Hello Captains,

Dit wordt een lang verhaal en voor sommigen waarschijnlijk niet erg spannend. Maar als je geïnteresseerd bent hoe de prestaties van je flightsim-pc worden beïnvloed, moet je dit lezen. Het kan zijn dat veel van de dingen die je dacht te begrijpen over prestaties, niet langer waar zijn. Tenminste, niet meer sinds we zijn overgestapt op het x64-platform van Prepar3D v4.
Voordat je verder leest, weet dan dat het niet de bedoeling is om een technische discussie aan te gaan over hoe processors werken, hoe rendering-engines werken, of om de kern van de Prepar3D-engine te ontleden. Wij vereenvoudigen een paar zaken om het onderwerp voor iedereen begrijpelijk te houden en natuurlijk is niet alles hier een waarheid als een koe....

In plaats daarvan is dit een algemene illustratie van waar we zijn begonnen, hoe we hier zijn gekomen, wat we hebben en hoe dit je simprestaties dagelijks beïnvloedt. Op een aantal punten heeft deze discussie ten doel om een aantal lang gevestigde misverstanden, (die hardnekkig herhaald worden door Simmers) te corrigeren. Hopelijk leer je met deze wetenschap hoe zaken te evalueren en leer je waardoor je sim minder goed of slecht presteert dan je verwacht.

De afgelopen jaren heb je op onze website al diverse publicaties kunnen lezen over performance van de pc's tijdens het FSX-tijdperk. Ook deze publicaties hadden ten doel om je bewust te maken van de beperkingen van zowel de flightsim software als ook het (toen nog 32-bits) operating system.
We zetten ze nog even op een rij ... en misschien is het leuk dat je nog eens terugblikt. Wij publiceerden eerder in:

Januari 2010:	game-performance-is-slower-than-expected
	flight-simulator-x-game-performance-is-slower-than-expected
Januari 2011:	affinity-what-it-is-and-how-to-configure
Februari 2012:	tweaking-fsx-cfg
Maart 2014:	the-sense-and-nonsense-of-fps

Recentelijk werd de PMDG737NGXu geïntroduceerd en naar aanleiding daarvan schreef Robert R. Randazzo van PMDG weer eens in zijn gebruikelijke klare taal hoe het heden ten dage zit met de invloeden op de performance van onze flightsim-pc.
Wij vonden het de moete waard om deze - toch wel droge -"tech-talk"- te vertalen` en hier en daar op te leuken, zodat je na het lezen in ieder geval weet waar je op moet letten, wanneer je jouw sim te lijf gaat om de laatste druppel performance eruit te persen... 😉

Klaar? Daar gaat ie ... hoe zat het ook alweer?

In 2005 werd PMDG uitgenodigd voor een bijeenkomst in Seattle om te leren over een nieuwe productlijn die eraan kwam. Deze productlijn zou een codedropping van FSX zijn en zou de industriële simulatiemarkt bedienen. Het stond bekend als Microsoft ESP **[1].** We spoelen een aantal jaren snel vooruit en Microsoft besloot om de simulatie-activiteiten te verlaten, maar op

weg naar de buitendeur gaven ze de ESP-licentie aan een klein bedrijf waar weinig Simmers toen ooit van hadden gehoord (Lockheed Martin). Deze zetten de groei en ontwikkeling van ESP voort onder hun eigen merknaam, "Prepar3D."
FSE Leesvoer [1]
https://docs.microsoft.com/en-us/previous-versions/microsoft-esp/ff798293(v=msdn.10)

Prepar3D zit nu in de 4e generatie. Het is een ander en superieur platform en in basis in vergelijking met de originele Microsoft ESP. Het heeft echter veel grotere mogelijkheden en heeft ons stapsgewijs een enorm verbeterde visuele ervaring opgeleverd. De toevoeging van bijvoorbeeld HDR-verlichting en reflecterende oppervlakken heeft bijvoorbeeld het uiterlijk van de simulatie dramatisch verbeterd. Deze evolutionaire verbetering is goed geweest omdat het voldoet aan de eisen van Simmers voor een betere visuele kwaliteit. Echter het heeft er ook voor gezorgd dat de extra aandacht niet is aangewend op plaatsen waar het waarschijnlijk net zo goed nodig was: de core-simulatie-engine en de rendering-engine.

