**THE PERFORMANCE TIME**

Phần này chúng ta sẽ nghiêm cứu khái quát về việc quản lý thời gian trong 1 chương trình game, các hàm thông dụng liên quan tới nó. Đây là một khái niệm rất quan trọng vì nó sẽ ảnh hưởng rất nhiều tới các hoạt động trong game.

Đầu tiên ta sẽ biết rằng các hàm quản lý game sẽ nằm trong thư viện window cho nên ta phải include header file <Windows.h>

chúng ta sẽ quan tâm tới một hàm mà nó trả về 1 đơn vị thời gian trong một thời điểm được gọi là count (một đơn vị để miêu tả thời gian đã trôi qua), **QueryPerformanceCounter(…)** bên trong hàm sẽ có 1 thông số là giá trị mà nó trả về dưới dạng \_int64. Ta coi thử cú pháp của nó

\_\_int64 prevTimeStamp = 0;

QueryPerformanceCounter((LARGE\_INTEGER\*)&prevTimeStamp);

Để tính toán thời gian làm việc giữa 2 điểm khác nhau, ta đơn giản gọi nó 2 lần ở 2 thời điểm, sau đó trừ ra ta sẽ tính được số cout (thời gian) mà chương trình đã trải qua.

\_\_int64 A = 0;

QueryPerformanceCounter((LARGE\_INTEGER\*)&A);

/\* Do work \*/

\_\_int64 B = 0;

QueryPerformanceCounter((LARGE\_INTEGER\*)&B);

Theo ví dụ như trên thì thời gian trôi qua là B – A

Tuy nhiên có vẻ không có lợi cho chúng ta khi tính toán với, bởi vì ta cần xử lý giá trị nào đó liên quan tới second, chứ không phải là count, do đó ta phải tìm cách chuyển số count này sang second. May mắn thay chúng ta cũng có được một hàm đáp ứng được nhu cầu đó **QueryPerformanceFrequency**. Ta hãy coi thử cú pháp của nó

\_\_int64 cntsPerSec = 0;

QueryPerformanceFrequency((LARGE\_INTEGER\*)&cntsPerSec);

QueryPerformanceFrequency sẽ trả về số lượng count trong 1 second, từ giá trị đó ta có thể chuyển tính số second trong 1 count.

float secsPerCnt = 1.0f / (float)cntsPerSec;

Có số second trong 1 count, ta có thể chuyển số count mà ta có từ hàm QueryPerformenceCounter sang số second.

Trên đây là một chút giới thiệu về các hàm ta sử dụng. Chúng ta sẽ đi vào vấn đề quan trọng trong việc sử dụng các hàm này cho chương trình của mình.

**Time Differential Between Frames**

Tính toán thời gian giữa các Frame rất đơn giản, ta chỉ việc tính ra số count giữa 2 frame rồi từ đó chuyển nó sang đơn vị second. (Lưu ý do việc phải chuyển đổi từ count sang second nên ta phải sử dụng hàm **QueryPerformanceFrequency** để tính số second trong 1 count).

Ta coi thử đoạn code thực thi

int D3DApp::run()

{

MSG msg;

msg.message = WM\_NULL;

**\_\_int64 cntsPerSec = 0;**

**QueryPerformanceFrequency((LARGE\_INTEGER\*)&cntsPerSec);**

**float secsPerCnt = 1.0f / (float)cntsPerSec;**

**\_\_int64 prevTimeStamp = 0;**

**QueryPerformanceCounter((LARGE\_INTEGER\*)&prevTimeStamp);**

while(msg.message != WM\_QUIT)

{

// If there are Window messages then process them.

if(PeekMessage( &msg, 0, 0, 0, PM\_REMOVE ))

{

TranslateMessage( &msg );

DispatchMessage( &msg );

}

// Otherwise, do animation/game stuff.

else

{

if( mAppPaused )

{

Sleep(20);

continue;

}

if( !isDeviceLost() )

{

**\_\_int64 currTimeStamp = 0;**

**QueryPerformanceCounter((LARGE\_INTEGER\*)&currTimeStamp);**

**float dt = (currTimeStamp - prevTimeStamp)\*secsPerCnt;**

updateScene(**dt**);

drawScene();

**// Prepare for next iteration: The current time**

**// stamp becomes the previous time stamp for the**

**// next iteration.**

**prevTimeStamp = currTimeStamp;**

}

}

}

return (int)msg.wParam;

}

Lưu ý ở ví dụ trên ta tính số second trong giữa 2 Frame, ta hoàn toàn có thể tính số milisecond trong 2 Frame (chính xác hơn nhiều).  
  
**Frame Per Second (FPS) Caculation**

Cách tính FPS là ta tính số lượng Frame trong 1s, ta có 1 biến numFrames tính số lượng Frame xảy ra. Như vậy mỗi lần Update ta tăng số Frame này lên, đồng thời ta dùng 1 biến timeElapsed để đếm thời gian đã trôi qua, biến này sẽ được cộng dồn giá trị khoảng thời gian giữa 2 Frame. Cho tới khi timeElapsed vượt qua 1.0f, tức là nó đã qua 1s. ta sẽ xuất giá trị Frame mà ta đếm được ở numFrames. Như vậy ta có số FPS. Lưu ý là khi đã qua 1s, ta khởi gán lại các giá trị timeElapsed và numFrames về 0 để tính FPS tiếp theo.

void GfxStats::update(float dt)

{

// Make static so that their values persist across

// function calls.

static float numFrames = 0.0f;

static float timeElapsed = 0.0f;

// Increment the frame count.

numFrames += 1.0f;

// Accumulate how much time has passed.

timeElapsed += dt;

// Has one second passed?--we compute the frame statistics once

// per second. Note that the time between frames can vary, so

// these stats are averages over a second.

if( timeElapsed >= 1.0f )

{

// Frames Per Second = numFrames / timeElapsed,

// but timeElapsed approx. equals 1.0, so

// frames per second = numFrames.

mFPS = numFrames;

// Average time, in milliseconds, it took to render a

// single frame.

mMilliSecPerFrame = 1000.0f / mFPS;

// Reset time counter and frame count to prepare

// for computing the average stats over the next second.

timeElapsed = 0.0f;

numFrames = 0.0f;

}

}

Trong đoạn code trên ta thấy có thêm phép tính số milisecond trong 1 Frame. Thực ra đối với một người choi game, họ chỉ quan tâm đến FPS. Tuy nhiên số milisecond trong 1 Frame sẽ giúp ít rất nhiều cho lập trình viên thao tác. Giả sử ta co FPS 1000 tức là 1ms render 1 Frame, nếu số FPS đó giảm xuống 250 tức là 4s ta render 1 Frame. Ta thấy rõ sự tăng lên thời gian của 1 Frame là 3s, sử dụng tính toán này cho chương trình của chúng ta sẽ tốt hơn sử dụng FPS để xử lý (như vậy FPS để kiểm tra tốc độ, sức mạnh của máy để thực thi game của chúng ta mà thôi).