชัน onAccept()ในบรรทัดที่96-106 ทำงาน จึงแสดง dialog ค่า ip ซึ่งเป็น ip ของเครื่องที่ connect เข้ามาคือเครื่อง client นั่นเอง การทดลองนี้ทดลองในเครื่องเดียว ip จึงเป็น 127.0.0.1 ตามรูปที่ 3.11

3.กดปุ่ม send ทางฝั่ง client ปุ่มsendนี้ ทำให้ฟังก์ชัน button3send () ในบรรทัดที่92-95 ทำงาน ฟังก์ชันนี้ทำการส่งข้อความไปให้ server

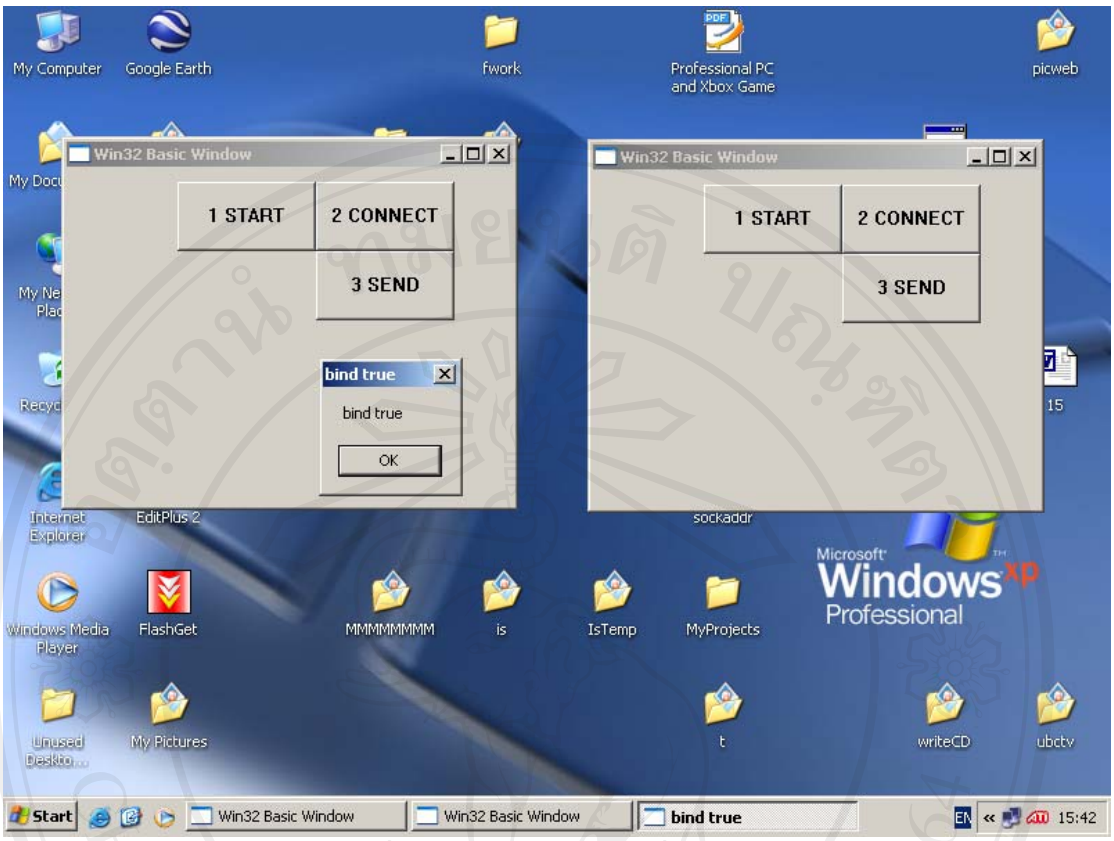
โปรแกรมฝั่ง server แสดง dialog ว่า TestData from Client เพราะว่าเกิด event FD_READ ซึ่ง event นี้จะทำการเรียกฟังก์ชัน onReceive() ในบรรทัดที่107-111

ตามรูปที่ 3.12

4.ปิดโปรแกรมฝั่ง client โปรแกรมฝั่ง client แสดง dialog ว่า FD_CLOSE เพราะว่าเกิด event FD_CLOSE ซึ่ง event จะเรียกให้ทำการแสดง dialog ว่า FD_CLOSE ในบรรทัดที่ 56 ผลลัพธ์ตามรูปที่ 3.13