De capaciteit van de hardware die we gebruiken, is in hetzelfde decennium exponentieel gestegen. De mogelijkheden van moderne grafische software engines bij gebruik van moderne hardware zijn gewoon verbazingwekkend. Kijk maar eens naar de grafische resultaten van Microsoft Flight Simulator 2020.

De Prepar3D-engine is NIET zo snel vooruitgegaan en dit geeft ons een sim-gebruik waarvan we allemaal weten dat het wordt belemmerd door processor-afhankelijkheid, gebrek aan GPU-gebruik, het onvermogen om zich over vele CPU-cores te verspreiden, enz. Met andere woorden, we draaien software die inherent is aan een architectuur uit de jaren '90 op hardware die eigenlijk wil dat de software profiteert van 20 jaar evolutie in software- en rendering- ontwerp. Echter ons huidige platform kan dat gewoon niet aan.

Rekening houdend met het feit dat onze simulator een inherent inefficiënte architectuur is voor moderne hardware, zijn er en paar zaken aan toegevoegd die echter wel en zeer efficiënt werken. We noemen hier HDR-rendering, Physics Based Rendering (PBR), Volumetrische verlichting en Re-shading. Kort gezegd, het zou onmogelijk zijn geweest om deze functies tien jaar geleden aan de Prepar3D-simulatie-engine toe te voegen, omdat de hardware onvoldoende reservecapaciteit had om alles te verwerken. Met de exponentiële groei van hardware-mogelijkheden in combinatie met enkele efficiëntieverbeteringen in de core-P3D-engine, hebben de ontwikkelaars ons deze mogelijkheden kunnen bieden en zoals eerder gezegd, verbeteren deze de visuele kwaliteit van de sim.

De belangrijkste taken in een simulatie-engine is de zogenaamde "draw-call" **[2]**. Simpel gezegd ... een "draw-call" is een vraag van de CPU aan de GPU om iets te tekenen. Het probleem is dat het voorbereiden van "draw-call" een groot deel van je CPU-tijd en energie steelt. De CPU moet uw scène-inhoud omzetten in een indeling die uw GPU begrijpt. En een heel groot gedeelte van dit proces is het instellen van de juiste render-parameters, zoals texturen, shaders, meshes, enz ...

Zie een "draw-call" dus als een functie die iets visueels in de scène aanpast, zodat het kan worden weergegeven en naar het scherm (lees GPU) kan worden verzonden zodat je het kunt zien en daarmee ook de simulatie-engine op gang houdt. Bedenk wel dat de scène die je op het display ziet, veel lagen heeft en dat elk van deze lagen wordt gemaakt door een draw-call.

FSE Leesvoer [2] CPU-performance/draw-call-optimization

Enkele voorbeelden van "draw-calls" zijn:
De vormen in de scène zijn bijvoorbeeld een draw-call.
De structuren die op die vormen worden toegepast, zijn een draw-call.
Het licht dat op oppervlakken speelt, is een draw-call.
De schaduwen die door de vormen worden veroorzaakt, zijn een draw-call.
De reflectie van licht op oppervlakken is een draw-call.
De manier waarop licht over afstand vervaagt, is een draw-call.
De locatie van uw muisaanwijzer bovenaan de scène is een draw-call.
De kleur van uw muisaanwijzer op de scène is een draw-call.

Bij elk grafisch gestuurd softwareproduct krijg je een bepaald aantal draw-calls om de scène op het display te krijgen. Met moderne software-engines worden deze draw-calls afgehandeld op de GPU, waardoor de hoofdprocessor (CPU) de tijd vrij krijgt om zijn eigen werk te doen.
In de FSX-engine werd dit echter allemaal gedaan door de CPU (single-core/dual-core/quad-core) en daarom benoemen we FSX als processor-beperkt. In de x86 (32-bits) FSX-dagen waren er twee primaire makers van "overtollige draw calls". Deze hadden betrekking op het renderen van de muiscursorpositie en op textuur gebaseerde verlichting.

