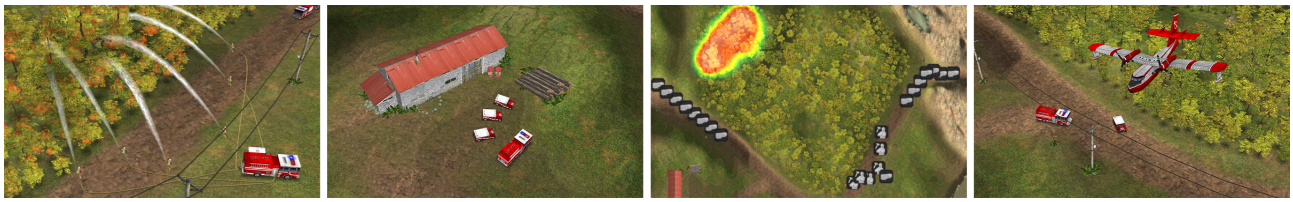
Excel@FIT 2015
http://excel.fit.vutbr.cz

Optimalizace zobrazování komplexních scén na mobilních zařízeních s využitím enginu Unity

Michal Matýšek*



Abstrakt
Tento článek popisuje optimalizační postupy, které umožňují, pomocí enginu Unity, efektivně vykreslovat složité herní scény na mobilních zařízeních. Proces zobrazování komplexního prostředí vytvořeného pomocí prezentovaných technik je na mobilních telefonech i tabletech několikanásobně efektivnější, oproti klasickým přístupům využívajícím dostupné komponenty a funkce enginu Unity. Výsledné scény vytvářejí testovací prostředí pro měření, analýzu a porovnávání efektivity různých optimalizačních metod.

Klíčová slova: Optimalizace — mobilní zařízení — zobrazování — engine Unity — hry

Přiložené materiály: Demonstrační videa

*xmatys03@stud.fit.vutbr.cz, *Fakulta informačních technologií VUT v Brně*

1. Úvod

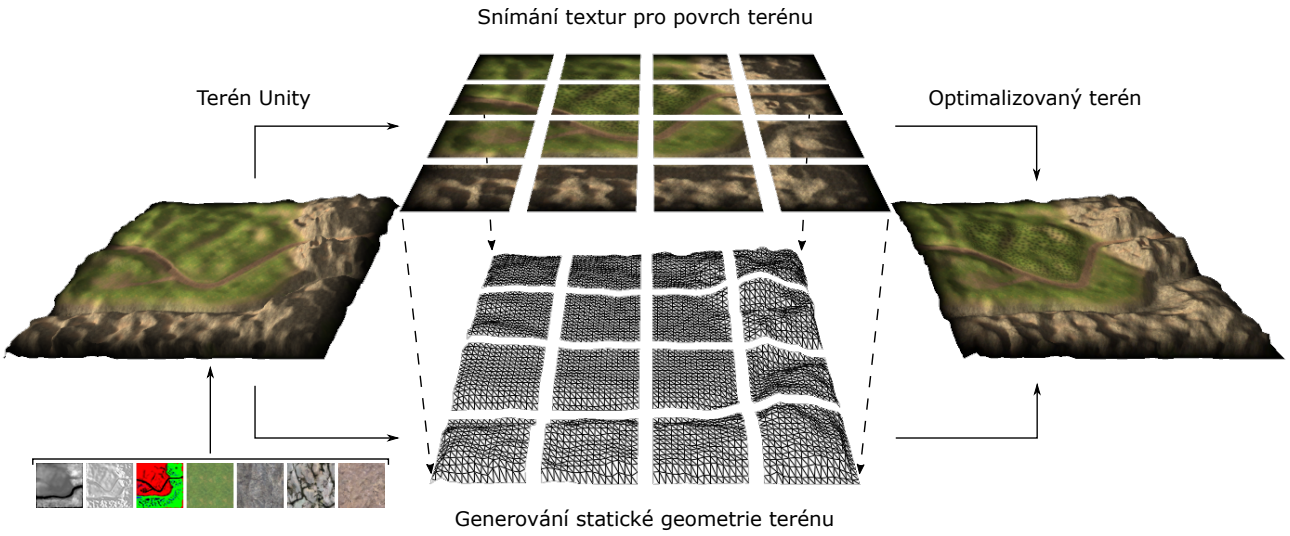
Videohry jsou interaktivní simulace orientované na *uživatelský zážitek*, který je výrazně ovlivňován nejen kvalitou vizuální zpětné vazby, ale také její odezvou a plynulostí. Iluze spojitého pohybu je, při vykreslování herních scén, vytvářena postupným promítáním rychle se střídajících snímků. Veškeré výpočty týkající se zobrazování, animací, fyzikální simulace a dalších oblastí, je proto nutné počítat s dostatečnou frekvencí v reálném čase, což klade vysoké nároky na výkon cílových platforem. Výkonnostní a paměťové limitace jsou pak obzvlášť znatelné při vývoji her pro mobilní telefony a tablety, vzhledem k jejich výrazně omezenému hardwaru. Optimalizace zobrazování scén je proto v této oblasti zcela nezbytná.

