ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

გელა ლომიძე

სამგანზომილებიანი ობიექტების დანაწევრება, ზედაპირის სტრუქტურის გამარტივება და ხმაურის რეგულირება

კომპიუტერული მეცნიერებები

სამაგისტრო ნაშრომი შესრულებულია კომპიუტერული მეცნიერებების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ალექსანდრე გამყრელიძე
ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტის

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტის

კომპიუტერული მეცნიერებების მიმართულების

სრული პროფესორი

გოდერძი ფრუიძე
ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტის

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტის მოწვეული ლექტორი,

კიბერნეტიკის ინსტიტუტის მათემატიკური კიბერნეტიკის განყოფილების

მეცნიერ თანამშრომელი

თბილისი, 2013

შინაარსი

ანოტაცია

ბიზნეს სამუშაო პლათფორმების ერთ მესამედს ამოძრავებს სამრეწველო, ან სამეცნიერო აპლიკაციები, რომლებიც ძლიერ დამოკიდებულნი არიან ინტერაქტიულ რეჟიმში სამგანზომილებიანი ობიექტების გამოსახულების გამოტანის ცოდნაზე, რომელთა სირთულეც გაცილებით უფრო სწრაფად იზრდება, ვიდრე გრაფიკული სისტემების ეფექტურობა. სამგანზომილებიან ვიდეო-თამაშებში არსებულ მარტივ სცენებს შეიძლება დასჭირდეს რამდენიმე ასეული ტექსტურირებული მრავალკუთხედი, მაგრამ ობიექტები, რომლებიც გამოიყენებიან მექანიკურ CAD(ავტომატიზირებული პროექტირების სისტემები), სამეცნიერო, სამედიცინო და გეოგრაფიულ აპლიკაციებში მოითხოვენ მილიონობით წახნაგიან სცენებს. დღეისათვის ხელმისაწვდომი მაღალი წარმადობის გრაფიკული აპარატურა ხშირად ვერ ახერხებს სამგანზომილებიანი მოდელების ისეთი კადრების სიხშირით(FPS) გამოტანას, რომ შეგვეძლოს პირდაპირი მანიპულირება და კამერის ინტერაქტიული კონტროლი. ამ პრობლემის გადასაჭრელად საჭიროა მოდელების გამარტივება ისე, რომ არ მოხდეს მათი სერიოზული ცვლილება და დამახინჯება. გრაფიკული მიზნებისათვის გავრცელებული სამგანზომილებიანი ფორმები მრავალწახნაგაა და გამოსახულების სახით გამოტანისათვის ხდება მათი ტრიანგულირება, ამიტომ მნიშვნელოვანია რთული ტრიანგულირებული მოდელების კომპრესიაზე და დეკომპრესიაზე ფოკუსირება.

სამგანზომილებიანი ობიექტის ზედაპირის სტრუქტურული გამარტივება და ხმაურის ჩახშობა არის დაბალი დონის ფუნდამენტური ამოცანა. ამ ამოცანის შედეგები და აპლიკაციები გამოიყენება მრავალ სხვადასხვა დარგში. იდეა მდგომარეობს იმაში, რომ თუ გვაქვს n წვეროიანი ზედაპირი, გამარტივების შემდეგ უნდა მივიღოთ გაცილებით უფრო მცირე წვეროებისგან შემდგარი ზედაპირი, მაგრამ ეს პროცესი სრულდება ისე, რომ არ იკარგება პირველადი ზედაპირის თვისებები. გამარტივება ხდება არა პირდაპირ მთელ ზედაპირზე არამედ დანაწევრების შემდეგ მიღებულ ნაწილებზე.

ნაშრომში აღწერილია საკმაოდ ცნობილი და აქტუალური ალგორითმი, რომელიც გვეხმარება აღნიშნული პრობლემების გადაჭრაში. ჩვენი ძირითადი მიზანია არა ამ ალგორითმის რაიმე სახით გაუმჯობესებაა, თუნდაც თეორიული სახით, არამედ ამ ალგორითმის პროგრამული რეალიზაცია და თეორიული საკითხებისათვის პრაკტიკული

იერსახის მიცემა. შედეგები შეიძლება გამოყენებული იქნას სხვა ალგორითმების მუშაობის დროის ოპტიმიზაციისათვის, ასევე თვითონ პროგრამული უზრუნველყოფა საკმაოდ კარგი დამხმარე საშუალებაა გრაფიკის დიზაინერებისათვის.

Abstract

One third of the workstation business is driven by industrial or scientific applications that heavily depend on the ability to interactively render 3D models whose complexity is increasing far more rapidly than the performance of the graphics subsystems. Simple scenes in 3D video-games may only need a few hundred textured polygons, but models used for mechanical CAD, scientific, geo-science, and medical applications involve scenes with millions of faces. Currently available high-end graphics hardware is often insufficient to render the models at sufficient frame rates to support direct manipulation and interactive camera control. To solve this problem we need model simplification without damaging or changing their structure. Because the prevalent representation for 3D shapes for graphics purposes is polyhedral and be cause polyhedra are in general triangulated for rendering, it is important to focus on the compression and decompression of complex triangulated models.

