DOOM ENGINE MODIFICATION

(Compiling and Adding Free Look, Accurate Aiming and Third Person

View to the Zdoom Engine)

Yufan Lu, Bhumitra Nagar

lu.yuf@husky.neu.edu
nagar.b@husky.neu.edu

1. ABSTRACT:
In this paper, we present how to find the code points where we can modify the ZDoom engine to implement:

i. Free look
ii. Accurate aiming
iii. Third personal camera view

Actually, these features have already existed in the code, nevertheless, hidden deeply and you have to modify the existing code to get them to work. Also, we will take you through the challenges we faced during compilation of the ZDoom source code.

2. COMPILING ZDOOM SOURCE CODE

- Operating System: Windows 8.1
- IDE: Visual Studio 2013
- Download: http://zdoom.org/Download
- Required Libraries: Windows SDK for Windows 8.1, DirectX SDK, FMOD Ex
- ZDoom version: ZDoom 2.7.1

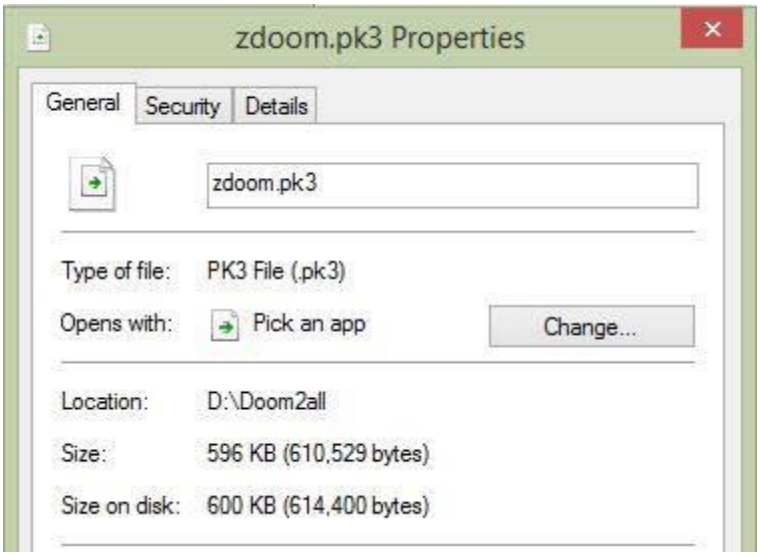
Steps to follow:

i. Install the required libraries listed above.

ii. Open the zdoom's project properties, add the FMOD's include and lib path into the path.

iii. Build the project.

iv. Make sure you have the zdoom.pk3 and DOOM2.wad, place them where the zdoomd.exe is.



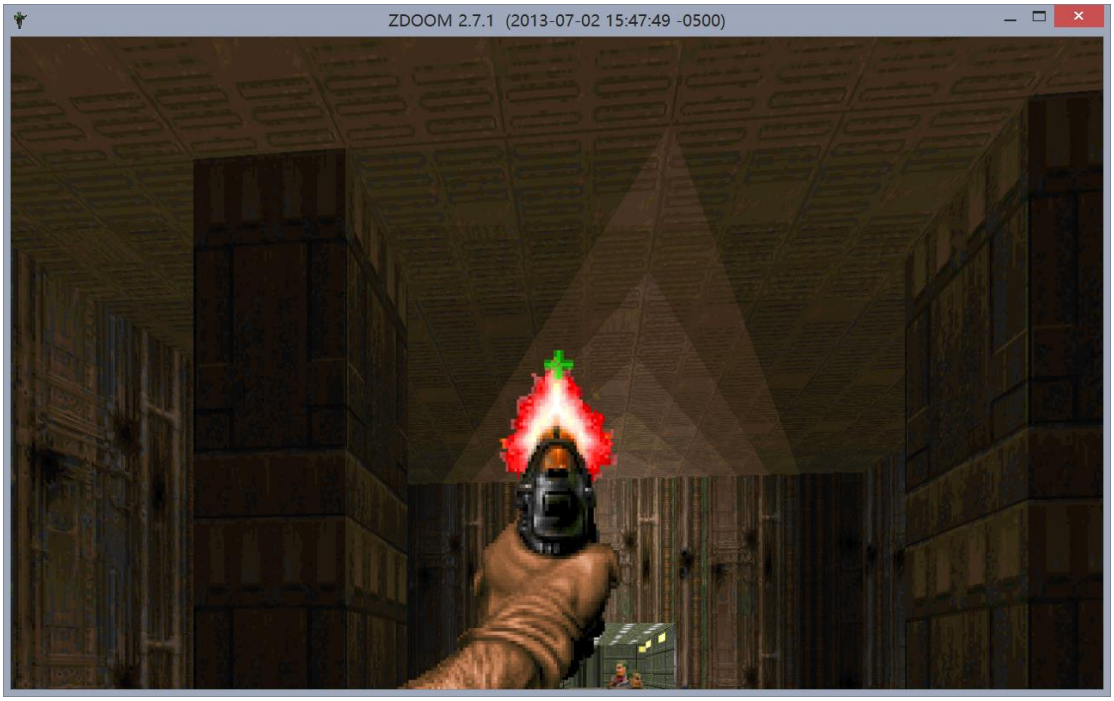
Double check the zdoom.pk3 is 596KB version, because when we run the game on another computer, we found that the generated zdoom.pk3 file size was 476KB, which resulted the failure of game execution

