

Re: Using sprites en masse and fluctuating frame rates in C# MDX

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.directx.managed/2004-09/00>

From: ZMan (*news-replies_at_thezbuffer*)

Date: 09/07/04

Date: Mon, 6 Sep 2004 22:44:46 -0700

1. Do you just see a single Gen 2 collection or is it repeatedly happening – this isn't good if its repeated. Probably means you are running short on memory becuase you are taking big chunks for textures or arrays or becuase you are creating object in your render loop which you are not disposing. For example run any of the directx samples – non of them do much but I don't see gen 1 or 2 collections happening too often. See Toms blog here

<http://weblogs.asp.net/tmiller/archive/2003/11/14/57531.aspx> for how easy it is to create new objects which will never go away untill you close the app.

Try this (you may lose automatic resize or anything else dependant on the event hooks but see if it improves things)

```
Device.IsUsingEventHandlers = false;
```

One really obvious sign of you messing this up is if your app seems to take a long time to quit – its calling dispose on the thousands of objects hooked to the device.

3. I didn't realise you had your own – I was referring to the one DX will set up for you. Ty this:

```
[PresentationParams].PresentationInterval = PresentInterval.Immediate;
```

when you set up your device. With your frame limiter commented out this should run flat out (you can leave DoEvents in there otherwise you may not be able to quit) – you may notice some flickering or tearing but it will at least help us see if there is a fundamental performance problem in your code.

Sorry I didn't have much time to look at your code in detail – maybe tomorrow night.

"charlie d" <charlied@discussions.microsoft.com> wrote in message news:5D25CC1C-1846-4405-908B-B60A17B5881C@microsoft.com...

> 1) Gen 0 seems to happen in initialization, and Gen 2 a couple seconds
> later.

> The second long slowdowns seem to have fixed themselves as I clean some
> things up... damned if I know exactly what did it.

>

> 2) I mentioned drawing the background multiple times just as an experiment
> -- it's just being drawn once.

```
>
> 3) FPS limiting... I've been curious about this. If I take off my frame
> limiter, (the kind of thing that says don't render unless elapsedTicks >
> 1000.0/60.0 ) my framerate stays at approximately 60.0. Is some DoEvents
> some how conspiring to keep it so the app only gets attention 60 times a
> second? But then I take out DoEvents all together -- so I'm in a
> completely
> selfish loop -- and the same frame rates (and the same hiccups every
> second.)
>
> Also, these hiccups to seem to be more than just one 60fps frame's worth.
> They happen maybe twice a second now that I take a second look.
>
> Here's some of my code.
>
> The main loop in Main():
> while(frm.Created)
> {
> frm.Loop();
> Application.DoEvents();
> }
>
> Loop():
> public void Loop()
> {
> switch(mode)
> {
> case JiezouMode.NormalMode:
>
> frameTimer.Update();
>
> //if ((frameTimer.getTime() - lastFrame) > (1000.0/60.0))
> //{
> playMode.Update();
> playMode.Render();
> lastFrame = frameTimer.getTime();
> if (playMode.quit) mode = JiezouMode.Quit;
> //}
> break;
> (I'm only running NormalMode for now)
> I can comment out playMode.Update() and still get the hiccups. As you
> can
> see i've commented out my frame limiter. Still getting 60fps hiccuping...
> and without update(), new notes aren't being generated to scroll by.
> Keyboard input isn't being gathered. No sound is playing. So we're just
> rendering the background and some stationary sprites. !
>
> public void Render()
> {
> if (device == null) return;
> if (noteActorList != null)
```

```

> for (int i = 0; i < noteActorList.Count; i++)
> {
> NoteActor na = (NoteActor)noteActorList[i];
> na.calculateSprites();
>
>
> }
> for (int i = 0; i < 8; i++)
> {
> AnimSprite a = (AnimSprite)beatSpriteList[i];
> a.calculate();
> }
>
> id.Scale(1.0f,1.0f,1.0f);
> device.Clear(ClearFlags.Target, System.Drawing.Color.White, 1.0f, 0);
> device.BeginScene();
>
> spr.Begin(SpriteFlags.AlphaBlend);
> bgSprite.draw(spr);
> if (started)
> {
> for (int i = 0; i < 8; i++)
> {
> AnimSprite a = (AnimSprite)beatSpriteList[i];
> a.draw(spr);
> }
>
> for(int i = 0; i < noteActorList.Count; i++)
> {
> NoteActor actor = (NoteActor)noteActorList[i];
> actor.draw(spr); // was device
> }
> }
> spr.Flush();
>
> spr.End();
> // DEBUG
> double speed = ((double)StaffEnd - (double)StaffStart) / ( 8.0 * (60.0 /
> (double)tempo)) / 1000.0;
> float mvmt = (float)(speed * timer.getDeltaTime());
> drawingFont.DrawText(null, timer.getFPS().ToString("F5") + " " +
> keyboard.keysPressed + " " + speed.ToString("F5") + " " +
> mvmt.ToString("F5")
> + " " + output + " getChar:" + keyboard.getChar().ToString() + "
> keysPressed:" + keyboard.keysPressed
> , new Point(10,10), Color.Blue);
> // END DEBUG
> device.EndScene();
> device.Present();
>
> }