3D object surface structural simplification and noise reduction is low level fundamental task. The results of this task are used in different fields and applications of computer graphics. Main idea behind the task is that if we have the surface consisting of n vertices, Then after simplification we should get surface with much less vertices without losing structural properties of the primal surface. Simplification process doesn't run on the surface as a whole, It runs on pre-decomposed parts of the surface.

In this paper we describe well known and popular algorithm to solve this problem. Our main goal is to implement its theoretical content and use it in practice, we do not intent to somehow modify or optimize the algorithm. The results can be used to optimize performance of other algorithms. Resulted software will be a useful tool for graphic designers.

შესავალი

მრავალკუთხედი რჩება პოპულარურ გრაფიკულ პრიმიტივად კომპიუტერული გრაფიკის აპლიკაციებისთვის. გარდა ამისა, უბრალო წარმოდგენა რომ გვქონდეს, მრავალკუთხედების კომპიუტერულ ვიზუალიზაციას ფართოდ უჭერს მხარს კომერციული გრაფიკული აპარატურა და პროგრამული უზრუნველყოფა. თუმცა, რადგან მრავალკუთხედი არის წრფივი, ხშირად საჭიროა ათასობით, ან მილიონობით პრიმიტივების გამოყენება რთული გეომეტრიის დეტალიზაციის მისაღწევად. ასეთი ზომის მოდელები საერთოდ არაპრაქტიკულია, რადგან გამოსახულების გამოტანის სიჩქარე და საჭირო მეხსიერება პირდაპირ პროპორციულია მრავალკუთხედების რაოდენობასთან. ამიტომ, აპლიკაციები, რომლებიც აგენერირებენ მრავალკუთხედებით შედგენილ დიდ ბადეებს, ხშირად იყენებენ კონკრეტულ საგანზე ორიენტირებულ ცოდნას, მოდელის ზომის შესამცირებლად. რჩება ალგორითმები, სადაც კონკრეტულ საგანზე ორიენტირებული ოპტიმიზაციის მეთოდები არ არის ხელმისაწვდომი, ან იქნებოდა უადგილო.

ერთ-ერთი ალგორითმი, რომელიც წარმოშობს ბევრ მრავალკუთხედს, არის *მოძრავი კუბების* ალგორითმი. ეს არის უხეში ზედაპირების წარმოქმნის ალგორითმი, რომელიც მოცულობითი მონაცემებიდან გამოყოფს თანაბარი სიმკვრივის ზედაპირებს. ამასთან, ზედაპირის თანამკვეთი თითოეული კუბი დისკრეტული ბადიდან შეიცავს ერთიდან ხუთამდე რაოდენობის სამკუთხედს. მიუხედავად იმისა, რომ მოძრავი კუბების ალგორითმი თავდაპირველად შემუშავებული იყო სამედიცინო აპლიკაციებისთვის, მან უფრო მეტი გამოყენება ნახა სამეცნიერო ვიზუალიზაციაში, სადაც მონაცემთა სიების მოცულობათა ზომა მნიშვნელოვნად ნაკლებია, ვიდრე სამედიცინო აპლიკაციებში. დიდი მოცულობის კომპიუტერული ნაკადის დინამიკას შეიძლება გააჩნდეს სასრული რაოდენობის განსხვავებული ბადე 100x100x100 ხარისხით, მაშინ როდესაც ტიპიური სამედიცინო კომპიუტერული ტომოგრაფია, ან მაგნიტური რეზონანსის სკანერი წარმოქმნის 100-ზე მეტ ნაჭერს 256x256, ან 512x512 გარჩევადობით. ინდუსტრიულ კომპიუტერულს ტომოგრაფიას აქვს უფრო მაღალი გარჩევადობა, კერძოდ 512x512 ან 1024x1024. ასეთი მონაცემთა სიმრავლეებისათვის დონის ზედაპირის მოპოვება მოძრავი კუბების გამოყენებით წარმოქმნის 500-დან 2 მილიონამდე სამკუთხედს. ასეთი ზომის

მოდელების შენახვა და გამოსახულების სახით გამოტანა თანამედროვე გრაფიკულ მანქანებსაც კი უჭირთ.

სხვა შერჩევის მოწყობილობებს შეუძლიათ წარმოქმნან მრავალკუთხედებით შედგენილი დიდი მოდელები: კამერათა სპექტრი, ციფრული მონაცემები რელიეფზე და სატელიტური მონაცემები. ამ მოწყობილობების გარჩევადობა ასევე უმჯობესდება და მიიღება ისეთი მოდელები, რომლებიც კონკურენციას უწევენ სამედიცინო სკანერებით მიღებულ მოდელებს.

