●●●●●株式会社殿
向け報告書

株式会社フュージョンシス

文書バージョン	日付	変更点
Ver.1.0	2004 年 10 月 4 日	作成
Ver.	年月日	

目次

この文書は、●●●●株式会社殿からご依頼のあった件について、弊社において行った調査をまとめたものである。

調査期間： 2004 年 9 月 15 日～2004 年 10 月 3 日
作業者： 高橋、山崎、他アルバイト1名

画面上に Z 上の線が見える問題について

症状

動画再生時に Z 形の線が見える。特に動画において、シーンが切り替わるところに多い。

原因

ハイパースレッディングとビデオボードとの関係によって生じると考えられる。

対策

BIOS 画面においてハイパースレッディングをオフにする。

ticker のスクロールががたつく

症状

プライマリーのモニターおよびセコンダリのモニターにおいて ticker のスクロールが、がたつく。がたつくのは同時でないときもあり、片方だけの時もあり、また周期性があるともいえない。

原因

プログラムの描画処理が 33msec **毎、モニタの** VSYNC **が** 60fps **であった。モニタの** 2 **倍の周期は** 33.33msec **なのに、これを** 33msec **に丸めていたため、この誤差が蓄積し、一定期間ごとにずれとなって表れている。また、ポーリングによって描画のタイミングを得ていた。そのため、** pdprdr.exe **のプロセスの処理がかなり重くなり、ほかのプロセスやスレッドに、しかるべきタイミングで処理が割り当てられていない。**

プログラムの Present()**コマンドが、ビデオの** VSYNC **と非同期に実行されていたため、** VSYNC **で切り替わるタイミングの付近で** Present()**を行うと、前後のフレームに描画がずれてしまう。また、プライマリとセカンダリの** 2 **つの** VSYNC **は同期していないので、どちらか一方に合わせただけでは、もう一方でガタツキが生じてしまうという問題があった。また、プライマリとセカンダリの** 2 **つの** VSYNC **は周期が微妙にずれているという問題もあった。**

図1

全体的なイメージ。primary**モニタと**secondary**モニタが**refresh**レート**60Hz**で表示をしている。ただそのタイミングは同期しているとは限らないし、正確に**60Hz**ではない。**

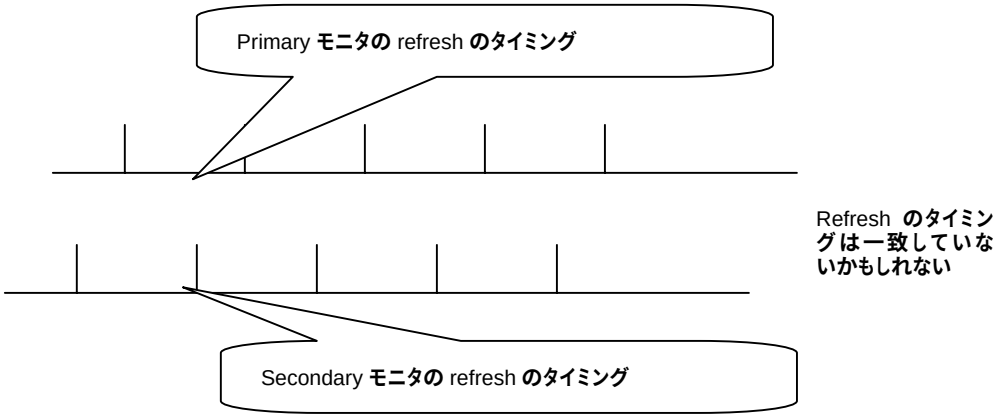


図2

プログラムがビデオボードのバッファに書き込む。プログラムの書き込みのタイミングと refresh **のタイミングの前後関係がいつも同じならば、がたつきは起こらない。**

