UDK: 004.2:72.01
Прегледни рад

НОВИ ПРИСТУП УБРЗАЊУ ПОСТУПКА КРЕИРАЊА ГРАФИЧКИХ АНИМАЦИЈА У АРХИТЕКТУРИ

Марко Радојчић

рад примљен: јануар 2011, рад прихваћен: март 2011.

A new approach to acceleration of methods of creating graphic animation in architecture

Апстракт:

Модерне методе техничке визуелизације нису могуће без примене савремених метода приказивања тродимензионалних модела у различитим формама анимације. Иако постоји потреба за брзим приказивањем резултата процеса пројектовања у грађевини, архитектури и другим техничким дисциплинама, ипак се користе класичне методе које су временски дуготрајне, и у случају измена захтевају поновно покретање целокупног поступка. У овом раду предложена је нова метода у којој се користе елементи софтвера отвореног кода и која се може модификовати тако да омогућава знатно бржу техничку визуелизацију, уз могућност исправки на лицу места, као и максимално ефикасно коришћење процесорских снага савремених графичких чипова.

Кључне речи: тродимензионалне анимације, убрзање рендеринга, програмирање графичких процесора, тродимензионалне анимације у реалном времену

Abstract:

Modern approach to technical visualization methods is not possible without using contemporary 3D animation methods. Obvious need of demonstrating the results of design process in civil engineering, architecture and other technical areas still includes classical time-consuming and non-flexible methods that require the process to restart in case of making some changes on the design. This method proposes changes based on some already available open-source software packages and components that can be modified in a manner that unlocks accelerated technical visualizations in real time and with instant modifications along with efficiently usage of the processing power of contemporary graphics processing units.

Key words: 3D animations, rendering acceleration, graphics processing unit programming, real-time 3D animations

Увод

Техничка визуелизација стара је колико и техника сама, тј. колико је стара и сама људска потреба да резултати мисаоног процеса буду прецизно графички представљени. Најстарији примери техничке визуелизације потичу још из времена настанка цивилизације и ова дисциплина развијала се упоредо са развојем науке која је управо покретач цивилизацијског развоја. У прошлости, веома су значајни били први пројекти које су стварале архитекте и проналазачи за описивање резултата свог креативног поступка. Увођењем стандардизације поступка пројектовања као и развојем нацртне геометрије и егзактне перспективе, ова грана техничке делатности добила је нову димензију. Модерна комуникација између различитих учесника у процесу развоја и израде техничке документације и њихова размена резултата пројектовања тешко се могу замислити без адекватног графичког и визуелно упечатљивог приказа објеката уз помоћ рачунара. Овај приказ се у све већем броју случајева реализује прављењем анимација на основу дизајнираних модела тродимензионалних објеката.

Сама методологија израде тродимензионалних анимација може се поделити у четири основна корака, где су прва два по обиму и значају испред осталих: 1) превођење пројеката у основне тродимензионалне моделе погодне за обраду у програмима за моделирање, 2) израда модела тродимензионалних објеката и дефинисање материјала и интеракција светлости са тим материјалима како би се дошло до јасне и упечатљиве слике објекта, 3) подешавање анимације, тј. дефинисање кретања и деформација објеката како би се

симулирали процеси које треба приказати, 4) за рачунар најзахтевнији процес - процес сачињавања (рендеринг) анимације од геометријски дефинисаних рачунарских модела. У овом раду посебно ће се разматрати процес рендеринга анимација услед потребе да се овај процес убрза и чињенице да алтернативне методе рендеринга могу да понуде квалитетан приказ у значајно краћем временском интервалу, као и синтеза поступка моделирања са поступком рендеринга.

ТЕХНОЛОГИЈА ЗА КРЕИРАЊЕ И ПРИКАЗ ТРОДИМЕНЗИОНАЛНИХ АНИМАЦИЈА

Савремене технологије за креирање тродимензионалних анимација заснивају се на програмима као што су *3D Studio MAX*, *Maya*, *Softimag*e, *Blender* итд. Сваки од ових програма има своје специфичности.

У основи, то су програми који омогућавају рад са датотекама у различитим форматима записа тродимензионалних просторних описа модела и њихових кретања и који цео процес рада са будућом анимацијом смештају у окружење које је више дизајнерско него техничко.

AutoCAD као универзални алат за цртање има такође добру подршку за рад са тродимензионалним моделима, а ради то на начин који је примеренији техничкој струци.

Заједнички именитељ поменутих алата је то да су то програми затвореног кода (што значи да су начин функционисања и измене у њиховом раду доступни само њиховим произвођачима), изузев програма *Blender* који је отвореног кода и који има бројне друге предности, међу којима је и посебан део који служи као основни мотор (*engine*) за видео игре, а који уз различите модификације може представљати одличну основу за приказивање детаљних анимација у реалном времену.

Суштина структуре података са којом се ради у програмима за *3D* моделирање је тачка тј. вектор положаја тачке у простору који представља основу за опис положаја, облика и величине објеката који се у датом простору приказују. Скуп тачака дефинише јединичну површину – најчешће троугао или четвороугао који служе за описивање облика површине објекта. Осим трокомпонентног вектора положаја тачке, за опис површине објекта, тј. једног његовог дела, неопходни су материјали додељени том објекту, који служе да унапред дефинишу интеракцију светлости са предметом, као и рефлексију, рефракцију, храпавост и друге особине стварних објеката које је неопходно узети у обзир у прорачунавању изгледа површине која се приказује у коначној слици. Ово подразумева додавање нпр. слике површине зида од цигле на зид који је дефинисан као геометријски објекат, што заједно са особинама материјала у смислу прозирности, спектра светлости при преламању и одбијању чини опис објекта.