ეს ნაშრომი აღწერს დამოუკიდებელი ალგორითმის აპლიკაციას, რომელიც ტრიანგულირებულ ბადეში, სამკუთხედების რაოდენობის შესამცირებლად, იყენებს ლოკალურ ოპერაციებს გეომეტრიასა და ტოპოლოგიაზე. მიუხედავად იმისა, რომ ჩვენი იმპლემენტაცია არის ტრიანგულირებული ბადისათვის, შესაძლოა მისი არაპირდაპირი გამოყენება უფრო ზოგადი, მრავალკუთხედებით შედგენილი, ბადისათვის. ჩვენ აღვწერთ დანაწევრების პროცესს და მის იმპლემენტაციას, ასევე განვიხილავთ შედეგებს, ორი განსხვავებული გეომეტრიული მოდელირების აპლიკაციიდან, რომელიც ილუსტრირებას უკეთებს ალგორითმის სიძლიერეს.

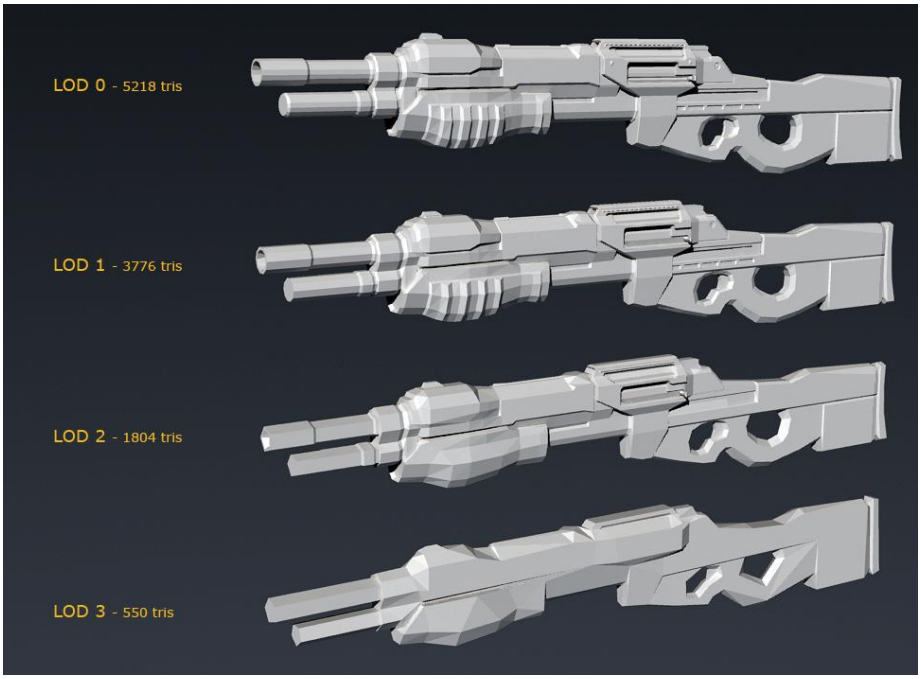
ტრიანგულირებული ბადეების დანაწევრება

1 შესავალი

როგორც უკვე აღვნიშნეთ სამგანზომილებიანი აპლიკაციების უმეტესობა იყენებს კონკრეტულ საგანზე ორიენტირებულ ცოდნას, მოდელთა ზომის შესამცირებლად. ყველაზე დიდი გამოყენება კი ამას აქვს კომპიუტერულ თამაშებში, სხვადასხვა (სამედიცინო, სამხედრო და ა.შ.) სიმულატორებში და ა.შ. სადაც გამოყენებულ კონკრეტულ საგანზე ორიენტირებულ ცოდნად შეიძლება ჩაითვალოს LOD (Level Of Detail) - დეტალიზაციის დონის მეთოდი.

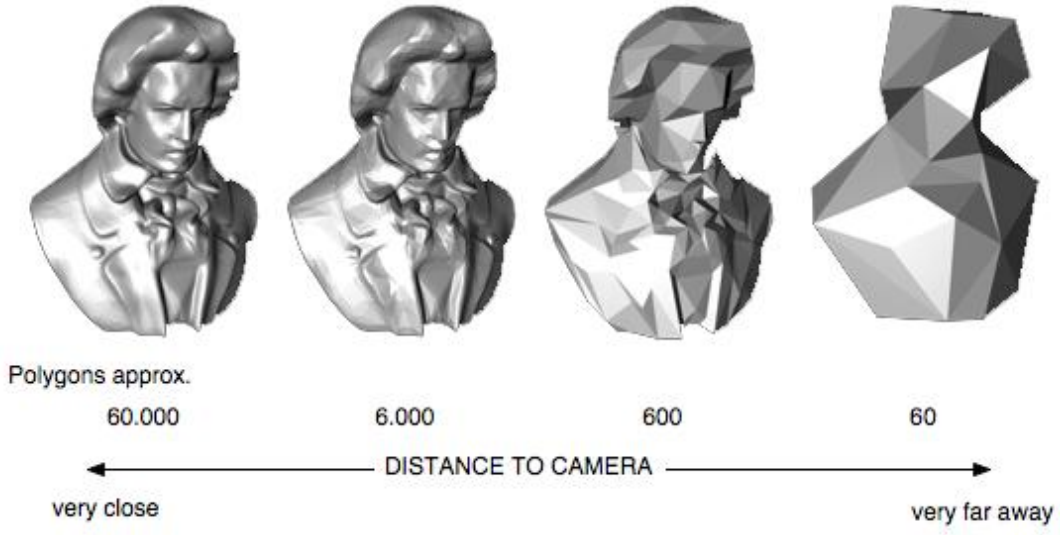
1.1 დეტალიზაციის დონის (LOD) მეთოდი.