Trojrozměrné hry pro mobilní platformy využívají řadu metod, které vhodně, pomocí různých triků, imitují jinak výpočetně náročné vizuální efekty. Mnoho optimalizačních technik je také založeno na principech, které byly dříve intenzivně uplatňovány při vývoji her pro stolní počítače i konzole.

Přestože se optimalizacím vykreslování scén na mobilních zařízeních věnuje celá řada, převážně webových, zdrojů, často jsou popisovány pouze obecné poznatky, doporučení a základní metody, které engine Unity implicitně podporuje (například *frustum culling*, *occlusion culling*, *draw call batching* a podobně). Dokumentace enginu Unity uvádí tuto problematiku hned v několika sekcích [1, 2, 3]. Existuje však velmi málo zdrojů, které se zaměřují na pokročilejší optimalizační metody a jejich efektivitu a využití v praxi.

Pokročilé způsoby optimalizací scén pro mobilní zařízení popisuje především článek [4] a přednášky z vývojářských konferencí [5, 6, 7].

Cílem této práce je nejen prezentace konkrétních optimalizačních metod zaměřených na vykreslování scén na mobilních platformách, ale i vývoj prostředí vytvářející *grafický benchmark* pro tyto platformy. Prezentované prostředí má podobu pokročilé 3D strategické hry. Zásadní výhodou tohoto přístupu je, oproti syntetickým testům, možnost porovnání výkonnosti různých optimalizačních algoritmů v kontextu komplexní aplikace.



Obrázek 1. Proces optimalizace terénu pomocí prezentovaného nástroje.

Naivní implementace prezentovaných scén s využitím dostupných komponent a funkcí enginu Unity (terén, částicové systémy, *dynamic batching*) byla na mobilních zařízeních extrémně neefektivní. Snímková frekvence při zobrazování takových scén se pohybovala v řádech jednotek FPS. Pomocí dále prezentovaných technik došlo k mnohonásobnému nárůstu výkonu zobrazování, a to i na relativně starých zařízeních. Výsledná efektivita zobrazování scén se pohybuje okolo 25 FPS u zařízení ThinkPad Tablet z roku 2011 a 60 FPS u první generace tabletu Google Nexus 7 z roku 2012.

2. Optimalizace terénu

Terén je jedním ze základních elementů scény. Klasické techniky používané při zobrazování terénu jsou však pro mobilní zařízení, z důvodu výpočetní náročnosti, nevhodné. Režie spojená s řízením úrovní detailů geometrie terénu i složitost shaderů (vzorkování většího množství textur na jeden fragment) výrazně ovlivňuje celkovou efektivitu vykreslování.

Nástroj prezentovaný v této kapitole byl navržen za účelem automatické optimalizace libovolného terénu, vytvořeného pomocí komponent enginu Unity. Vývojáři tak mohou pro modelování terénu používat klasické pracovní postupy, které engine nabízí.

Výsledný optimalizovaný terén je tvořen jednou úrovní detailů pomocí statické geometrie, která je rozdělena do nezávislých dlaždic. Na každou dlaždici je namapována textura, jež obsahuje veškeré detaily povrchu terénu (barvy a statické stíny). Proces optimalizace se skládá z několika fází:

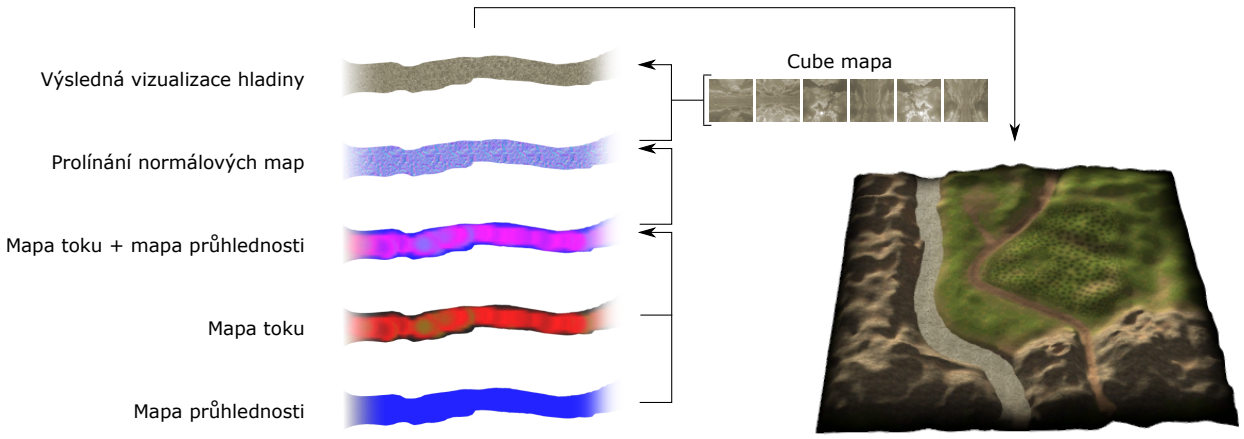
- Inicializační fáze vyžaduje určení vstupních parametrů optimalizačního procesu (například počet dlaždic nebo rozlišení geometrie a textur) a výběr zdrojového terénu – herního objektu s připojenou komponentou `Terrain`.
- Proces optimalizace rozdělí původní terén na zadaný počet dlaždic. Pro každou dlaždici proběhne vzorkování výšky původního terénu a generování nové statické polygonální sítě.
- Generování textur pro výslednou geometrii je realizováno ortografickým snímáním povrchu původního terénu. Jednotlivé snímky (textury) odpovídají povrchu konkrétních dlaždic, viz obrázek 1.

Zobrazování scén s takto vygenerovaným terénem je, oproti terénu Unity, výrazně efektivnější, a to především kvůli možnosti využití jednodušších shaderů, které vzorkují pouze jednu texturu na jeden fragment. Porovnání efektivity vykreslování optimalizovaného terénu a terénu reprezentovaného komponentou `Terrain`, při přibližně stejném počtu trojúhelníků (5 tisíc na každou variantu), je uvedeno v následující tabulce.

Tabulka 1. Porovnání výkonu při zobrazování scén s využitím terénu dostupného v enginu Unity a optimalizovaného terénu (se šestnácti dlaždicemi) vytvořeného prezentovaným nástrojem.

Zařízení	Terén Unity	Optimalizovaný terén
Google Nexus 7	19 FPS	(VSync limit) 60 FPS
ThinkPad Tablet	7 FPS	50 FPS

Nástroj pro generování optimalizovaného terénu umožňuje, mimo jiné, i automaticky vytvářet kolizní geometrii pro fyzikální simulaci. Rozdělení terénu na jednotlivé dlaždice podporuje *frustum culling* dostupný v enginu Unity.



Obrázek 2. Proces zobrazování vodní hladiny.

3. Zobrazování animované vodní hladiny

Následující sekce popisuje efektivní metodu pro vykreslování animované a přirozeně vypadající vodní hladiny. Klasické techniky, které se pro tyto účely typicky využívají, jsou, stejně jako v případě zobrazování terénu, pro mobilní platformy nevhodné, a to především kvůli omezenému přístupu k paměti hloubky (*depth buffer*) u řady mobilních telefonů i tabletů.

Výše uvedené omezení mobilních zařízení neumožňuje efektivně zobrazovat plynulé přechody mezi terénem a vodní hladinou. Z toho důvodu využívá prezentovaná metoda texturu, která udává průhlednost vodní hladiny v definovaných oblastech. Využití takové textury výrazně zjemní přechody v okolí hranice vody a terénu.

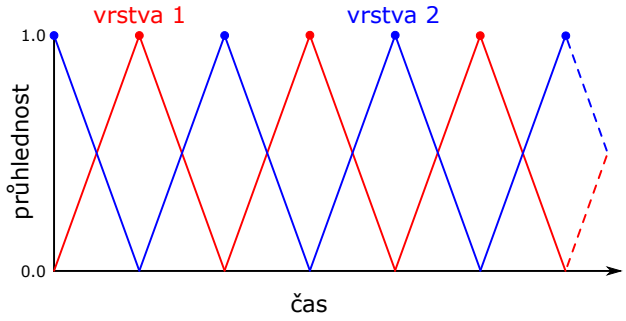
Animace toku vody je tvořena periodickým prolínáním dvou pohybujících se textur (normálových map), namapovaných na polygonální model hladiny. Tento postup však z principu umožňuje vizualizovat tok vody pouze v jednom libovolném směru, což je nevhodné pro vizualizaci vodní hladiny řek, kde směr toku vody kopíruje řečiště. Prezentovaná metoda proto umožňuje využít i takzvanou *mapu toku* (*flow map* [8]).