Сви ови подаци улаз су за процес рендеринга у поступку праћења путање светлосног зрака (познатије као *raytracing*[1]). Наиме, стандардне методе израде тродимензионалних анимација почивају на томе што се процес стварања слике заснива на методи праћења путање светлосних зрака и анализи њихових интеракција са објектом. Овај поступак је временом еволуирао, од основног приказа који је бојом попуњавао контуру објекта, до све сложенијих поступака који су обухватали све већи број параметара. Велика мана ове методе је што се прорачун врши и за делове простора који су празни, па тако прорачун обухвата јако велики број параметара и када то није неопходно. Иако даје фотореалистичну слику, овај поступак је процесорски веома захтеван и програмски сложен. За сваку врсту интеракције са светлошћу која далази до камере од дате сцене са предметом који је

[1] Python - интерпретирани програмски језик, објектно оријентисан и релативно једноставан за употребу, уграђен је у систем програма: *Blender 3D modeler*

у видном пољу, постоји посебан поступак који рачунар мора да изврши како би произвео квалитетну и реалистичну слику. Нажалост, овај поступак је спор, пре свега зато што се слика прорачунава по целој запремини тродимензионалне сцене, а прорачун није довољно оријентисан ка самим објектима који ту сцену чине, непотребно је сложен и извршава се на централном процесору, игноришући напредак процесорских потенцијала данашњих графичких чипова (*Shreiner, Woo, Neider, Davis,* 2008).

За потребе развоја видео игара развијене су алтернативне методе које су остале недоступне корисницима дизајнерских алата јер се углавном заснивају на методама нацртне и аналитичке геометрије. У овом раду предложена је једна таква алтернативна метода, код које се за израду анимације уместо централног процесора рачунара користе процесори графичке картице, чиме се поступак енормно убрзава.

Модерни програми за праћење путање светлосног зрака *(raytracing),* без обзира да ли су интегрални део неког пакета за тродимензионално моделирање (*3D Studio MAX*, *Maya*, *Softimage*...) или су самостални програми сачињени искључиво за ту намену (*Yaf(a)ray*, *PovRay* и др.), у своју функционалност успешно интегришу и веома сложене особине материјала: рефлексије, преламања, површинску порозност, неравнине, бразде, сложене функције расподела преламања, као и додатне ефекте: преламање спектра, прозирност, замућење провидних површина и др.

Оваква сложеност софтвера за продукцију слика (рендеринг) заснованих на методама праћења светлосног зрака, довела је до тога да се један (можда релативно једноставан) модел пропушта кроз сложену серију захтевних филтера који, уколико су правилно конфигурисани, дају веома веродостојан приказ дизајна модела, понекад и непотребно детаљан.

Пошто је процес рендеринга сложен и захтеван, било је много покушаја да се он оптимизује и убрза (*http://en.wikipedia.org/wiki/Rendering_%28computer_graphics%29*)

Алгоритамски модели који садрже *KD-Tree* оптимизације или праћење зрака, заснивају се на моделу униформне мреже. Примена бафера дубина као и многе друге методе убрзали су овај процес. Ипак, за потребе техничке визуелизације, где је углавном довољно приказати сврху и функционалност објекта или неког елемента конструкције или дела машине, није неопходно увек вршити детаљну анализу методама *raytracing*-а, те постоје методе које се заснивају на технологијама примењеним у видео играма и окружењима виртуелне стварности које могу дати, у вишеструко краћем временском року, визуелно сасвим задовољавајући резултат, привлачан на око и технички функционалан.

Поређења ради, методама детаљног *raytracing*-а, стварање једне слике која чини основну јединицу анимације, може трајати од неколико минута до неколико сати на савременом рачунару са два или више процесорских језгара. Иако је ово време пропорционално сложености модела и примењених ефеката, процес је и уз оптимизације захтеван и користи искључиво процесорску снагу централног процесора (*CPU*), не користећи предности технологија развијених за приказ тродимензионалне графике у реалном времену, а сведоци смо да се визуелно импозантне симулације и игре појављују у свим сферама рачунарског посла.

Алтернативне методе које се заснивају на примени високо специјализованих чипова за обраду графике (***G****raphics* ***P****rocessing* ***U****nit – GPU*) користе другачији приступ иако не дају увек фотореалистичне резултате, користећи се методама нацртне геометрије и брзих трансформација над објектима аналитичке геометрије. Уз прорачун сенчења специјалним јединицама (процесорима) за сенчење (*shader units*), ове методе могу произвести анимацију у реалном времену, тј. 60 и више слика у секунди, насупрот класичним програмима у којима је за једну слику у анимацији потребно више минута. Ово омогућава да се после подешавања параметара система за приказ графике у реалном времену, анимација која треба да траје 3 минута добија за 3 минута, а не за 3 мин x 60 с/мин x 30 фрејмова(слика)/с x 1 мин/слици = 5400 мин, што износи 90 сати.

Постоји неколико технологија које омогућавају приказивање захтевних анимација у реалном времену. Међу њима, најзаступљеније су: *Microsoft*-ов власнички *DirectX* и *OpenGL* – библиотека отвореног кода.

Иако имају много заједничких особина, а и сав модернији хардвер за приказ анимација тродимензионалних објеката углавном подржава најновије верзије обе од ових технологија, у контексту примене за убрзану техничку визуелизацију и лакоће модификације алата услед отворености кода, у овом раду пажња ће бити окренута *OpenGL* технологији.