რეალისტური სამგანზომილებიანი მოდელები, ბევრ ინტერაქტიულ გრაფიკულ აპლიკაციაში, თავისუფლად შეიძლება შეიცავდეს მილიონობით მრავალკუთხედს - იმაზე ბევრად მეტს, ვიდრე მიმდინარე სამუშაო პლათფორმას შეუძლია გამოსახულების სახით გამოიტანოს ინტერაქტიული კადრების სიხშირეზე. გრაფიკული ამაჩქარებლის მეთოდებს, როგორებიცაა ხედვის არის არჩევა, სცენის ხედვის არეებად დაყოფა და იერარქიული z-ბუფერი, საერთოდ შეუძლიათ მნიშვნელოვნად გააუმჯობესონ გრაფიკული წარმადობა, მაგრამ უმწეონი არიან ისეთ სცენებთან, სადაც ერთდროულად თავმოყრილია დიდი რაოდენობით ობიექტები. სწორედ ასეთი პრობლემების გადასაჭრელად გამოიყენება LOD ტექნიკა, რომელიც ერთ-ერთი ყველაზე გავრცელებულია მსოფლიო მაშტაბით.



http://blenderartists.org/forum/showthread.php?117868-Shotgun

LOD მეთოდის მუშაობის პრინციპიდან გამომდინარე, როდესაც ობიექტი ახლოსაა ხედვის არესთან (მაგ. კამერასთან), ჩვენ გვჭირდება მაღალი გარჩევადობის მოდელი, რამოდენიმე ათეული, ან ასეული ათასი მრავალკუთხედით. ხოლო, როდესაც ობიექტი შორსაა ხედვის არედან, ან რაღაც კუთხითაა მის მიმართ, ჩვენ გვჭირდება დაბალი გარჩევადობის მოდელი, შესაძლოა მრავალკუთხედების რაოდენობა ჩამოვიდეს რამოდენიმე ათეულზე. LOD ტექნიკა იყენებს ერთი ობიექტის სხვადასხვა გარჩევადობის მოდელებს დეტალიზაციის სხვადასხვა დონეზე იმის გათვალისწინებით თუ რა დამოკიდებულება არსებობს ხედვის არესა და ობიექტს შორის.



http://www.opensg.org/projects/opensg/wiki/Tutorial/OpenSG2/NodeCores

LOD-ის მიმდინარე სტანდარტი იყენებს მოდელების განსაზღვრულ სიმრავლეს, როგორც წესი n, n/2, n/4,…. მრავალკუთხედიან მოდელებს, რომლებიც, ან ხელითაა გაკეთებული, ან ორიგინალი მოდელისგან მიიღება ***დანაწევრების*** ალგორითმის საშუალებით. დანაწევრების ალგორითმის გამოყენების შემთხვევაში ვიღებთ მაღალი გარჩევადობის ბაზისურ მოდელს და ვქმნით მიახლოებულ მოდელებს, მანამ სანამ ნარჩუნდება მისი ვიზუალური იერსახე. მოდელები ენაცვლებიან ერთმანეთს იმის მიხედვით თუ რა მანძილია ობიექტსა და კამერას შორის, ან ეკრანზე პროექცირებული გამოსახულების ზომის მიხედვით. საუკეთესო საზომი არის ვიზუალური მნიშვნელობის მინიჭება (მაგ. ობიექტები რომლებიც პირდაპირ ხედვის არეშია უნდა იყვნენ მაღალი გარჩევადობის, ხოლო პერიფერიული ობიექტები კი დაბალი გარჩევადობის), მაგრამ ეს რთულია იმპლემენტაციისათვის.

2 დანაწევრების ალგორითმი

დანაწევრების ალგორითმის მიზანი არის ის, რომ შეამციროს სამკუთხედების საერთო რაოდენობა ტრიანგულირებულ ბადეში, შეინარჩუნოს თავდაპირველი ტოპოლოგია და კარგი მიახლოება ორიგინალ გეომეტრიასთან.

2.1 მიმოხილვა

დანაწევრების ალგორითმი მარტივია. საჭიროა გავიაროთ ბადის ყველა წვეროზე. გავლის დროს, ყოველი წვერო მოიაზრება წაშლის კანდიდატად და თუ ის აკმაყოფილებს დანაწევრების განსაზღვრულ კრიტერიუმს, წვეროც და ყველა სამკუთხედიც, რომელსაც იყენებს ეს წვერო, წაიშლება. შედეგად ბადეში მიღებული ხვრელი გასწორდება ადგილობრივი ტრიანგულაციის ფორმირებით. წვეროების წაშლის პროცესი მეორდება, დანაწევრების კრიტერიუმის საშუალებით, მანამ სანამ რაიმე დასრულების პირობა არ შეგვხვდება. ჩვეულებრივ შეჩერების პირობა განსაზღვრულია, როგორც ორიგინალი ბადის (ან ეკვივალენტის) პროცენტული შემცირება, ან როგორც დანაწევრების რამდენიმე მაქსიმალური მნიშვნელობა. ალგორითმი შედგება შემდეგი სამი ძირითადი ბიჯისაგან:

1. ლოკალური წვეროს გეომეტრიისა და ტოპოლოგიის დახასიათება
2. დანაწევრების კრიტერიუმის შეფასება
3. მიღებული ხვრელის ტრიანგულაცია.