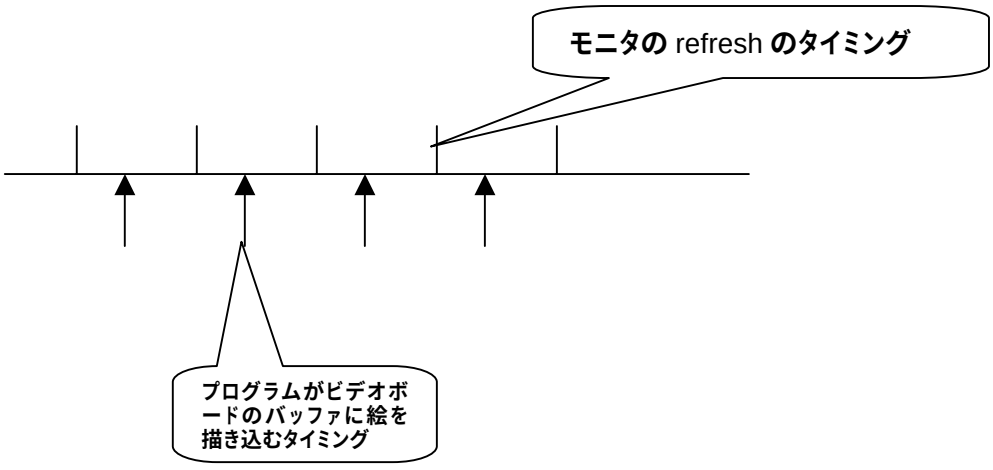


図3

何らかの原因でプログラムがビデオボードに絵を書き込むタイミングがずれたとする。赤で示したところは絵が変化しないところである。1／60秒のがたつきが生じることになる。

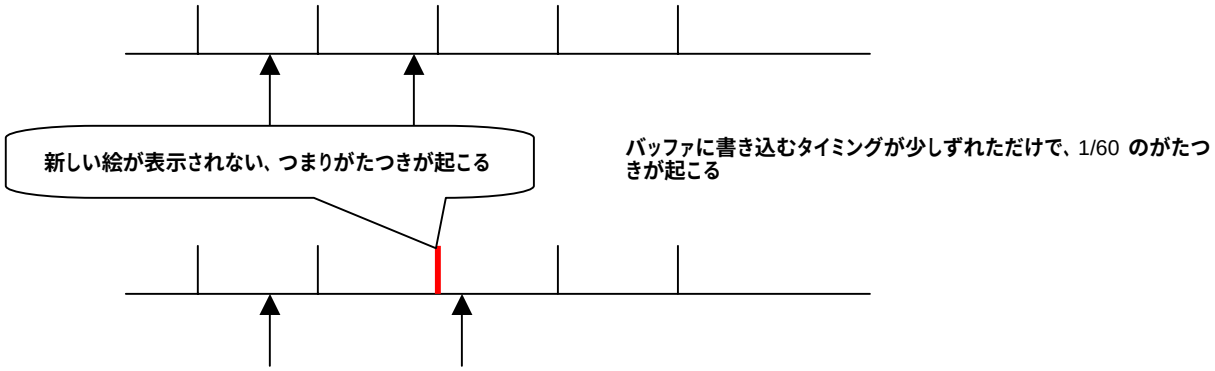
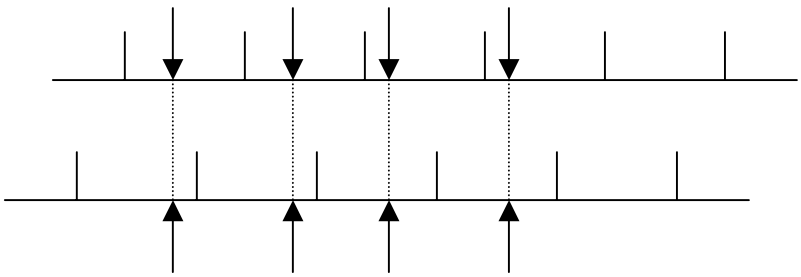


図4

プログラムは primary モニタ用のバッファと seconary モニタ用のバッファに＞＞同時に＜＜書き込んでいる。しかし、refresh レートは primary が 59.999Hz、secondary が 60.001Hz かもしれないし、同期もしていない。従ってプログラムが書き込むタイミングがたとえ一定でも、どちらかに、あるいは両方にがたつきが起こる。



対策

対策1：(9月30日)

プライマリ、セカンダリ、それぞれ独立に Present()を行うように変更。VSYNC の ScanLine を取得し、ずれに対する許容度の高い位置 (ScanLine=200) で Present()を呼び出すようにした。タイミン