OpenGL (*Open Graphics Library*) је технологија коју је иницијално покренула компанија *Silicon Graphics Inc*. 1992. године и од тада се *OpenGL* развијао пратећи модерне захтеве за квалитетном и брзом анимацијом и пратећи и подстичући развој модерног графичког хардвера.

Принцип рада убрзаних метода

Без улажења у детаљне спецификације *OpenGL API*-ја (*Application Programming Interface*), битно је истаћи да систем за рендеринг који се заснива на *OpenGL* библиотеци функционише слично цевоводу кроз који пролазе елементи графике један по један, бивају обрађени и на крају приказани на екрану. Овај процес има веома висок ниво паралелизација у делу који се односи на напредну обраду операција над тачкама у *3D* простору (*vertex shader*), као и у делу који извршава напредне операције сенчења и обраде слике над појединачним елементима излазне слике (*fragment* или *pixel shader*).

Како су временом еволуирали графички адаптери и потребе информатичке индустрије за већом процесорском снагом посвећеном обради реалистичније тродимензионалне графике, осим генералних убрзања у дефинисању модела *3D* објеката и њиховог облагања сликама (текстурирање), највећи помаци учињени су у архитектури *pixel* и *vertex shader* процесора који функционишу као минијатурни процесори у оквиру графичког чипа, способни да извршавају програме за обраду графике који су им задати, те одатле потиче њихова програмабилност.

Програмабилност пригушница (јединица за сенчење и прецизну обраду слике) омогућава да се веома брзо обраде сложене графичке композиције, замењујући обраду слике праћењем светлосног зрака (замењујући *raytracing*) високо паралелизованом обрадом елемената слике у специјализованим процесорима за графику.

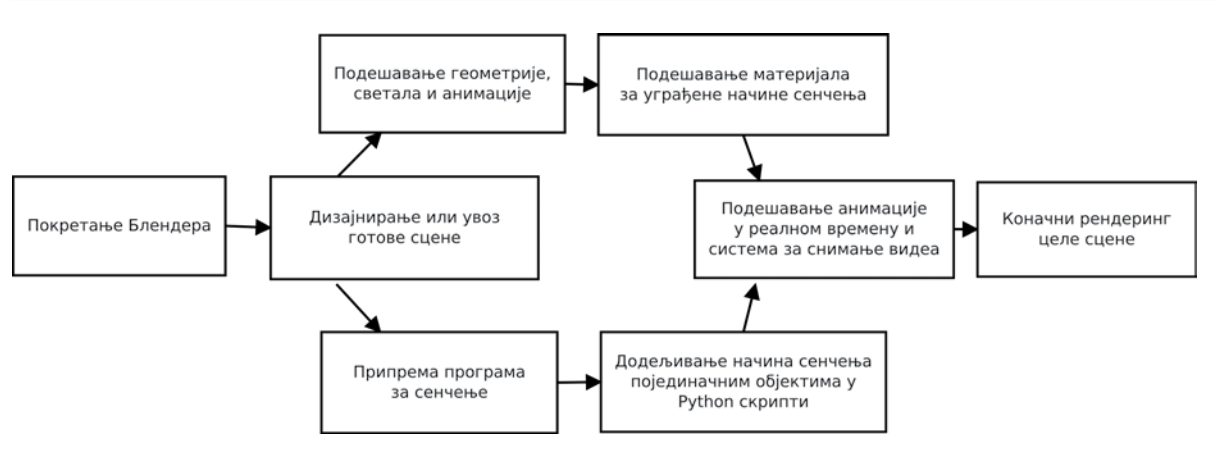
Пошто *OpenGL* библиотека својим скупом функција омогућава детаљно управљање процесом рендеринга и програмирање пригушница, изабрана је за платформу за реализацију рендеринга анимација у реалном времену (*Astle*, *Hawkins*, 2002).

Осим тога, *OpenGL* библиотека је основа приказа слике па су тако у систему за приказ игара у оквиру дизајнерског окружења *Blender*-а спојене две крајности – довољно близак приступ брзој системској функционалности и програмабилности графике и поштовање потреба и навика аутора тродимензионалних анимација којима су програмски језици и *OpenGL* команде непознате.

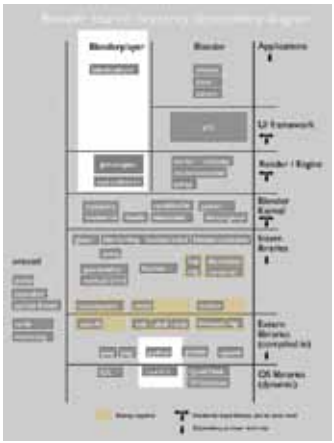
Основа овог решења је *Blender Game Engine*, уз програмирање *Python*1 програмима и *OpenGL GLSL* (*GL Shading Language*) програмима за прецизно управљање детаљном графиком. Шематски приказ редоследа операција у *OpenGL* систему рендеринга приказан је на слици 1 која је настала модификацијом слике дате у литератури (Rost, 2006).

Основна платформа за реализацију приказивања анимација у реалном времену је *Blender*. Овај програм отвореног кода у себи садржи све битне елементе за увоз пројеката и модела из других програма за рад са тродимензионалном графиком, садржи потребне опције за уређивање и модификацију таквих сцена, има напредне механизме за рад са материјалима и текстурама појединачних објеката као и ефикасни *Python* интерфејс који омогућава да се из једноставних скрипти писаних у програмском језику *Python* приступи свим објектима у некој сцени и њиховим особинама и елементима.