2.2 დამახასიათებელი ლოკალური გეომეტრია/ტოპოლოგია

დანაწევრების ალგორითმის პირველი ბიჯი ახასიათებს მოცემული წვეროს ლოკალურ გეომეტრიასა და ტოპოლოგიას. ამ პროცესის შედეგი განსაზღვრავს, წვერო წარმოადგენს თუ არა წაშლის პოტენციურ კანდიდატს და თუ კი რომელი კრიტერიუმი გამოიყენოს.

ყოველი წვერო შეიძლება მიეკუთვნებოდეს ერთს, შემდეგი ხუთი კლასიფიკაციიდან: მარტივი, რთული, საზღვრის, შიდა წიბოს, ან კუთხის წვერო. ყოველი მათგანის მაგალითი ნაჩვენებია ქვემოთ, სურათზე.

მარტივი წვერო არის გარშემორტყმული სამკუთხედების სრული ციკლით და ყოველი წიბო, რომელსაც იყენებს ის, გამოიყენება ზუსტად ორი სამკუთხედის მიერ. თუ წიბო არ გამოიყენება ორი სამკუთხედის მიერ, ან წვერო გამოიყენება ისეთი სამკუთხედის მიერ, რომელიც არ არის სამკუთხედების ციკლში, მაშინ წვერო არის რთული. ეს ხდება იშვიათ შემთხვევებში.

წვერო, რომელიც მდებარეობს ბადის საზღვართან, ანუ სამკუთხედების ნახევარციკლის ფარგლებში, არის საზღვრის წვერო.

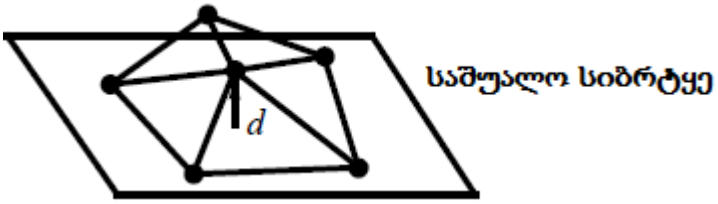
მარტივი წვერო შეიძლება დამატებით იყოს კლასიფიცირებული, როგორც შიდა წიბოს, ან კუთხის წვერო. ეს კლასიფიკაციები ემყარებიან ბადის ლოკალურ გეომეტრიას. თუ ორწახნაგა კუთხე ორ მიმდებარე სამკუთხედს შორის არის უფრო დიდი, ვიდრე განსაზღვრული *დამახასიათებელი კუთხე*, მაშინ *დამახასიათებელი წიბო* არსებობს. როდესაც წვეროს იყენებს ორი დამახასიათებელი წიბო, ის არის შიდა წიბოს წვერო. თუ ერთი, სამი ან მეტი დამახასიათებელი წიბო იყენებს წვეროს, მაშინ ის კლასიფიცირდება კუთხის წვეროდ.

რთული წვეროები არ იშლებიან ბადიდან, რადგან ამ შემთხვევაში ხელახალი ტრიანგულაცია ძალზედ რთულია. ასევე ვინარჩუნებთ კუთხის წვეროებსაც, რადგან ისინი ხშირად განსაზღვრავენ ობიექტის ფორმის მნიშვნელოვან თვისებებს, ხოლო ყველა დანარჩენი ხდება წაშლის კანდიდატი.