Mapa toku je textura, která definuje dvoudimenzionální směrové vektory v rovině hladiny a určuje tak směr a rychlost toku vody v daných oblastech, což umožňuje vytvořit přirozeně vypadající vizualizaci řeky.

Protože je pro určení směru toku využit pouze dvousložkový vektor, je možné mapu toku sloučit s texturou, která v jedné složce definuje průhlednost vodní hladiny. Sloučení těchto textur přispívá k efektivitě metody. Složky r a g nesou směrový vektor pohybu vody v daném místě, složka b pak definuje průhlednost vodní hladiny.

Textury, které se dle mapy toku pohybují po vodní hladině, určují normálové vektory, na základě nichž probíhá vzorkování *cube* mapy [9]. Tímto postupem je docíleno přirozeně vypadajících odrazů. Celý proces vykreslování vodní hladiny je ilustrován na obrázku 2.

Vizuální kvalita prezentované metody je výrazně závislá na použitých texturách a rychlosti pohybu textur po hladině. Při použití mapy toku dochází k deformaci těchto textur. Aby bylo možné deformace skrýt, textury hladiny se periodicky aditivně prolínají a posouvají pouze po malých úsecích. Průhlednost obou textur hladiny je v součtu vždy rovna jedné a střídá se dle principu ilustrovaného na obrázku 3. Tento postup umožňuje jednoduše maskovat stavy, ve kterých dochází k resetování posunu textur – posun každé vrstvy je resetován v době, kdy má daná vrstva plnou průhlednost. Nedochází proto k viditelným skokům v pohybu hladiny.



Obrázek 3. Princip střídání průhlednosti vrstev – restart posunu textur probíhá v místě, kdy je daná vrstva plně průhledná.

Prezentovaný postup vykreslování vodní hladiny generuje přirozeně vypadající výsledky. Výpočetní náročnost metody je i přesto stále dostatečně nízká. Tuto techniku je proto možné využít i na mobilních telefonech a tabletech.

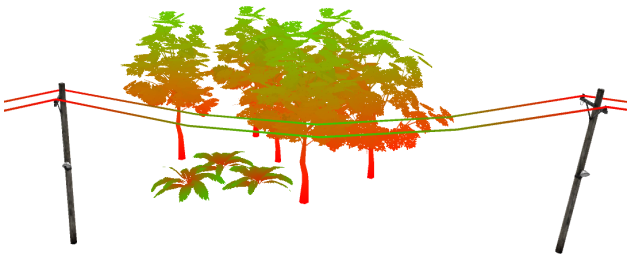
Integrovaná metoda vizualizace toku vody je založena na principu prezentovaném v dokumentu [8].

Obrázek 4. Výsledná vizualizace vodní hladiny.

4. Animace na bázi vrcholů

Animace jsou zásadní součástí procesu vykreslování přirozeně působícího prostředí. Postup prezentovaný v následující sekci představuje velmi efektivní způsob animace libovolných elementů scén, jejichž pohyb lze aproximovat například pomocí goniometrických funkcí. Nízká výpočetní náročnost této metody, umožňuje, bez zásadního vlivu na výkon, animovat stromy a jiné rostliny, případně další objekty, například dráty elektrického vedení nebo vlajky a tkaniny pohybující se ve větru.

Veškeré výpočty související se samotnou animací jsou prováděny ve fázi transformace vrcholů ve vertex shaderu na grafickém akcelerátoru. Geometrie animovaných modelů je deformována (animována) na základě goniometrických funkcí. Atributy vrcholů animovaného modelu určují, vedle typických hodnot, jako jsou pozice, normálové vektory nebo texturové souřadnice, také *váhu*, která definuje intenzitu aplikace pohybu na daný vrchol. Hodnota váhy se obvykle nachází v intervalu $< 0, 1 >$. Vrchol s nulovou váhou se nepohybuje, naopak vrchol s váhou rovnou jedné se pohybuje s plnou intenzitou. Obrázek 5 vizualizuje příklad použití váhy při animaci různých objektů.



Obrázek 5. Příklady animovaných objektů s vizualizací intenzity pohybu - červené oblasti objektů se nepohybují (váha je rovna nule), zelené oblasti se pohybují (váha je rovna jedné).