Ова могућност динамичке манипулације објектима у сцени омогућава да се реализује *3D* графика која ће осим стандардних особина материјала имати додате и програмабилне компоненте и тако омогућити интерактивну



Сл. 1.
Шематски приказ редоследа операција у OpenGL систему рендеринга
Fig 1.
OpenGL rendering (in Blender 3D modeler)



Сл. 2.
Програмска архитектура Blender 3D modeler-a
Fig. 2
Blender 3D modeler source directory dependency

графику високог квалитета у реалном времену, додатно побољшаних графичких карактеристика помоћу *OpenGL GLSL* (*GL Shading Language*) програма који детаљно програмирају процесоре за геометријске операције и сенчење на графичком чипу из елегантног и приступачног интерфејса једноставних *Python* скрипти.

Такође, додатним унапређењима рада са скриптама и *shader* програмима омогућено је да се систем обједини и аутоматизује, а напреци учињени у најновијим верзијама програма *Blender 2.48a* и *2.49a* омогућавају да се за већину стандардних материјала и метода сенчења и мапирања графике овај процес у потпуности аутоматизује.

Blender 3D modeler има програмску архитектуру приказану на слици 2 - преузето са интернет стране Блендер фондације (*Blender foundation*) *(http://www.blender.org/development/architecture/, http://www.blender.org*)

Битно је приметити да су у основи *Blender*-а са једне стране *OpenGL* библиотеке, а са друге стране *GameEngine* и *BlenderPlayer*. Ове компоненте су елементи који спајају могућност рада са прецизно програмираним напредним ефектима, везујући се за оперативни систем рачунара и сам графички процесор са једне стране и флексибилност и интерактивност *Blender Game Engine*-а као система за анимацију у дизајнерском окружењу, са друге стране.

Компромис – однос квалитета слике и брзине приказивања

Изразите разлике технологија за приказ графике уочавају се у примерима у којима су напредни ефекти за обраду слике укључени (преламања светлости, напредне рефлексије, прозрачност и др.) и у оним када су у графици у реалном времену искључени. Разлика се може видети на излазним резултатима, али није толика да елиминише оправданост употребе убрзане методе (видети слике 3 и 4). Елементи графичког приказа који се делимично губе приликом преласка са класичне методе на методу графике у реалном времену су: квалитетне сенке, преламања светлости на површинама попут стакла или површине воде, као и неки ефекти везани за просторна замућења, као што су магла или дим.

Већина графичких композиција намењених техничкој визуелизацији не користи овакве напредне светлосне ефекте, јер успева да оствари своју сврху – преношења јасне и концизне просторне замисли – и без оваквих ефеката.

Користећи доступне алате у *Blender 3D modeleru,* могуће је управљати битним аспектима сцене и обезбедити пуну функционалност анимације, као и искористити бројне предности алтернативног решења. Осим брзине израде саме анимације, механизам у ком се појединачна слика ствара скоро тренутно, омогућава да се анимација поново креира у року од само неколико минута, уколико је због захтева инвеститора или неког другог учесника у пројекту неопходно изменити одређене аспекте просторне композиције сцене – нова анимација биће на располагању практично одмах. Такође, без већих проблема могуће је креирати интерактивне анимације у којима би посматрач могао да се креће кроз виртуелни свет новопланираног објекта, или би пројектант могао на време уочити недостатке у машинском склопу који „ради" пред њим.

Губитак детаља присутан у коначној анимацији приликом преласка на нови метод, везан је и за проблематику недовољне стандардизације формата записа тродимензионалних анимација.

Пошто постоји велики број програма за тродимензионално моделирање и анимацију, који се константно унапређују новим верзијама и записују своја документа у сложеније формате, поступак превођења оригиналне датотеке са тродимензионалном сценом чију анимацију треба сачинити јесте активност која се временом све више и више компликује.

Будући да различити програми третирају геометријске објекте и њихове особине на различите начине, практично је немогуће створити систем за конверзију формата који би био стопостотно компатибилан са свим форматима.

Blender 3D modeler садржи већи број потпрограма за тумачење и увоз података из формата других програма за тродимензионалну анимацију, а једно од решења је и стандардизација у погледу коришћења *OpenCollada* формата који постаје посреднички формат за размену

докумената са тродимензионалним анимацијама. Да би се овај проблем ублажио, у оквиру развоја методе анимације у реалном времену, развијала се и метода за увоз докумената у *Blender 3D modeler*, али нужно, нека подешавања или детаљи анимације не могу имати потпуну компатибилност са осталим форматима. Чак иако систем за приказ анимација у реалном времену подржава одређене опције за приказ слике, нису увек све опције добро протумачене и минимална одступања могу се приметити на документима преведеним у *Blender 3D* формат.

Главни разлог за постојање разлике између анимација сачињених класичним методама и методом приказивања графике у реалном времену јесте у начину рада самог система.

Примери приказивања тродимензионалне графичке композиције која може бити довољна по количини детаља као израз архитектонске или опште просторне замисли, могу се видети на сликама 3 и 4.