```

```
> }
>
>
>
> "ZMan" wrote:
>
>> 20 sprites plus a background shoulnd't tax anything (in fact I would
>> think
>> the reference rasterizer could probably keep up!).
>>
>> For the GC checking – try running perfmon at the same time and watching
>> when
>> then GC happens – do they line up with the sudden drop in frame rate. Are
>> you seeing Gen 2 runs? Gen 0 should be qucik enough not to show up, but
>> gen
>> 2 takes longer.
>>
>> The fact that you say you are running at 60fps makes me think you are
>> sync'd
>> to the refresh rate on your screen. Ttry changing this so you are
>> presenting
>> your screen as often as you can – you may get some tearing but it will
>> give
>> you a better idea. If seen some jumpyness due to this becuase eventually
>> one
>> frame takes a little longer and you have to wait for 2 screen refreshes
>> to
>> see it.
>>
>> You also shouldn't have to calculate matrices in advance – most game
>> engines
>> are calculating all manner of transformations in real time for animation
>> so
>> again your 20 sprites are not a big deal.
>>
>> I assume sprite.Render is your code which is calling sprite.Draw2D or
>> sprite.draw – what happens if you comment out indiviual lines inside
>> there – is there any one of them which also makes the frame rate stay
>> consistent. Maybe you can post this bit of code.
>>
>> Why do you draw your background 5 times (not that this should be causing
>> the
>> problem)?
>>
>> Zman
>>
>> "charlie d" <charlie.d@discussions.microsoft.com> wrote in message
>> news:0BD646FA-FA36-43CD-AAE7-5E0B6B1C51B6@microsoft.com...
>> > Hello all,
>> >
>> > I am developing an interactive music application that involves notes
```

>>> *sprites*
>>> *flying about and precisely timed keyboard input. Since I'd rather*
>>> *focus*
>>> *on*
>>> *the theory and the music rather than the programming, I figured C# and*
>>> *MDX*
>>> *would be a good choice. Things are going reasonably well considering*
>>> *the*
>>> *mystery swirling about MDX and managed languages... here's the issue.*
>>>
>>> *It runs at 60 fps with little hiccups about every second. But about*
>>> *every*
>>> *twenty seconds there is one second of around 40 fps. I've profiled my*
>>> *timer*
>>> *and have graphs to prove this. This is consistent. 60 fps... and then*
>>> *a*
>>> *moment of choppiness. Back to 60 fps.... choppiness.*
>>>
>>> *About my software: Currently there is a background with note sprites*
>>> *going*
>>> *from right to left as if music is scrolling by. I create the note*
>>> *sprites*
>>> *dynamically so there are never more than 20 alive at any one time.*
>>>
>>> *Here's where things get mysterious: I'm using Sprite for my rendering!*
>>> *Sprite.Draw with the Vector3 overloads (much more consistent acting*
>>> *than*
>>> *the*
>>> *Vector2 ones btw) I calculate() my matrices before beginning the scene*
>>> *and*
>>> *then draw things with one application-wide Sprite that is created at*
>>> *the*
>>> *beginning of the app and disposed at the end. (Not created every scene,*
>>> *in*
>>> *other words -- which created terrible choppiness.) Each sprite in turn*
>>> *has*
>>> *its own transformation matrix which it pops into*
>>> *mySceneSprite.Transform.*
>>> *The background is a sprite like this. Seems somewhat silly to repaint*
>>> *the*
>>> *background sprite every frame.. ?*
>>>
>>> *Other info: Renderloop is the doEvents() type. I've used CLRProfiler*
>>> *and*
>>> *nothing's jumped out at me. It's not garbage collection (at least not*
>>> *the*
>>> *kind CLRP highlights.) Also I'm in China and working off of a Toshiba*
>>> *convertible tablet M200 with a nvidia GF5200 portable video card.*
>>>
>>> *My own investigations: Taking out the backgroundSprite.draw in the*
>>> *render*

>> > *loop takes away this ~40fps fluctuations. Drawing the background 5*
>> > *times*
>> > *a*
>> > *frame causes massive slowdowns which surprised me. Still can't get rid*
>> > *of*
>> > *the hiccups no matter what I do. It's like one frame that takes 100ms*
>> > *instead of 16.66ms or something.*
>> >
>> > *I'm thinking I should give up Sprite and make some quads proper. I*
>> > *don't*
>> > *know if anyone on this Earth has tried to use Sprite in a practical*
>> > *manner*
>> > *as*
>> > *I have... I gave it my best, Bill! I'll definitely write up my*
>> > *experiences*
>> > *in more detail for the sake of documentation. Hardly quirkless... but*
>> > *it*
>> > *seems to hang together OK except for this irksome framerate issue.*
>> >
>> > *Whatcha'll think ??*
>>
>>
>>