グ調整は、ポーリングを使用せずに、正しい待ち時間を計算した上で、Sleep()を呼び出すようにした。

対策2：(9月28日)
・VSYNC に同期して描画処理を行うようにした。
・描画タイミングの待ち時間に、ほかのスレッドに CPU を与えるようにした。
・pdprdr.exe のプロセスの優先度を「高」にした。

■結果：
・数分程度の周期で表れる、大きなスクロールのがたつきがなくなった。
・数秒に一回の割合での定期的にがたつくことがなくなった。
　　これは、画面5（2分割(上コンテンツ/下コンテンツ/下スクロールエリア）が確認しやすい
・CPU 負荷が下がった（100% -> 50〜60%）
・スクロールのバックグランドの描画が有効になった。

■限定条件
・アダプタの数が2つであることを前提にコーディングしている。
　そのため、アダプタが1つの場合や、3つ以上の場合には、動作しない。

・30fps 用に各種定数が決定されているため、異なる fps では使用できない。
注意：

・VSYNC に同期するようにしたため、1つの VSYNC を取得し損なうと、完全に1フレームずれてしまう。したがって、他のプロセスの負荷が高くなると、がたつきが生じることがある。可能な限り、不要なサービスやアプリを止めることを推薦（アンチウィルスソフト、タスクマネージャなど）

■ソース修正（9月30日）
　pdprdrmain.cpp / dxapprication.cpp の2つ

[pdprdrmain.cpp]
1) プロセス優先度の変更
　　WinMain()中のウィンドウメッセージ処理ループの手前に一行追加

```
// タイマー精度を上げる
timeBeginPeriod(1);

// プロセスの優先度を「高」に
SetPriorityClass(GetCurrentProcess(), HIGH_PRIORITY_CLASS);
--------
```

[dxapprication.cpp]

2) DXApprication::FrameStep() 16msec **以下のスキップ処理をはずす**

```
--------
#ifdef NOSYNC_MODE
/*
        //long nowtime = timeGetTime();
        if( now <= m_next_strp) {
                return S_OK;
        }
        //m_next_strp = now + one_frame_draw;
        m_next_strp = now + 16;
*/
--------
```