(you can use the one that comes with zdoom installation rather than the one that is generated after compilation).

v. Run the game.

3. FREE LOOK[1]

The original DOOM doesn't allow you to look up and down when you move your mouse up and down. Here, the character will be moving forward/backward instead of looking up/down.
Can we modify it to look up and down?



(Fig 1. Aiming at the ceiling)

Code snippet:

```
// D_main.cpp, line 322, D_PostEvent function
int look = int(ev->y * m_pitch * mouse_sensitivity * 16.0);
if (invertmouse)
    look = -look;
G_AddViewPitch (look);
events[eventhead].y = 0;
```

What this snippet does is:

i. Calculate the movement of mouse along in y axis.
ii. Invert it if the invertmouse is selected.
iii. Use the function G_AddViewPitch to pitch the camera
iv. Set the y movement to be zero, so the character won't be moving forward or backward.

4. ACCURATE AIMING[2]

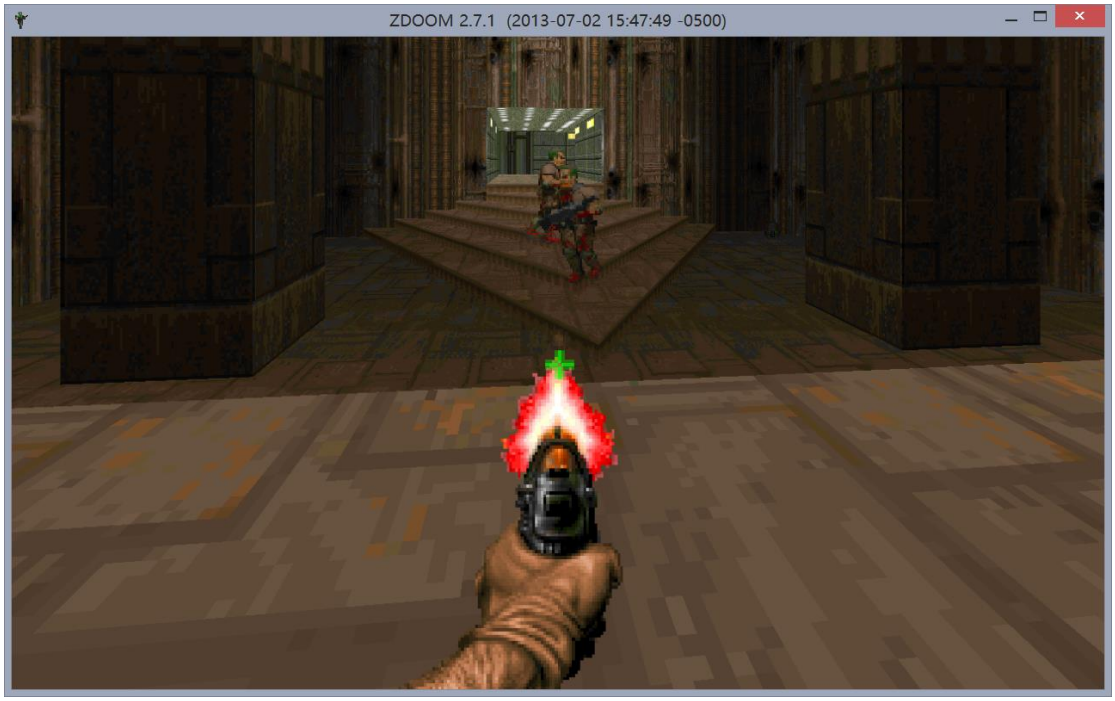
Now we have the free look, but in original DOOM, the auto aim mechanics exists, which will compensate the shortage that you can't look up and down. So we need to cancel this, and make it an accurate aiming.