รูปที่ 3.9 โปรแกรมฝั่ง server และ client



รูปที่ 3.10 กดปุ่ม start ทางฝั่ง server

รูปที่ 3.11 กดปุ่ม connect ทางฝั่ง client

รูปที่ 3.12 กดปุ่ม send ทางฝั่ง client

รูปที่ 3.13 ปิดโปรแกรมทางฝั่ง client

3.8 ตัวอย่างที่ 8 เกมค้นหาวัตถุ

มีวัตถุอยู่ในเกม 5 วัตถุ ตอนแรกวัตถุจะแสดงให้เห็นแต่เมื่อเริ่มเดินวัตถุจะหายไปแล้วให้ผู้เล่นค้นหาวัตถุ โดยการชนวัตถุจะทำให้วัตถุ ปรากฏขึ้น

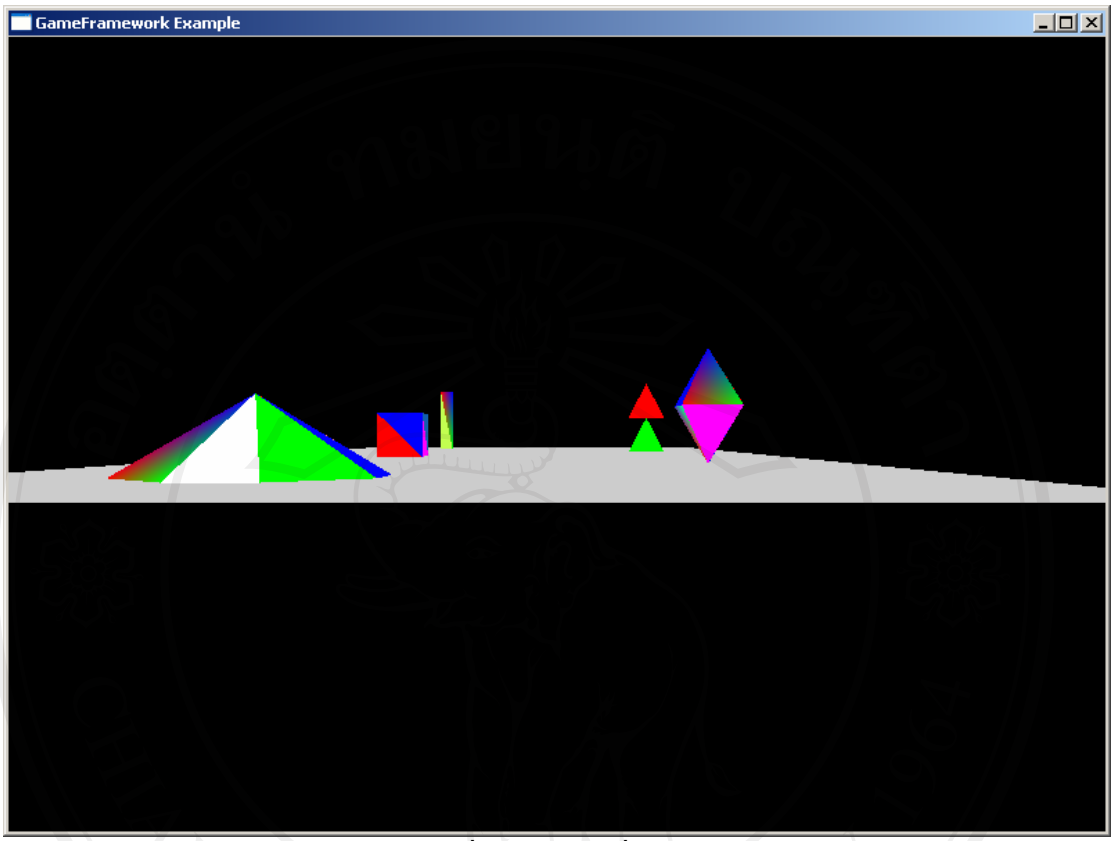
```
        if(k==keyUP) {
        c.cameraForward();
1.      if(bool1==false) bool1=c.hitObjectCylinder(gameObject1);
2.      if(bool2==false) bool2=c.hitObjectCylinder(gameObject2);
3.      if(bool3==false) bool3=c.hitObjectCylinder(gameObject3);
4.      if(bool4==false) bool4=c.hitObjectCylinder(gameObject4);
5.      if(bool5==false) bool5=c.hitObjectCylinder(gameObject5);
        r.clearScene();
        r.setCamera(c);
        r.setProjection();
6.      if(bool1) r.drawtoBackBuffer(gameObject1);
7.      if(bool2) r.drawtoBackBuffer(gameObject2);
8.      if(bool3) r.drawtoBackBuffer(gameObject3);
9.      if(bool4) r.drawtoBackBuffer(gameObject4);
10.     if(bool5) r.drawtoBackBuffer(gameObject5);
        r.drawtoBackBuffer(floor1);
        r.drawtoBackBuffer(floor2);
        r.present();
        sprintf( buffer, "E:rotateSpeed-  R:RotateSpeed+  A:speed-  S:speed+");
        r.drawText(hWnd, 0, 0, buffer, 4200000000, 0xFFFFFFFF, false);
        sprintf( buffer, "speed = %f , rotateSpeed = %f", c.speed, c.rotateSpeed);
        r.drawText(hWnd, 0, 20, buffer, 4200000000, 0xFFFFFFFF, false);
        }
```