3) DXApprication::Render() **以下のように修正**

```
--------
HRESULT DXApprication::Render(long now) throw()
{
        HRESULT hr;
    LPDIRECT3DSWAPCHAIN9 pSwapChain[2]    = {NULL,NULL};      //アダプタに関連付けれられたスワップチェーン
        LPDIRECT3DSURFACE9     pBackBuffer[2] = {NULL,NULL};   //バックバッファ

        if( !m_pd3dDevice) {
                hr = S_FALSE;
                return hr;
```

```
		}
		m_bDeviceLost = false;
		//long nowtime = timeGetTime();

	#ifdef NOSYNC_MODE
	/*	if( now <= m_next_draw) {
			return S_OK;
		}
		//m_next_draw = nowtime + tow_frame_draw;
		m_next_draw = now + 32;
	*/
	#else
		/*
		//m_sleep_val = 30;
		//Sleep(m_sleep_val);
				// スリープ (横線チェック用)
		*/
		void SetDrawMode(bool);

		if( m_last_draw!=0 && (now - m_last_draw)>tow_frame_draw){
			int revice = (int)(now - m_last_draw)/one_frame_draw;
			revice = revice;
			//LogWrite(LKL,LV0,"[ 補 正 ]:%ld(%ld:%ld)",revice,nowtime - m_last_draw, one_frame_draw);
			//ReviceVal(revice);		// スキップ値の設定
		}
	#endif
		//LogWrite(LKL,LV0,"@@@@@@@@@@ %d @@@@@@@@@",nowtime - m_last_draw);
		m_last_draw = now;

		DrawEnterCriticalSection();

		long st = timeGetTime();

		// アダプター別に処理を行う。
```

```
for(int i = 0;i<m_AdapterNum;i++){

        //アダプター毎にスワップチェーンからバックバッファをセットする。
        hr = m_pd3dDevice->GetSwapChain( i, &(pSwapChain[i]) );
        if( hr) {
            LogWrite(LKL,LV0,"      スワップチェーン
[Render]ERR=%ld,%ld,i=%d,m_AdapterNum=%d",hr,D3DERR_DEVICELOST,i,
m_AdapterNum);
            if( hr == D3DERR_DEVICELOST ) {
                LogWrite(LKL,LV0,"   デバイスロスと");
            }
            break;
        }
        hr = pSwapChain[i]->GetBackBuffer( 0,
D3DBACKBUFFER_TYPE_MONO, &(pBackBuffer[i]) );

        // レンダリングターゲットの設定
        hr = m_pd3dDevice->SetRenderTarget( 0, pBackBuffer[i] );


        // バックカラーを青に設定
        if( i==0) {
            hr = m_pd3dDevice->Clear( 0, NULL,
D3DCLEAR_TARGET|D3DCLEAR_ZBUFFER, D3DCOLOR_XRGB(100,100,100), 1.0f, 0 );
            if(!SUCCEEDED( hr =
m_pd3dDevice->BeginScene() ) ){
                // シーン描画失敗
                break;
            }
        } else {
            hr = m_pd3dDevice->Clear( 0, NULL,
D3DCLEAR_TARGET|D3DCLEAR_ZBUFFER, D3DCOLOR_XRGB(100,100,100), 1.0f, 0 );
            //hr = m_pd3dDevice->Clear( 0, NULL,
D3DCLEAR_TARGET|D3DCLEAR_ZBUFFER, D3DCOLOR_XRGB(0,0,255), 1.0f, 0);
            if(!SUCCEEDED( hr =
m_pd3dDevice->BeginScene() ) ){
```

```
                                        // シーン描画失敗
                                        break;
                                }
                        }

                        // アダプター毎のシーン描画処理
                        if( FAILED(hr = RenderSide(i, pSwapChain[i])) {
                                // 描画エラー
                        }

                        // FPS 描画（アダプタ 1 にのみ描画)
                        if( i==0 && m_FpsDraw) {
                                fps.Draw();
                        }

                        // 描画終了
                        hr = m_pd3dDevice->EndScene();
                        if( D3DERR_DEVICELOST == hr ) {
                                m_bDeviceLost = true;
                        }



                        // リソース開放
//                      SAFE_RELEASE( pBackBuffer[i] );
//                      SAFE_RELEASE( pSwapChain[i] );
                }
                //long et = timeGetTime();
                //if( et-st> 10) {
                        //LogWrite(LKL,LV0,"  待ち=%d", et-st);
                //}

                DrawLeaveCriticalSection();

#ifdef FPS_RATEFIX
                // ------------------------------------------------------------
```

```
		//	FPS を画面のリフレッシュレートにあわせる WAIT
		//
		D3DRASTER_STATUS rs;
		do{
			if(D3D_OK!=m_pd3dDevice->GetRasterStatus(1,&rs)){
				rs.InVBlank = TRUE;
			}
			Sleep(0);
		}while(!rs.InVBlank);
	#endif

		// VSYNC に同期した Present 処理
		{
	#define	PRSCLINE	200		// Present するべき ScanLine
	#define	MAXSCLINE	776		// ScanLine の最大値
	#define	VTIME		(16.67-2)	// 垂直の有効期間[msec]

			static UINT	tLastP0 =0;	// 前回のプライマリの Present 時刻
			static UINT	tLastP1 =0; // 前回のセカンダリの Present 時刻
			UINT	tNow = timeGetTime();

			D3DRASTER_STATUS rs0, rs1;
			m_pd3dDevice->GetRasterStatus(0,&rs0);
			m_pd3dDevice->GetRasterStatus(1,&rs1);

			// Wait タイムをそれぞれ求める
			int		tWait0=0;
			if(rs0.InVBlank)
	tWait0=(int)((PRSCLINE+60/2)*VTIME/MAXSCLINE);
			else		if(rs0.ScanLine<PRSCLINE)
	tWait0=(int)((PRSCLINE-rs0.ScanLine)*VTIME/MAXSCLINE);
			else
	tWait0 = (int)((MAXSCLINE+60+PRSCLINE-rs0.ScanLine)*VTIME/MAXSCLINE);
			if(tNow+tWait0-tLastP0<8)	tWait0	=
33+tLastP0-tNow;
```

```
                    else if(tNow+tWait0-tLastP0<25)     tWait0 += 17;       //      25=
16.6*3/2    17=16.666

                    int                 tWait1=0;
                    if(rs1.InVBlank)
        tWait1=(int)((PRSCLINE+60/2)*VTIME/MAXSCLINE);
                    else                                if(rs1.ScanLine<PRSCLINE)
        tWait1=(int)((PRSCLINE-rs1.ScanLine)*VTIME/MAXSCLINE);
                    else
        tWait1 = (int)((MAXSCLINE+60+PRSCLINE-rs1.ScanLine)*VTIME/MAXSCLINE);
                    if(tNow+tWait1-tLastP1<2)           tWait1 = 33+tLastP1-tNow;
                    else if(tNow+tWait1-tLastP1<25)     tWait1 += 17;       //      25=
16.6*3/2    17=16.666


                    // どちらを先に Present するか判定
                    if(tWait0<=tWait1){
                            if(tWait0>0)        Sleep(tWait0);
                            pSwapChain[0]->Present(NULL, NULL, NULL, NULL ,
D3DPRESENT_DONOTWAIT);
                            tLastP0 = timeGetTime();
                            int ttt = tWait1-(tLastP0-tNow); // 上の Present でまれに
時間がかかるため
                            if(ttt>0)   Sleep(ttt);
                            pSwapChain[1]->Present(NULL, NULL, NULL, NULL ,
D3DPRESENT_DONOTWAIT);
                            tLastP1 = timeGetTime();
                    }
                    else {
                            if(tWait1>0)        Sleep(tWait1);
                            pSwapChain[1]->Present(NULL, NULL, NULL, NULL ,
D3DPRESENT_DONOTWAIT);
                            tLastP1 = timeGetTime();
                            int ttt = tWait0-(tLastP1-tNow); // 上の Present でまれに
時間がかかるため
                            if(ttt>0)   Sleep(ttt);
```

```
                                pSwapChain[0]->Present(NULL, NULL, NULL, NULL ,
D3DPRESENT_DONOTWAIT);
                                tLastP0 = timeGetTime();
                        }
                }


                // リソース開放
                SAFE_RELEASE( pBackBuffer[0] );
                SAFE_RELEASE( pSwapChain[0] );
                SAFE_RELEASE( pBackBuffer[1] );
                SAFE_RELEASE( pSwapChain[1] );



                // ------------------------------------------------------------
                //バックバッファからプライマリバッファへ転送(全画面同時転送)
                //
        /*
                if(FAILED(hr = m_pd3dDevice->Present( NULL, NULL, NULL, NULL ))) {
                        //アダプタごとにプレゼントパラメータリセット
                        for(int i=0;i<m_AdapterNum;i++) {
                                m_pd3dDevice->Reset(&m_param[i]);
                        }

                }
        */

                RenderEnd();
                return hr;
        }
        --------
```