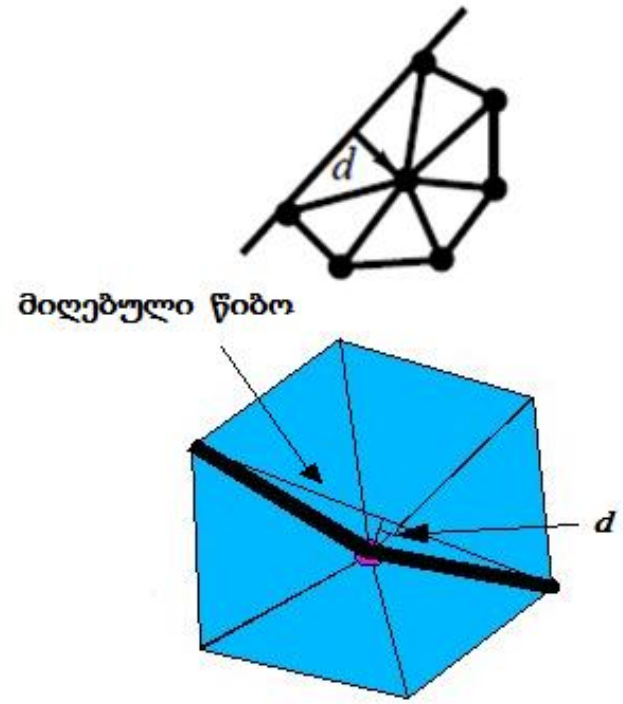
2.3 დანაწევრების კრიტერიუმების შეფასება

დახასიათების ბიჯის შემდგომ, როდესაც უკვე კლასიფიცირებული გვაქვს თითოეული წვერო გადავდივართ შეფასების ბიჯზე, სადაც განვსაზღვრავთ წარმოიქმნება თუ არა წაშლის შეცდომა. ეს გულისხმობს, რომ უნდა განისაზღვროს, შეიძლება თუ არა სამკუთხედებით ფორმირებული ციკლი წაიშალოს და ჩანაცვლდეს სხვა ტრიანგულაციით, ორიგინალი წვეროს გარეშე. მიუხედავად იმისა, რომ დანაწევრების ფუნდამენტურ კრიტერიუმად ვიყენებთ წვეროსა და ზედაპირს შორის მანძილს, ან წვეროსა და წიბოს შორის მანძილს, შეგვიძლია გამოვიყენოთ სხვა მიდგომებიც.

მარტივი წვეროები იყენებენ სიბრტყემდე მანძილის კრიტერიუმს (იხ. სურათი ქვემოთ). თუ წვერო მითითებულია საშუალო სიბრტყემდე მანძილის ფარგლებში, შეიძლება ის წაიშალოს. წინააღმდეგ შემთხვევაში ის შენარჩუნდება. საშუალო სიბრტყე განისაზღვრება, როგორც სიბრტყე, რომელსაც აქვს გასაშუალებული წერტილი და ნორმალი, რომლებიც შესაბამისად მიიღება კანდიდატი წვეროს მეზობელი წვეროებისა და მათი ნორმალების გასაშუალებით.



საზღვრის და შიდა წიბოს წვეროები იყენებენ წიბომდე მანძილის კრიტერიუმს (იხ. სურათი ქვემოთ). ამ შემთხვევაში ალგორითმი განსაზღვრავს მანძილს ახალ წიბომდე, რომელიც მოიცემა ორი მეზობელი საზღვრის წვეროს მიერ, ან იმ ორი მეზობელი წვეროს მიერ, რომელიც მდებარეობს დამახასიათებელ წიბოებზე. თუ წვერო მითითებულია წიბომდე მანძილის ფარგლებში, მაშინ წვერო შეიძლება წაიშალოს.



საერთოდ ორიგინალი მოდელის ძირითადი თვისებებისა და მახასიათებლების შესანარჩუნებლად ვცდილობთ დამახასიათებელი წიბოების შენარჩუნებას, თუმცა ეს ყოველთვის არაა სასურველი. მაგალითად ტრიანგულირებული ბადე შეიძლება შედგებოდეს შედარებით პატარა სამკუთხედების არეებისაგან, დიდი დამახასიათებელი კუთხეებით, შედეგად მივიღებთ შედარებით მცირე გეომეტრიულ მიახლოებას. ან, პატარა სამკუთხედებმა შესაძლოა წარმოქმნან „ხმაური“(დამახინჯება) ორიგინალ ბადეში. ასეთ შემთხვევებში, კუთხის წვეროები, რომლებიც ჩვეულებრივ არ იშლებიან და შიდა წიბოს წვეროები, რომლებიც შეფასებულია წიბომდე მანძილის კრიტერიუმის გამოყენებით, შესაძლოა შეფასდნენ ზედაპირამდე მანძილის კრიტერიუმის გამოყენებით. ჩვენ ამას ვეძახით წიბოს შენარჩუნებას, მომხმარებლის მიერ განსაზღვრული პარამეტრით. თუ წვერო უნდა ამოვარდეს, ეს ხდება მის მიერ გამოყენებული სამკუთხედების წაშლის გზით, ხოლო მიღებული ციკლი უნდა იქნას ტრიანგულირებული.

2.4 ტრიანგულაცია

წვეროს და მასთან დაკავშირებული სამკუთხედების წაშლა ქმნის ერთ (მარტივი ან საზღვრის წვერო) ან ორ ციკლს (შიდა წიბოს წვერო). ყოველ ციკლში უნდა წარმოიქმნას ტრიანგულაცია, რომლის სამკუთხედებიც არ იკვეთებიან და არ არიან გადაგვარებულნი. გარდა ამისა, სასურველია შეიქმნას სამკუთხედები მხარეების კარგი შეფარდებით და რაც შეიძლება კარგი მიახლოება ორიგინალ ციკლთან.