Výpočet animace pohybu je založen na různých postupech. V případě animace stromů lze pohyb aproximovat jednoduše pomocí goniometrických funkcí a závislosti na čase. Pro zobrazování celého lesa je vhodné zanést do fáze pohybu stromů určitou náhodnost. Toho lze dosáhnout například připojením dalšího atributu vrcholu, který ponese, pro každý strom, náhodně vygenerovanou hodnotu, jež se využije pro modifikaci času a dojde tak k rozptýlení fází pohybu jednotlivých stromů. Stejné postupy lze využít i pro další rostliny a jiné objekty. V případě animovaných drátů elektrického vedení jsou navíc použity pro výpočet pohybu modifikované normálové vektory, které určují směr, jímž se dráty mohou pohybovat.

Následující tabulka prezentuje efektivitu animací na bázi vrcholů. Testovací scéna zahrnovala v záběru kamery více než 800 stromů a přibližně 80 dalších objektů, na které byly animace aplikovány (přibližně 85 tisíc trojúhelníků a 125 tisíc vrcholů). Přesto došlo pouze k minimálnímu ovlivnění výkonu.

Tabulka 2. Vliv animací na efektivitu zobrazování scén.

Zařízení	Bez animací	S animacemi
Google Nexus 7	52 FPS	49 FPS
ThinkPad Tablet	26 FPS	25 FPS

Prezentovaná metoda poskytuje velmi efektivní způsob animace velkého množství objektů i na mobilních zařízeních a umožňuje tak vytvářet přirozeně působící prostředí.

Obrázek 6. Efektivní animace lesa, rostlin, drátů elektrického vedení a dalších objektů.

5. Analýza výkonu

Analýza výkonu prezentovaných scén je, kromě klasického profilování, prováděna také pomocí implementovaných skriptů, které umožňují zaznamenávat efektivitu vykreslování za běhu aplikace na mobilním zařízení. Statistiky výkonu jsou pak přímo ze zařízení odesílány na server pro zpracování a vyhodnocení.

Obrázek 7. Scény optimalizované pro mobilní platformy – z původních jednotek snímků za sekundu na obou testovacích zařízeních došlo při použití optimalizačních technik ke zrychlení na průměrných 25 FPS u tabletu ThinkPad Tablet z roku 2011 a 60 FPS u první generace tabletu Google Nexus 7 z roku 2012.

6. Závěr

Vybrané metody popisované v tomto článku jsou, společně s dalšími algoritmy, využity při vývoji komplexních scén, které umožňují vývojářům analyzovat vliv konkrétních optimalizačních postupů na výkon vykreslování. Prezentované techniky jsou vhodné nejen pro mobilní platformy. Jejich užitím lze dosahovat vizuálně kvalitních výsledků na průměrných mobilních telefonech i dalších zařízeních s nižším výpočetním výkonem. Vytvořené komplexní scény zahrnují řadu různých objektů (mimo jiné také vozidla, postavy nebo částicové systémy) a demonstrují tak, na praktických příkladech, mnoho přístupů k řešení konkrétních problémů při vývoji her pro mobilní zařízení.

7. Poděkování

Tento příspěvek vznikl pod vedením pana Ing. Rudolfa Kajana. Některé vizuální materiály použité v této práci (textury, modely) jsou autorským dílem Michala Zachariáše, Davida Jozefova a Martina Wilczáka.

Literatura

[1] Unity Technologies. Optimizing graphics performance. In *Unity Manual*. Unity Technologies, 2015 [cit. 2015-03-19]. documentation.

[2] Unity Technologies. Practical guide to optimization for mobiles. In *Unity Manual*. Unity Technologies, 2015 [cit. 2015-03-19]. documentation.

[3] Unity Technologies. Rendering optimizations. In *Unity Manual*. Unity Technologies, 2015 [cit. 2015-03-19]. documentation.

[4] Will Goldstone. Shadowgun: Optimizing for mobile sample level. In *Technology - Unity Blog*. Mar 2013 [cit. 2015-03-19].

[5] Aras Pranckevičius and Renaldas Zioma. Optimizing unity games for mobile platforms, Aug 2011 [cit. 2015-03-19]. SIGGRAPH 2011.

[6] Alexander Dolbilov. Optimizing unity games, Aug 2014 [cit. 2015-03-19]. Google I/O 2014.

[7] Angelo Theodorou. Optimizing unity games for mobile platforms, Aug 2013 [cit. 2015-03-19]. Unite 2013.

[8] Alex Vlachos. Water flow in portal 2, 2010 [cit. 2015-03-19]. SIGGRAPH Course on Advances in Real-Time Rendering in 3D Graphics and Games.

[9] Unity Technologies. Cubemap. In *Unity Manual*. Unity Technologies, 2015 [cit. 2015-03-19]. documentation.