คำอธิบาย ซอร์สโค้ด (Source Code)

บรรทัดที่1-5 เรียกฟังก์ชัน hitObjectCylinder ของคลาส camera เพื่อทดสอบการชน

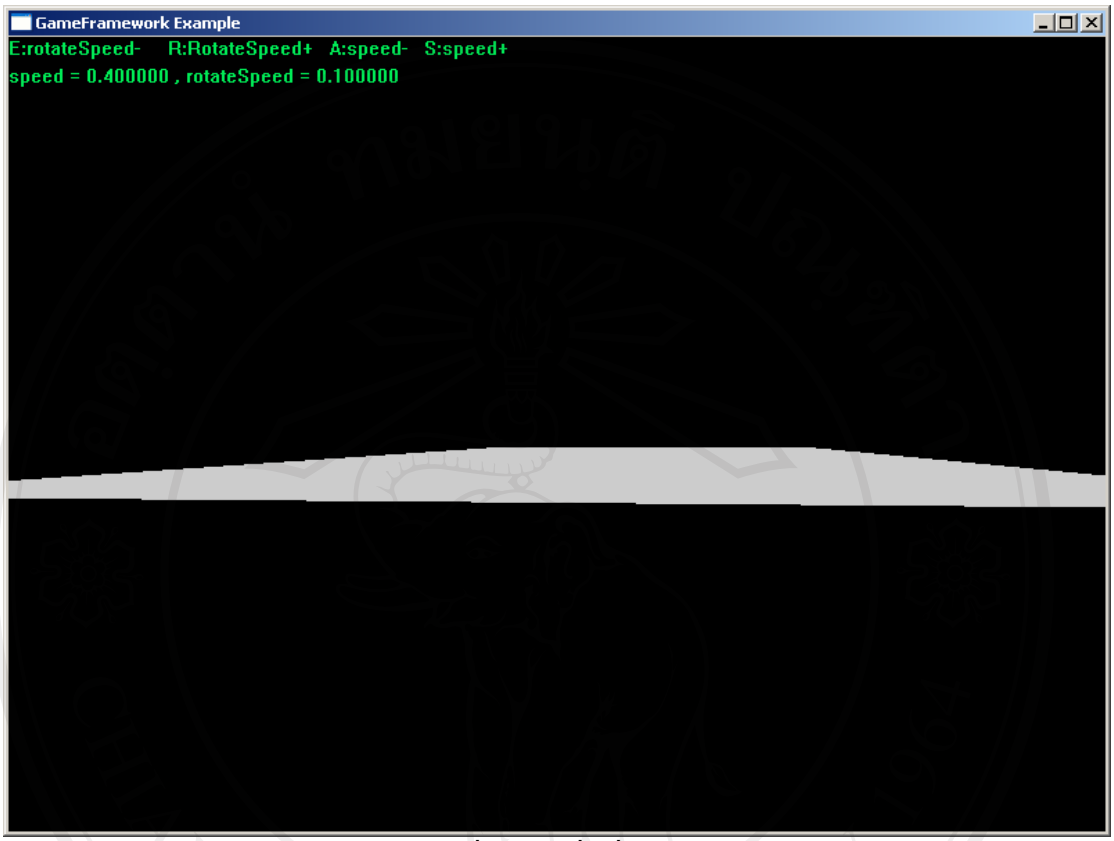
บรรทัดที่6-10 ถ้าตัวแปรบูลีนมีค่าเป็นจริงแสดงว่าเกิดการชนให้วาดรูปวัตถุ วัตถุก็จะปรากฏขึ้น