საზოგადოდ, შეუძლებელია გამოვიყენოთ ორგანზომილებიანი ალგორითმი ტრიანგულაციის შესაქმნელად, რადგან საერთოდ ასეთი ციკლი არაპლანარულია. გარდა ამისა, არსებობს ციკლის ორი მნიშვნელოვანი მახასიათებელი, რომელიც შესაძლოა გამოყენებული იქნას უპირატესობისათვის. პირველი, თუ ციკლი არ შეიძლება ტრიანგულირდეს, მაშინ ციკლის წარმომშობი წვერო არ უნდა წაიშალოს. მეორე, რადგან ყველა ციკლი ვარსკვლავის ფორმისაა, ციკლის რეკურსიულ გახლეჩაზე დაფუძნებული ტრიანგულაციის სქემები ეფექტურნი არიან.

ტრიანგულაციის დასრულების შემდგომ, ორიგინალი წვერო და მისი სამკუთხედების ციკლი წაიშლება. ეილერის [8] შეფარდებიდან გამომდინარე, თუ წავშლით მარტივ, კუთხის, ან შიდა წიბოს წვეროს, მაშინ ბადე შემცირდება ზუსტად ორი სამკუთხედით, ხოლო თუ საზღვრის წვეროა წაშლილი, მაშინ ბადე მცირდება ზუსტად ერთი სამკუთხედით.

3 იმპლემენტაცია

ამ თავში განხილულია ზემოთ აღწერილი ალგორითმის იმპლემენტაცია და დაწვრილებით არის აღწერილი ტრიანგულაციის პროცესი.

3.1 მონაცემთა სტრუქტურები

მონაცემთა სტრუქტურა უნდა შეიცავდეს მინიმუმ ორი სახის ინფორმაციას: ყოველი წვეროს გეომეტრია კოორდინატების სახით და ყოველი სამკუთხედის განმარტება მისი სამი წვეროს ტერმინებში. გარდა ამისა, რადგან წვეროს გარშემო სამკუთხედების თანმიმდევრული სიები ხშირად მოთხოვნადია, სასურველია შენარჩუნება იმ სამკუთხედების სიების, რომლებსაც იყენებს თითოეული წვერო.

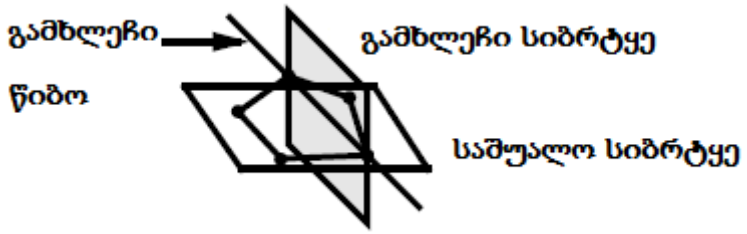
ჩვენი რეალიზაცია იყენებს წვეროსა და სამკუთხედის იერარქიულ წრიულ სტრუქტურას, რომელიც საკმაოდ კომპაქტურია. ეს მონაცემთა სტრუქტურა შედგება იერარქიული მიმთითებლებისაგან, სამკუთხედებიდან წვეროებისკენ და მიმთითებლებისგან, წვეროებიდან უკან სამკუთხედებისკენ. ერთად აღებული ეს ორი მიმთითებელი ქმნის წრიულ კავშირს. ჩვენი იმპლემენტაცია იყენებს სამ სიას: წვეროს კოორდინატების სია, სამკუთხედის განსაზღვრებების სია და ისეთი სამკუთხედების სიების სია, რომლებიც იყენებენ თითოეულ წვეროს. წიბოს განსაზღვრებები არაცხადია, ისინი ცხადად განისაზღვრება, როგორც წვეროთა დალაგებული წყვილები სამკუთხედში.

3.2 ტრიანგულაცია

მიუხედავად იმისა, რომ შეიძლება სხვა ტრიანგულაციის სქემების გამოყენებაც, ჩვენ ავირჩიეთ და იმპლემენტაცია გავუკეთეთ ციკლის გახლეჩვის რეკურსიული პროცედურა. ყოველი ციკლი, ტრიანგულაციისათვის, გაყოფილია ორ ნაწილად. გაყოფა ხდება, ციკლში ორი არამეზობელი წვეროდან განსაზღვრული წიბოს საშუალებით (გამყოფი წიბო). ყოველი ახალი ციკლი იყოფა თავიდან, მანამ სანამ მხოლოდ სამი წვერო

არ დარჩება ყოველ მათგანში. სამი წვეროსაგან შემდგარი ციკლი წარმოქმნის სამკუთხედს, რომელიც შესაძლოა დაემატოს ბადეს და დაასრულებს რეკურსიის პროცესს.