Dit is de reden waarom producten, zoals de PMDG 747 die heel veel muisklikgebieden gebruikten, zo slecht presteerden. Het is ook de reden waarom bijna iedereen een boost in prestaties kreeg in deze 747 toen Dovetail dit probleem al (naar ik meen) in 2014 aanpakte.

Vooruitgaand naar de huidige versie van Prepar3D, zijn de belangrijkste makers van overtollige draw calls HDR-verlichting, PBR (dynamische verlichting, dynamische reflecties) en volumetrische verlichting. Tot overmaat van ramp lijkt het effect exponentieel, omdat de scènes zo vaak opnieuw moet worden gerenderd, dat alle effecten van kleurverschuivingen door HDR, reflecties, volumetrische verlichting, enz. daarin zijn begrepen.

Nu, zodra alle draw-calls zijn voltooid, dan heeft de sim nog zijn eigen inefficiëntie in de schemering/ zonsopgang. Immers de hele scène moet opnieuw worden gerenderd om rekening te houden met kleurverschuivingen in verband met de veranderende lichtomstandigheden;
Dit activeert het gehele draw calls proces opnieuw, om de manier waarop de verschillende lichtbronnen met elkaar omgaan, herhaaldelijk te berekenen om zo de scène weer te geven.

De meeste moderne games laten dit proces afwerken op de GPU, maar Prepar3D doet dit niet volledig. Een groot deel van het weergaveproces blijft bij de processor (CPU). In plaats van complexe weergaveprocessen over te dragen aan de GPU, wat uiteindelijk de taak van de GPU is, doet de CPU al het bijbehorende werk om de scène weer te geven. Aangezien de meeste taken worden gedaan op de processor, betekent dit dat de processor de bottleneck is. Als de processor, bij overtollige draw call-oproepen die door alle nieuwe effecten worden geactiveerd, dit niet aankan, zullen de processen vertragen en dus de performance afnemen.

Dus dit brengt me tot het gebruik van NGXu als een specifiek voorbeeld van hoe al dit samenspel prestaties vermindert:

PMDG heeft hier twee machines met heel verschillende mogelijkheden.
De eerste machine [M1] is een I7 6700K met een Nvidia 980-kaart.

De tweede machine [M2] is een I9 9900K met een Nvidia 2800Ti.
Op deze machines worden de prestaties van de PMDG 747-8 (PBR op extern model, maar niet in VC, textuur-gebaseerde verlichting) en de PMDG 737NGxu (PBR extern en intern met volumetrische verlichting in de VC) vergelijken. Dit is een redelijk goede vergelijking omdat beide vliegtuigen hetzelfde aantal cockpitdisplays hebben.

De tabel met resultaten:
Scenes: [1] mid-day, [2] schemering en [3] nacht
AC= PMDG aircraft
M1= I7 6700K + GTX980
M2= I9 9900K + GTX2080Ti
PD= Performance Delta

	AC	M1	M2	PD	AC	M1	M2	PD
[1] mid-day	747-8	34	62	82%	NGXu	30	53	76%
[2] schemering	747-8	21	52	147%	NGXu	15	47	213%
[3] nacht	747-8	30	58	93%	NGXu	25	50	100%

Het doel van deze vergelijking is om je de impact te laten zien van overtollige draw-calls op hardware die niet de capaciteit heeft om het werk te absorberen. Je kunt direct naar de regel "NGXu-schemering" gaan en de prestatieresultaten op de twee machines vergelijken. Het prestatieverschil in dit scenario is dramatisch en veel groter dan het vergelijken van de prestaties van NGXu en 747 in andere identieke scenario's.