(Fig 2. Shooting the position underneath the enemy but the enemy takes the damage)

Code snippets:

```
// p_map.cpp, line 3369, P_AimLineAttack function
if (t1->player != NULL)
{
    aim.shootz += FixedMul (t1->player->mo->AttackZOffset,
t1->player->crouchfactor);
}
else
{
    aim.shootz += 8*FRACUNIT;
}
// p_map.cpp, line 3405, P_AimLineAttack function
aim.toppitch = t1->pitch - vrange;
aim.bottompitch = t1->pitch + vrange;
```

This code snippets makes the shooting offset, thus we just comment them, and then the auto aiming is cancelled.



(Fig 3. Shooting underneath and the enemy does not take the damage)

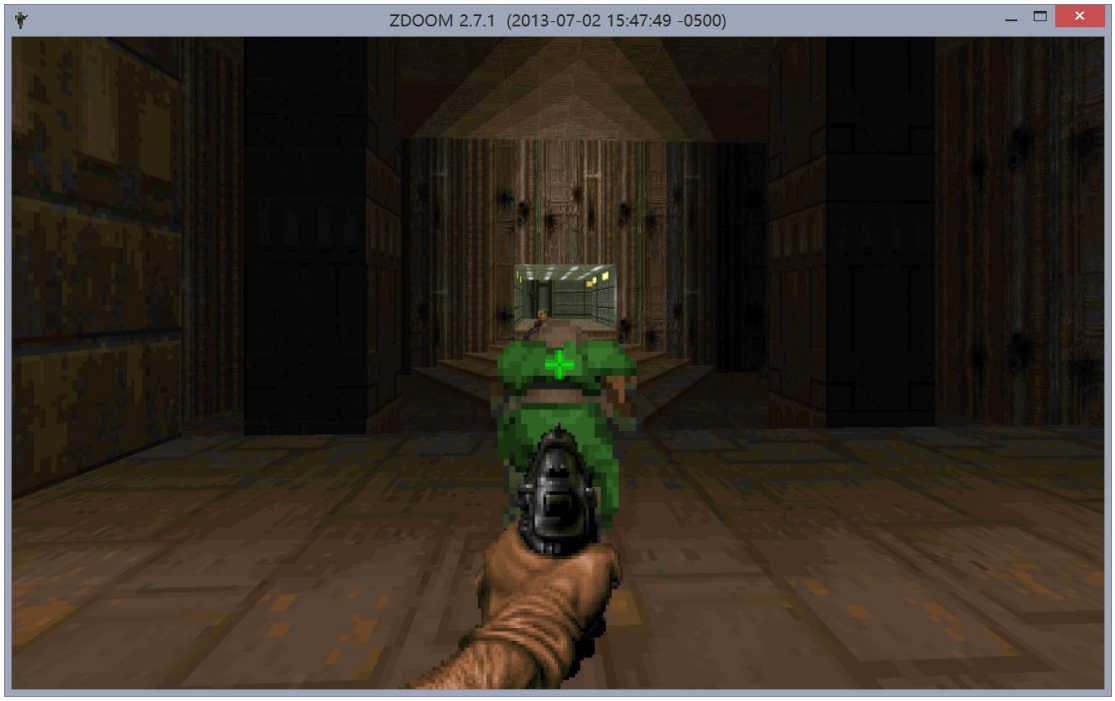
5. THIRD PERSON CAMERA[3]