■変更ポイント（9月28日）

・プロジェクト-設定

リンカの設定に ddraw.lib を追加

（WaitForVerticalBlank()関数を利用するため）

■ソース修正（9月28日）

pdprdrmain.cpp / dxapprication.cpp の2つ

- pdprdrmain.cpp WinMain()中のウィンドウメッセージ処理ループを下記のように変更

```
// ウィンドウメッセージ処理ループ
//        WaitForVerticalBlank()による VSYNC 同期
//        ddraw.lib を追加のこと！
LPDIRECTDRAW lpDD;                                          // IDirectDraw のポインタ
DirectDrawCreate(NULL, &lpDD, NULL);                        // DIRECTDRAW オブジェクト作成
SetPriorityClass(GetCurrentProcess(), HIGH_PRIORITY_CLASS);  // プロセスの優先度を「高」に
long lastTime = timeGetTime();
while (1) {
    if( PeekMessage( &msg, NULL, 0, 0, PM_NOREMOVE)) {
        if( 0 == GetMessage(&msg, NULL, 0, 0)) {
            break;
        }
        // キーアクセラレーターを使わない場合不要
        //if( 0 == TranslateAccelerator( App.hWnd, hAccel,&msg)) {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        //}
    } else {
        // 同期（ lastTime から 33.33msec 毎に描画 ）
        long period = lastTime+33-timeGetTime()-3;  // 定数値 3 は誤差のため
        if(period>0)        Sleep(period);
```

```
                    lpDD->WaitForVerticalBlank(DDWAITVB_BLOCKBEGIN,0);
//VSYNC 待ち
                    lastTime = timeGetTime();

                    // ウィンドウが表示されている場合のみ
                    // レンダリング処理
                    WINDOWPLACEMENT wndpl;
                    GetWindowPlacement(App.hWnd, &wndpl);  // ウインドウの状態
を取得
                    if((wndpl.showCmd != SW_HIDE)
                                && (wndpl.showCmd != SW_MINIMIZE)
                                && (wndpl.showCmd != SW_SHOWMINIMIZED)
                                && (wndpl.showCmd != SW_SHOWMINNOACTIVE)  )
                    {
                        long nowtime = timeGetTime();

                        // -- アクション計算
                        App.FrameStep(nowtime);

                        // -- 2 面描画
                        App.Render(nowtime);
                    }
                }

                // OS に制御を明け渡すためのスリープ
                //Sleep(1);
            }
            // DirectDraw-object free
            lpDD->Release();
```