Waarom is dit zo?
De NGXu heeft een aantal functies die de 747 momenteel niet heeft: volumetrische verlichting en een met PBR behandelde cockpit. Deze twee functies creëren een groot aantal draw-calls. Terwijl de software werkt om de cockpitdisplays en de EFB's in kleur te corrigeren, (ondanks de effecten van Prepare3D's vreselijk geïmplementeerde HDR-lightbloom-effect), worden ook extra draw-calls gemaakt ten behoeve van de kleurcorrectie die moet gelden voor de impact van volumetrische verlichting op alle oppervlakken/ displays/ efb's in de NGXu-cockpit.

Een scène weergegeven met het NGXu-cockpit, bij zonsondergang/ zonsopgang met volumetrische verlichting en de EFB die aanwezig is op een Prepar3D-platform, vereist waarschijnlijk 60% meer 'draw-calls' dan de 747 in een identieke scène, simpelweg vanwege deze nieuwe functies.

Op de mid-range hardware presteert de 747-8 prima in alle verlichtingsconfiguraties en -omstandigheden, maar NGXu komt in dezelfde scènes met veel hogere draw-call-nummers vanwege alle extra rendering-processen die moeten worden verwerkt om de interactie van schemering/ dageraad, volumetrische verlichting, dynamische verlichting, PBR-behandeling, reflectiviteit, HDR-kleurcorrectie, enz. tot stand te brengen
Deze extra verwerkings- en draw-call-vereisten verslinden de verwerkingscapaciteit van de i7-6700K en veroorzaken een verslechtering van de prestaties. Op de i9-9900K absorbeert de niet gebruikte CPU-capaciteit de extra draw-calls.
Conclusie: de processor is de bottleneck.

Dit brengt ons op een belangrijk punt, waarvan ik denk dat iedereen een moment moet nemen om dit te verteren. Jarenlang vertelden simontwikkelaars je (inclusief, tot tenminste 2010, je trouwe dienaar en PMDG) dat de reden dat je sim tot stilstand kwam toen je een complexe add-on in een complexe omgeving uitvoerde, was omdat "we zoveel functies bevatten in je vliegtuig. " Dit was echt niet de hele waarheid, hoewel we dat in de meeste van onze gevallen destijds niet wisten. Men zou kunnen beweren dat dit al zo was in de 32-bit FSX-dagen, maar dat is zeker niet waar in het genre van het x64-simulatieplatform.

Wij horen vaak gebruikers zeggen "die add-on werkt langzaam", omdat het zoveel simuleert". Niets is minder waar en voor iedereen die begrijpt hoe software wordt gecompileerd en omgezet in machinecode, en hoe die code door een moderne processor wordt uitgevoerd, weet dat elke ontwikkelaar die zegt: "onze dingen lopen langzaam omdat het zo complex is" niet weet wat er aan de hand is. Als een moderne add-on die in een moderne simulator wordt uitgevoerd, langzaam werkt, is dit hoogstwaarschijnlijk omdat het product een groot aantal draw-calls genereert en dus de simulator deze allemaal probeert op processorniveau af te handelen, in plaats van op de GPU.

M.a.w., dus als je merkt dat deze software-mythologie wordt voortgezet, weet dan dat het afval is. Als je iemand ziet zeggen "oh, dit product is efficiënter omdat het beter werkt", weet dan dat dit ook afval is. Kijk in plaats daarvan naar wat het product wel en niet gebruikt in termen van de belangrijkste functies van het Prepar3D-platform. Als het slechts enkele van de hierboven besproken functies gebruikt, zal het betere prestaties leveren dan wanneer het allemaal gebruikt. Er is gewoon geen weg omheen.

Laat je dus niet misleiden door marketing slogans, hypes en regelrechte waarheden. Excessieve aantallen draw-calls zijn de dood voor de performance. Het is de taak van ontwikkelaars als PMDG om de maximale prestaties uit de simulator te proberen te halen, terwijl ze het platform ook niet laden met draw-calls, want dat is uiteindelijk wat je vertraagt. Het is niet gemakkelijk en het vereist een behendige besluitvorming als we dingen aanpassen/ afstemmen/ verplaatsen om de kleine marginale gebieden die we op ontwikkelingsgebied kunnen beïnvloeden, te beheersen.