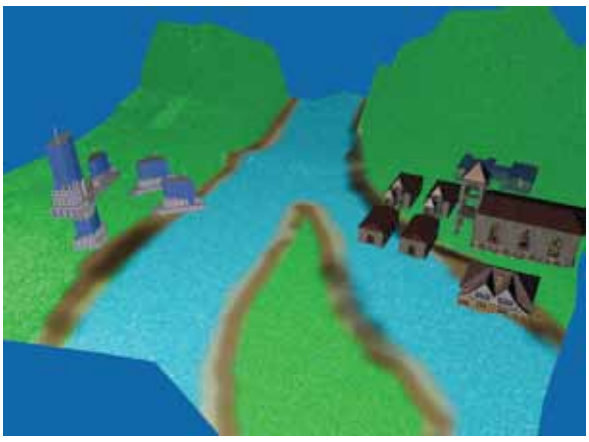
Може се приметити да за композиције са мање детаља и без напредних ефеката разлика није значајна, али и даље постоји. Различитост излазних слика последица је принципијелних разлика у начину обраде слике и огледа се у чињеници да приликом праћења путање светлосног зрака основна мера моћи разлагања слике је појединачни светлосни зрак који долази до ока камере. Оваква метода у детаљно дефинисаним сценама може произвести фантастичне фотореалистичне ефекте, док се при анимацији у реалном времену добија нешто другачији приказ, али у много краћем временском интервалу. При анимацији у реалном времену проблему се приступа са гледишта нацртне геометрије, тј. основни предмет обраде су геометријске карактеристике објекта које се међусобно усклађују са параметрима осветљења, не трошећи ресурсе на прорачуне у деловима сцене у којима нема објеката или они нису видљиви зато што се налазе иза неких непровидних објеката у сцени.

Иако скоро потпуна фотореалистичност има својих предности у неким применама, за стандардне потребе техничке визуелизације ова метода је непотребна, јер се на апсолутној фотореалистичности не инсистира. Пример приказа архитектонских објеката са довољном количином детаља може се видети на слици бр 4. и може се поредити са истом сценом приказаном на слици бр. 3 која је произведена коришћењем методе детаљног *raytracing*-а. Најбитнија разлика у раду са програмом је начин обраде сенки, јер се у *raytracing* методи сенке третирају као запремински објекти, са јасним контурама, док су у обради путем графичког процесора, без додатног програмирања сенки, приказани само ефекти осветљења на површинама објеката, што је уједно и демонстрација фундаменталних разлика у начину рада једне и друге методе.

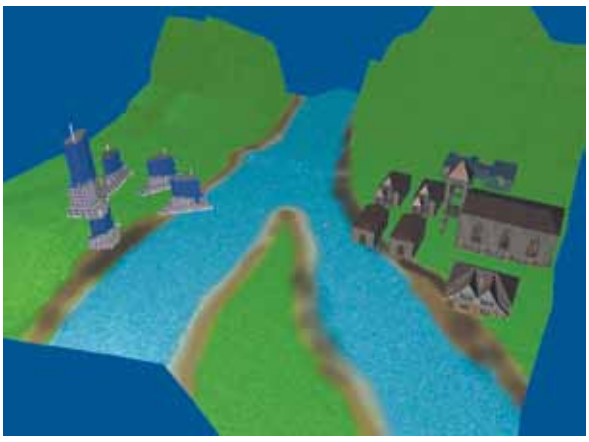
Са друге стране, перформансе које се могу остварити методом приказивања графике у реалном времену имају извесна ограничења, али имају и велику предност која се огледа у производњи десетина слика једне анимације у секунди. Елементи анимације као што је покретање камере или тренутна модификација елемената сцене могу се остварити у оквиру *Blender Game Engine*-а, тако да се уз мале модификације параметара анимације – после евентуалне конверзије формата датотеке *3D* анимације – добија видео запис простим снимањем дешавања на екрану, тј. резултата рендеринга који обавља графички процесор коришћењем неког од програма за хватање садржаја екрана, као што су на пример *FRAPS* или адекватни *Linux* програми који врше исту функцију, а постоје одлична решења и за друге оперативне системе.

Овако се остварују значајна убрзања која се мере разликом у времену које је потребно да се произведе појединачна слика.

Сл.3.
Рендеринг средње сложене сцене методом праћења путање светлосног зрака (време сачињавања једне слике је око 3 с)
Fig. 3.
Moderately complex scene rendered by raytracing (3s per single frame)



Сл. 4.
Сачињавање анимације у реалном времену применом *OpenGL* технологије – преко 100 слика у секунди
Fig. 4.
OpenGL realtime animation – over 100 frames per second



Сл.5.
Приказ разлике између слика произведене методом графике у реалном времену (горе) и прорачуна путање светлосног зрака (доле)
Fig 5.
Comparison between images produced using realtime graphics (upper figgure) and raytracing (lower)

На слици 4 може се уочити приказ брзине рендеринга који указује на то да више од 100 слика бива произведено у секунди, за разлику од сцене за чији је рендеринг, методом *raytracing*-а, било потребно више секунди за сваку појединачну слику.

Иако се у процесу конверзије анимације губе извесна фина подешавања сцене и нису обрађени сви графички елементи истом прецизношћу, примена *Blender Game Engine*-а или уопште *OpenGL* технологије и производа за видео игре базираних на овој технологији, јавља се као оправдан компромис.

На слици 5 дат је упоредни приказ графике коју за исту сцену производи убрзана (горе) и класична (доле) метода.

ПРОЦЕСОРИ ЗА СЕНЧЕЊЕ И ЊИХОВО ПРОГРАМИРАЊЕ

У првим фазама развоја *OpenGL* технологије постојала је ограничена и фиксна функционалност система за обраду слике. Временом је број подржаних стандардних процедура од стране произвођача графичких процесора растао и јавила се потреба за проширењем сета ових опција. Све више модерних видео игара користило је различите „трикове" како би произвеле специјалне ефекте, али ове могућности ипак су биле ограничене.

Временом, јавила се потреба за развојем програмабилних делова *OpenGL pipeline*-а[2].