Actually, the original DOOM has the third personal camera implemented as a cheating mode. So we'll dig out where these codes are.

```
// r_utility.cpp, line 776, R_SetupFrame function
if (player != NULL && gamestate != GS_TITLELEVEL &&
    ((player->cheats & CF_CHASECAM) || (r_deathcamera && camera->health
<= 0)))
{
    // [RH] Use chasecam view
    P_AimCamera (camera, iview->nviewx, iview->nviewy, iview->nviewz,
viewsector);
    r_showviewer = true;
}
// p_map.cpp, line 4124
void P_AimCamera (AActor *t1, fixed_t &CameraX, fixed_t &CameraY, fixed_t
&CameraZ, sector_t *&CameraSector)
{
    fixed_t distance = (fixed_t)(chase_dist * FRACUNIT);
    angle_t angle = (t1->angle - ANG180) >> ANGLETOFINESHIFT;
    angle_t pitch = (angle_t)(t1->pitch) >> ANGLETOFINESHIFT;
    FTraceResults trace;
    fixed_t vx, vy, vz, sz;

    vx = FixedMul (finecosine[pitch], finecosine[angle]);
```

```
    vy = FixedMul (finecosine[pitch], finesine[angle]);
    vz = finesine[pitch];

    sz = t1->z - t1->floorclip + t1->height + (fixed_t)(chase_height *
FRACUNIT);

    if (Trace (t1->x, t1->y, sz, t1->Sector,
        vx, vy, vz, distance, 0, 0, NULL, trace) &&
        trace.Distance > 10*FRACUNIT)
    {
        // Position camera slightly in front of hit thing
        fixed_t dist = trace.Distance - 5*FRACUNIT;
        CameraX = t1->x + FixedMul (vx, dist);
        CameraY = t1->y + FixedMul (vy, dist);
        CameraZ = sz + FixedMul (vz, dist);
    }
    else
    {
        CameraX = trace.X;
        CameraY = trace.Y;
        CameraZ = trace.Z;
    }
    CameraSector = trace.Sector;
}
```

In the first snippet, we found ***player->cheats & CF_CHASECAM***, it's to test if the current player's cheating status, if it's in CF_CHASECAM mode (chase camera), then execute the following code. That's it, the third personal camera mode. Then we make it true.



(Fig 4. Third person camera enabled but conflicting with first person cam)

That's cool. Wait, what's that hand with the gun? We need to correct this. We think it's the game developers who separated the drawing process, they drew the 3D space, then the hand decay, so we need to disable it.

```
// r_things.cpp, line 1166
void R_DrawPlayerSprites ()
{
    int         i;
    int         lightnum;
    pspdef_t*   psp;
    sector_t*   sec = NULL;
    static sector_t tempsec;
    int         floorlight, ceilinglight;
    F3DFloor *rover;
    if (!r_drawplayersprites ||
        !camera->player ||
        (players[consoleplayer].cheats & CF_CHASECAM))
        return;

    if(fixedlightlev < 0 && viewsector->e &&
viewsector->e->XFloor.lightlist.Size()) {
        for(i = viewsector->e->XFloor.lightlist.Size() - 1; i >= 0; i--)
            if(viewz <= viewsector->e->XFloor.lightlist[i].plane.Zat0()) {
                rover = viewsector->e->XFloor.lightlist[i].caster;
                if(rover) {
```

```
                    if(rover->flags & FF_DOUBLESHADOW && viewz <=
rover->bottom.plane->Zat0())
                        break;
                    sec = rover->model;
                    if(rover->flags & FF_FADEWALLS)
                        basecolormap = sec->ColorMap;
                    else
                        basecolormap =
viewsector->e->XFloor.lightlist[i].extra_colormap;
                }
                break;
            }
        if(!sec) {
// ……
```

We found:

```
if (!r_drawplayersprites ||
    !camera->player ||
    (players[consoleplayer].cheats & CF_CHASECAM))
    return;
```

If the player's cheating mode has the ***CF_CHASECAM*** flag turned on, it'll return the function directly. So we return the function directly. It's done.

(Fig 5. Third person camera enabled but conflicting with first person cam)

6. CONCLUSION

Finding the part where the code needs to be added in ZDoom engine's source code, is like finding a needle in a haystack. Our first step towards this was getting the doom source code compile successfully. The original Doom game didn't have the features of free look, accurate/manual aiming and a 3rd person view. The ZDoom engine provides us with the necessary structures to support these features. Modifying ZDoom is easy, provided you understand 'what is where' in the source code.

REFERENCES

[1] Zdoom wiki. 2012. "Free Look"
http://zdoom.org/wiki/Free_look
[2] Zdoom wiki. 2010. "Auto Aim"
http://zdoom.org/wiki/Autoaim
[3] Zdoom wiki. 2012. "Chasecam"
http://zdoom.org/wiki/Chasecam