ก่อนเดินเป็นดังรูปที่ 3.14



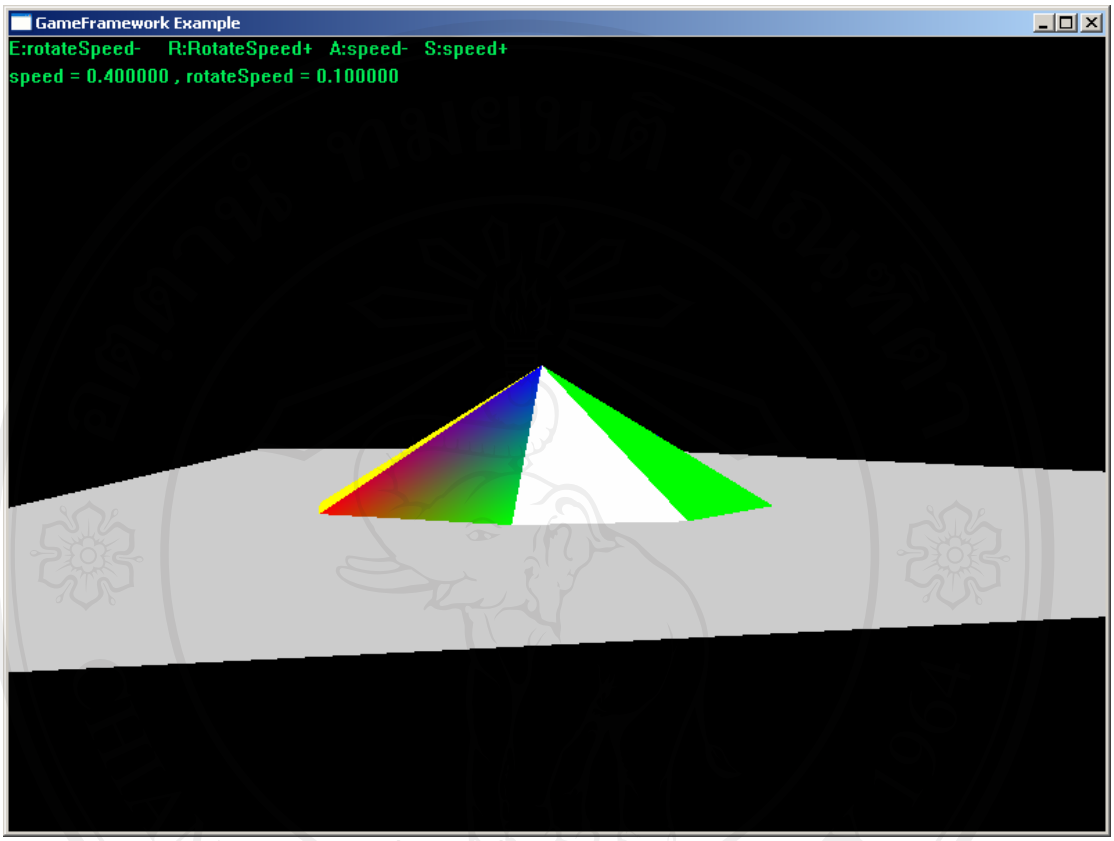
รูปที่ 3.14 ตอนเริ่มเกม

เริ่มเดินเป็นดังรูปที่ 3.15



รูปที่ 3.15 เมื่อเริ่มเดิน

เมื่อเดินชนวัตถุจะปรากฏขึ้นเป็นดังรูปที่ 3.16



รูปที่ 3.16 เมื่อเดินชนวัตถุจะปรากฏ