Фиксна функционалност *OpenGL* операција у светлосним ефектима и трансформацији слика које се налазе на површини објеката (текстурирање), замењена је системом који омогућава да се ефекти који се користе програмирају произвољним алгоритмом који може садржати веома сложене процедуре.

Тако је настао *GLSL* (*GL Shading Language*). Ово је програмски језик високог нивоа који је сличан програмском језику *C* (*Rost*, 2006).

Програмирање пригушница може се поделити на два важна сегмента. Први је *vertex shader* (програмирање сенчења које се односи на геометријску трансформацију површине објекта, како би се добио детаљнији приказ). Резултати *vertex shader* потпрограма могу бити прослеђени *pixel shader*-у (који се назива и *fragment shader*), који је друга фаза поступка и који обрађује унапред трансформисану слику у складу са перспективом и геометријским трансформацијама и на њој врши операције приступа текстурама и обраде завршних ефеката као што су: операције осветљења, сенчења, пресијавања специфичних материјала и генерисање слика за материјале (дрво, мермер и сл.).

2 pipeline- цевовод, у овом контексту, систем који резултате једног сегмента користи као улазне податке за следећи

Метода програмирања *GLSL* програма је да се припреме унапред дефинисани програми за различите ефекте у виду текстуалних датотека и њихови улазни параметри (вредности одређених величина, слике које се користе за текстурирање, геометријски описи објеката), а затим се приликом приказа објеката ови потпрограми пошаљу графичкој карти на обраду те се резултат може видети одмах на екрану.

Модерни графички чипови имају десетине, па и стотине *shader* процесора, који омогућавају да чак и када се ради о додатно побољшаној слици напредним *GLSL* програмирањем, резултат сачињавања анимације буде доступан у реалном времену.

Такође, најновији чипови имају „Уједињену *shader* архитектуру" , али то не утиче битно на примену у програмирању *OpenGL shader* процедура, већ само позитивно утиче на перформансе тих чипова и нуди неке додатне програмерске могућности.

Осим примене за унапређење графике, *shader* процесори, првенствено *pixel* (*fragment*) *shader*-и имају примену и у обради података у оквиру *GPGPU* пројеката (*General Purpose Graphics Processing Unit*) напредујући ка циљу да се на графички чип пренесе што више опште обраде података која захтева високе перформансе.

Метод који омогућава обраду података на графичком чипу заснива се на чињеници да је *shader* процесоре могуће програмирати скоро произвољним кодом који може у себи садржати најзахтевније операције обраде података.

Уколико се појединачна површина објекта (један правоугаоник) постави тако да прекрива целу површину екрана или прозора, успоставља се бијекцијска релација између *pixel*-а на екрану и позива програма за обраду тог *pixel*-а, па је могуће тако покренути обраду података на графичком чипу и егзактно очитати резултате кроз боје које су приказане на екрану. Излазни податак после процеса обраде су компонентне боје - црвена, зелена и плава, а пошто свака може имати 256 различитих вредености свака носи 8 бита. На крају процедуре обраде, резултати нумеричких операција могу се спаковати у излазне боје и после прочитати са екрана.

Иако је то перспективно добар приступ, недавно су развијене методе директног програмирања графичких процесора за обраду континуалних низова података, те је ова метода нашла адекватну замену.

РАД У *BLENDER 3D MODELER*-У

У контексту коришћења перформанси *shader* процесора и савремених могућности приказа слике помоћу графичких картица,у овом поглављу пажња ће бити усмерена на примену у оквиру *Blender 3D modeler*-а, јер овај програм представља платформу која омогућава једноставан рад са *3D* објектима и ефикасно програмирање *GLSL* програма.

Blender је сложен програм који има неколико додатних слојева између графичког процесора и једноставности *Python* скрипти, тако да је могуће из овог окружења, без већег губитка перформанси, програмирати функционалност графичког процесора на различите начине.

Пример примене у архитектури када је модел обложен сликама (текстурама), са задатим *pixel* и *vertex shader* програмима, може се видети на слици бр. 6.

Сл. 6.
Пример примене у архитектури - презентације
Fig 6.
An example of architectural presentation rendering

БРЗА И ДЕТАЉНА *3D* АНИМАЦИЈА

Процес креирања брзе и детаљне *3D* анимације није претерано сложен уколико се овлада свим потребним елементима за рад у *Blender* програму и уколико се разуме механизам рада самог процеса.

Како већина дизајнера, архитеката и пројектаната користи програме као што су *3D Studio MAX*, *Maya*, *SoftImage*, неопходно је превести овако припремљене објекте и анимације у формате погодне за обраду у *Blenderu*. Постоји сасвим довољан број скрипти које омогућавају *Blender*-у да овако припремљене податке увезе и са њима даље ради.

После прикупљања потребних елемената геометрије, информација о камери, светлима, методама сенчења, потребно је извршити извесне припреме како би се добила анимација у реалном времену задовољавајућег квалитета.

Почев од верзије *2.48a Blender*-а, већина основних типова сенчења објеката са којима *Blender* и други програми раде, при методама праћења путање светлосног зрака, како би се постигла реалистичност излазне слике, интегрисана је у *Blender Game Engine* и није неопходно додатно подешавање материјала и писање скрипти да би се обезбедила ова функционалност. Ово је свакако предност, јер у време настајања овог пројекта та опција није постојала, а у складу са најсавременијим трендовима, оправдано ју је користити.

После дефинисања основних материјала и њиховог понашања, потребно је специјалним материјалима, или објектима који имају посебне особине (просијавање светлости, рефлексије, флуоресцентни сјај), додати графичке детаље применом посебно писаних *shader* програма.