- dxapprication.cpp DXApprication::FrameStep(long now)
NOSYNC_MODE **の後の** 6 **行をコメントアウト**

```
#ifdef NOSYNC_MODE
	//long nowtime = timeGetTime();
	if( now <= m_next_strp) {
		return S_OK;
	}
	//m_next_strp = now + one_frame_draw;
	m_next_strp = now + 16;

#endif
```

vvvvvvvvvvvvvvvvvvvvvvvvvvvv

```
#ifdef NOSYNC_MODE
/*
	//long nowtime = timeGetTime();
	if( now <= m_next_strp) {
		return S_OK;
	}
	//m_next_strp = now + one_frame_draw;
	m_next_strp = now + 16;
*/
```

- dxapprication.cpp DXApprication::Render(long now)
NOSYNC_MODE **の後の** 5 **行をコメントアウト**

```
	m_bDeviceLost = false;
	//long nowtime = timeGetTime();

#ifdef NOSYNC_MODE
	if( now <= m_next_draw) {
		return S_OK;
	}
```

```
		//m_next_draw = nowtime + tow_frame_draw;
		m_next_draw = now + 32;
#else

vvvvvvvvvvvvvvvvvvvvvvvvvvvv

		m_bDeviceLost = false;
		//long nowtime = timeGetTime();

#ifdef NOSYNC_MODE
/*
		if( now <= m_next_draw) {
			return S_OK;
		}
		//m_next_draw = nowtime + tow_frame_draw;
		m_next_draw = now + 32;
*/
#else
```

60fps で動かしたとき、セコンダリのモニターに tearing が入る

症状

原因

おそらく refresh のタイミングと Present 関数との関係。すなわち ticker のスクローリングにがたつきが入るのと同じ原因。

対策

ticker **のスクロールのがたつきのタイミングの修正方法と同じでよいと考えられる。ただし** ticker **の弊社修正プログラムは** 30fps **にしか対応していない**（60fps **だと計算時間が足りないので** 30fps **でとった方法は** 60fps **には用いることが可能ではない**）

その他わかっていること

宇多田のtraveling**という**mpeg**ファイルがいくつか使われているが、すべて**30fps**でエンコードされている。従って**60fps**のプログラムを使っても**60fps**の画質で再生されることはない。**

ビデオドライバについて
今年8月25日にビデオボードの新しいドライバが出ているようなので、試しにそれをインストールした（復元ポイントを設定し、元のドライバに戻せるようにしながら）。症状は 60fps **の** secondary **モニタでは、**tearing **よりさらにひどい画面の乱れが出た。ただ画面の乱れは、**MPEG **の動画が再生されている部分のみで見られた。というわけで** 60fps、tearing **の原因としてきわめて怪しいのは** MPEG **のデコーダと考えられる。この作業終了後、もとのドライバに戻した。**

CuMain**を立ち上げてから**Pdprdr**が立ち上がるあたりで、時々**OS**自体が落ちてリブートが始まる（正常終了でないので、ブート時にディスクチェックが入る）。弊社作業中に5回起こった。**

プライマリとセコンダリのモニタの描画を完全に別スレッド化することによって、いろいろな問題が解決する可能性がある。

メモリの確保の仕方で改良した方がよいと思われる箇所やメモリの解放がされていない箇所がある。

以上