Om dit alles samen te vatten: NGXu wil echt goede hardware als je goede prestaties wilt zien tijdens schemering/ zonsopgang en overgangsperioden met slecht zicht. Als de EFB aanwezig is, verlichting is ingeschakeld, HDR actief is, een schemering benadert bij weinig zicht, dan creëert het product een groot aantal draw-calls boven wat normaal nodig is om elk frame binnen Prepar3D weer te geven. De NGXu is een van de eerste producten waarin de modellen oorspronkelijk zijn ontwikkeld door Prepar3D en specifiek voor het Prepar3D v4.5-platform. Andere producten zijn beschikbaar die een mix van volumetrische verlichting en PBR gebruiken, maar geen enkele in deze mate - dus op die manier lopen we voorop in de volledige opname van functies op dit product.

Om de prestaties te verbeteren, is er een direct verband tussen de mogelijkheden van de processor, de mogelijkheden van de GPU en de prestaties die je zult zien. (Er zijn een aantal factoren met betrekking tot de GPU genegeerd waardoor je onbedoeld denkt dat het onbelangrijk is ... het IS nog steeds belangrijk, maar op een ander gebied dat dit argument niet zoveel beïnvloedt ...

Als onderdeel van ons normale ontwikkelingsproces blijven we dingen aanpassen, aanpassen en aanpassen om te zien welke voordelen we kunnen bieden aan gebruikers met prestatieproblemen. De implementatie van de verlichting aanpassen, PBR verwijderen van oppervlakken die niet profiteren van de mogelijkheden, zoeken naar efficiëntie in reflecterende oppervlakken, enz. Het doel is om op veel plaatsen een klein beetje efficiëntie te vinden om een grotere boost in prestaties te bieden zonder verslechtering van de kwaliteit van het product voor de zeer veel gebruikers met geavanceerde hardware die geen problemen ondervinden.

We hebben gehoord dat gebruikers die de EFB's opbergen een aanzienlijke prestatieverbetering zien, wat logisch is omdat u het aantal weergegeven schermen op het cockpitdek met 20% vermindert, waardoor het aantal draw-calls met betrekking tot kleuraanpassing voor HDR wordt verminderd en dynamische verlichting, enz. Andere gebruikers hebben geconstateerd dat het eenvoudig uitschakelen van HDR tijdens de overgang van zonsondergang naar zonsopgang een positieve impact heeft.

Zoals zowat alles in vliegen en computergebruik, zullen er altijd een aantal afwegingen zijn, en hoe je ze wilt benaderen zal voor elke gebruiker verschillend zijn. (Bijvoorbeeld: ik denk dat Prepar3D's HDR de sim belachelijk maakt vanuit de VC- dus ik schakel het uit. Ik vind het echter geweldig in externe weergaven- dus als ik het externe model ga laten zien, draai ik het terug op ... Andere gebruikers voelen zich helemaal het tegenovergestelde. Voor elk wat wils, zoals ze zeggen!)

Als je zo ver hebt gelezen, bedankt. Dit is lang, saai en niet erg vermakelijk, maar ik hoop dat je een paar dingen hebt geleerd die je zullen helpen begrijpen hoe je een impact kunt hebben op je sim door slimme keuzes te maken die ondersteunen wat belangrijk voor je is. Aan onze kant blijven we, als onderdeel van de gebruikelijke ontwikkelaar-klant sociale compact, op zoek naar efficiëntie en pleiten we voor prestaties door de ontwikkelaars te vragen manieren te vinden om moderne technieken te gebruiken om het volledige potentieel van het simulatieplatform te ontsluiten.

Bij Microsoft hebben ze dit duidelijk gedaan met hun nieuwe simulatieplatform en we kijken er erg naar uit om te zien wat ermee gedaan kan worden om onze producten echt te laten neuriën en zingen.

Het werk met Prepar3D gaat door.

Robert R Randazzo

Bewerkt door Jan Vaane
FSE Weekly Update 2020-51
FS Eindhoven
8-2-2020