GLSL shader програми омогућавају да се испрограмира произвољан ефекат на материјалу одређеног објекта у сцени. Постоји известан број већ јавно доступних алгоритама за *GLSL* програме који покривају ефекте као што су рефлексије и прозрачност воде, стакла, текстурираност дрвета, мермера, гранулираност и рефлексивност површина метала и слично.

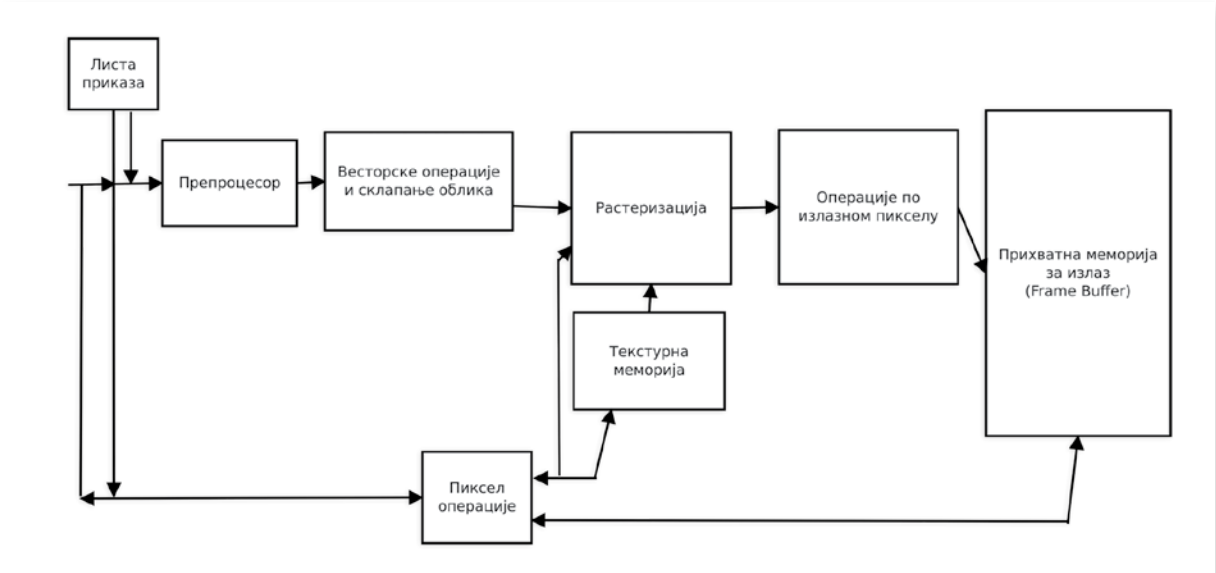
Python скрипте у оквиру *Blender*-а омогућавају директан приступ особинама објеката у сцени, између осталих и *shader* програмима који су за дати објекат дефинисани.

Довољно је у *Python* скрипту убацити жељени *GLSL* код, доделити га имену објекта са којим се ради и проследити му вредности улазних параметара (различите коефицијенте који дефинишу крајњи излаз процеса рендеринга) (*Rost*, 2006). Ова функционалност може се лако енкапсулирати у једну скрипту са графичким интерфејсом, која би омогућавала да се графички бирају особине и дефинишу променљиве за *shader* програме, те тако уклони потреба за текстуалним уношењем параметара.

Када имамо овако дефинисану сцену, потребно је само подесити неколико основних опција *Blender Game Engine*-а, и припремити хватање садржаја екрана, како би се добила детаљна анимација у реалном времену. Шематски приказ овог поступка може се видети на слици 7.

Сл. 7.
Шема поступка припреме брзе и детаљне 3D анимације

Fig 7.
Complete and fast 3D animation - schematics



Сл.8.
Фотореалистична графика у реалном времену
Fig. 8.
Realistical graphics in real time

Овакве методе омогућавају сачињавање детаљне *3D* анимације у времену које је више десетина пута краће од времена потребног методом праћења путање светлосних зрака, а дају сасвим задовољавајући визуални ефекат – у реалном времену, са потенцијалном интерактивношћу – што се може видети на слици бр. 8.

КОРИШЋЕЊЕ ПОТЕНЦИЈАЛА ГРАФИЧКИХ ПРОЦЕСОРА

У светлу развоја нових архитектура графичких процесора, многе њихове функције и могућности примене још увек су у фази истраживања, тј. процес развоја различитих метода приступа овим функцијама и развој напредних апликација способних да упоредо користе снагу централног процесора (*CPU*) и графичког процесора (*GPU*) тек је у повоју.

Услед нарастајућих прохтева индустрије видео игара и истраживачких центара заинтересованих за коришћење овог огромног процесорског потенцијала, дошло је до унапређења у дизајну и начину рада графичких процесора.

Делови са фиксном функционалношћу, који служе за основно приказивање графике и тродимензионалних објеката, у складу са потребама индустрије, сасвим су довољно развијени, али „битка" за гигахерце развила се на пољу процесора за сенчење, чија се примена нагло интензивирала у последњих неколико година.

Пошто класичне архитектуре у којима су постојали посебно дизајнирани процесори за обраду геометријских карактеристика објеката (*vertex* процесори) и елемената излазне слике (*pixel* процесори) нису могле да постигну потребну флексибилност, водеће компаније (*ATI* и *NVIDIA*) развиле су нову архитектуру јединствених *shader* процесора (*Unified Shader Architecture*). Ова архитектура подразумева већи број вишенаменских јединица за обраду геометријских карактеристика и излазне графике, потпуно програмабилних, тако да се могу користити и за једну и за другу намену, по потреби, а притом се и њихови потенцијали програмабилности могу ефикасно користити.

Закључци

Предности и мане новог система

Предности новог система огледају се у могућности да се анимације које су визуелно коректне и графички довољно детаљне за техничку визуелизацију креирају у реалном времену, па ако се начине промене у сцени, резултате је могуће одмах проверити или представити у некој презентацији.

Blender 3D modeler као платформа нуди многе погодности у које спадају: рад са различитим форматима улазних докумената, једноставност алата за уређење сцене и моћан мотор за приказ анимације у реалном времену. Производ је заједнице отвореног кода и лако доступан без надокнаде на практично свим оперативним системима и процесорским архитектурама. Као производ заједнице која је стално у контакту са потребама корисника, брзо се развија, узимајући у обзир потребе корисника и њихове предлоге за додавања нових могућности, нудећи могућност свакоме да приступи изворном коду апликације и измени апликацију, прилагођавајући је својим потребама.

Мана примене методе нацртне и аналитичке геометрије за приказ анимација јесте што се не могу постићи сви ефекти које производи поступак *raytracing*-а, при чему је најочигледнија одсустност пројектовања сенки на објекте иза осветљеног објекта, без додатног програмирања система за рендеринг тродимензионалне анимације које некада може

бити специфично за сваку сцену посебно. Приликом увоза различитих формата докумената, у *Blender*-у није могуће постићи апсолутну веродостојност сцене, јер различити програми за рад са тродимензионалном графиком имају различите начине којима баратају са објектима у сцени, па је практично немогуће направити систем који би подржавао све постојеће опције. Ово се може избећи ако се сцена од почетка припрема у *Blender 3D modeleru.*

Без обзира на недостатке, нови метод рада са тродимензионалним анимацијама нуди функционалну и заокружену целину која у многим случајевима представља добру алтернативу класичним методама.

Перспективе

Прве примене графичких процесора за обраду података настале су са појавом *pixel* процесора у графичким картама које су припадале генерацији *ATI Radeon 9600* и *NVIDIA GeForce FX5200*. *Pixel* процесори су још пре пар година имали могућност рада са произвољним програмским кодом, али ову могићност није било могуће директно користити .

У оквиру примене *pixel* процесора за оно за шта су намењени (додатни прорачуни графике на површинама геометријских објеката), коришћена је специјална метода како би се омогућила прецизна комуникација са процесором.

Геометријска поставка сцене тродимензионалне анимације сачињена је тако да је преко излазног прозора (који може бити и виртуелан, тј. не мора имати излаз директно на екран) постављен један правоугаоник који покрива целу површину, тако да се пројекције његових тачака пресликавају бијекцијски на излазне тачке (*pixel*-е). Пошто ће програм за обраду слике бити позван за сваку излазну тачку, а улаз се може обезбедити кроз податке у слици која је на основи почетног правоугаоника, излазна слика у ствари представља излазне податке, те се у боје слике могу уписати резултати прорачуна.

Ова метода има неколико мана – пре свега, компликована је за употребу и излазни резултати не могу покривати довољан број различитих типова података, а и број цифара резултата је смањен или се мора додатно проширивати „мапирањем" више тачака за један излазни податак.

Развијајући нову генерацију графичких процесора, компаније *ATI* и *NVIDIA* омогућиле су програмерима да користе предности процесора које су развиле додајући посебне механизме управљања меморијом и улазним и излазним подацима који нису везани за обраду графике, те су тако омогућиле дефинисање 3 кључна елемента брзе обраде података на графичком чипу – слање улазног низа, извршавање програмског кода и прикупљање излазних података.

Напомена аутора:

Тамо где је било неопходно, коришћени су страни изрази у изворној форми како би се читаоцима омогућило да путем интернет мреже, користећи претраживаче, лакше пронађу додатне информације о програмима и пројектима о којима се говори у овом тексту.

Аутор се захваљује Проф. др Николи Клему на свесрдној сарадњи и подршци у писању рада, на бројним сугестијама и корекцијама које су аутору омогућиле да рад приведе крају и прилагоди га развоју будућег пројекта који се односи на примену графичких процесора за проналажење решења великих система једначина.

Посебна захвалност се упућује и организацији *International Center for New Media*, као и организаторима конференције *Europrix Multimedia Awards 2008*. за разумевање значаја пројекта рендеринга анимација у реалном времену поготово у време када је реализован (2006. до 2008. године), јер је тада таква технологија била скоро потпуно недоступна широј јавности (у дизајнерском окружењу).

Литература и извори

1.Astle D., K. Hawkins, (2002): OpenGL Game Programming, Course Technology PTR, 1 edition
2. Ebert D.S., K. Musgrave, D. Peachey ,K. Perlin , S. Worley, (2002): Texturing and Modeling: A procedural approach, Morgan Kaufmann, 3rd edition
3. Rost R.J. (2006): OpenGL Shading Language, Addison-Wesley Professional, 2nd edition
4. Shreiner D., M. Woo, J. Neider, T. Davis, (2008): Open GL Programming Guide, Addison Wesley, 6th edition

Интернет стране:

1. Интернет презентација Blender фондације: http://www.blender.org/development/architecture/
http://www.blender.org
2. Интернет сајт посвећен Blender 3D modeleru, садржи форум, упутства и галерију радова. http://blenderartists.org/
3. Интернет страна посвећена основама и историјату рендеринга:
http://en.wikipedia.org/wiki/Rendering_%28computer_graphics%29