იმის გამო, რომ ციკლი არის არაპლანარული და ვარსკვლავის ფორმის, მისი გახლეჩა გამოითვლება გამხლეჩი ზედაპირით. გამხლეჩი ზედაპირი, როგორც ნაჩვენებია სურათზე ქვემოთ, ორთოგონალურია საშუალო ზედაპირის მიმართ, რომელიც შეიცავს გამყოფ წიბოს. იმისათვის, რომ განვსაზღვროთ წარმოშობს, თუ არა გახლეჩა ორ არაგადამფარავ ციკლს, გამხლეჩი ზედაპირი გამოიყენება ნახევარი სივრცის შესადარებლად. თუ კანდიდატი ციკლის ყველა წერტილი არის გამყოფი ზედაპირის ერთ მხარეს, მაშინ ორი ციკლი არ ფარავს ერთმანეთს და გამყოფი ზედაპირი მისაღებია. რა თქმა უნდა, ადვილია ისეთი მაგალითების შექმნა, სადაც ეს ალგორითმი ვერ შეძლებს წარმატებული გახლეჩვის წარმოებას. ასეთ შემთხვევებში ჩვენ უბრალოდ მივუთითებთ ტრიანგულაციის პროცესის ჩავარდნაზე და წვეროს არ ვშლით.



როგორც წესი, ყოველი ციკლი შეიძლება გაიხლიჩოს ერთზე მეტი გზით. ამ შემთხვევაში საუკეთესო გამყოფი ზედაპირი უნდა აირჩეს. მიუხედავად იმისა, რომ ბევრი შესაძლო ვარიანტი არსებობს, ჩვენ წარმატებით ვიყენებთ მხარეთა შეფარდებაზე დაფუძნებულ კრიტერიუმს. მხარეთა შეფარდება განისაზღვრება, როგორ ციკლის წვეროებიდან გამხლეჩ ზედაპირამდე მანძილებს შორის მინიმალურის შეფარდება გამხლეჩი წიბოს სიგრძესთან. საუკეთესო გამხლეჩი ზედაპირი არის ის, რომელიც გვაძლევს მაქსიმალურ მხარეთა შეფარდებას. ამ შეფარდების შემოსაზღვრა ისე, რომ ის იყოს განსაზღვრულ მნიშვნელობაზე მეტი, მაგალითად 0.1-ზე მეტი, გვაძლევს დასაშვებ ბადეს.

რასაკვირველია განსაკუთრებული შემთხვევები შესაძლოა წარმოიშვას ტრიანგულაციის პროცესში. განმეორებითმა დანაწევრებამ შესაძლოა წარმოშვას უბრალო დახურული ზედაპირი, ისეთი როგორიცაა ტეტრაედრი. წვეროების გაუქმება ამ შემთხვევაში ცვლის ბადის ტოპოლოგიას. სხვა განსაკუთრებული შემთხვევა ხდება, როდესაც „გვირაბები“, ან ტოპოლოგიური ხვრელები გვაქვს ბადეში. გვირაბი შესაძლოა საბოლოოდ შემცირებულ იქნას სამკუთხედამდე გადამკვეთ ნაწილში. წვეროს მოშორება გვირაბის საზღვრიდან, შემდეგ აშორებს გვირაბს და ქმნის იშვიათ შემთხვევას.

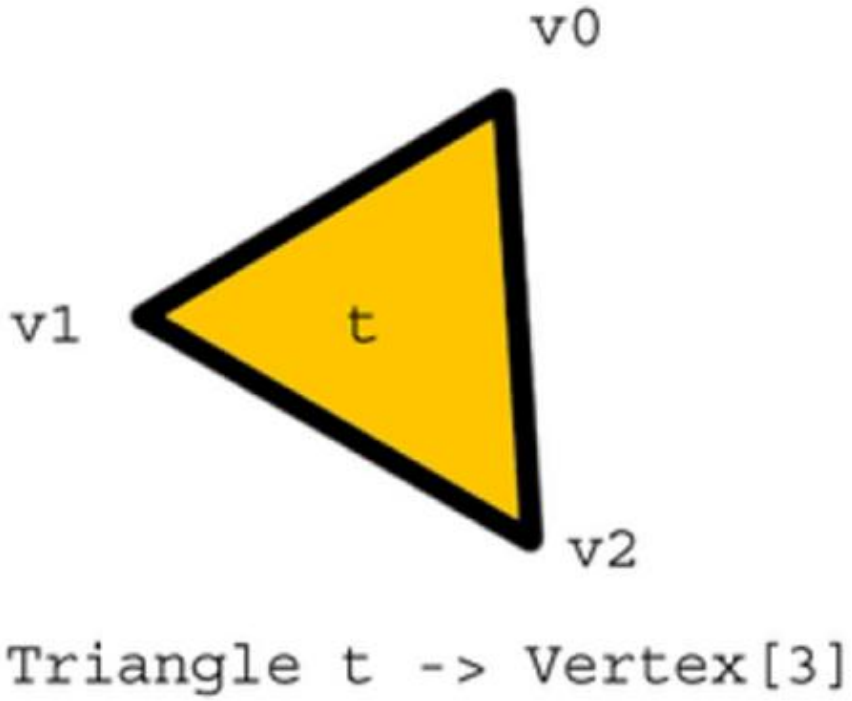
ეს შემთხვევები მუშავდება ტრიანგულაციის პროცესში. როგორც კი იქმნება ახალი სამკუთხედი, ტარდება შემოწმებები იმაში დასარწმუნებლად, რომ დუბლიკატი სამკუთხედები და სამკუთხედის წიბოები არ იქმნება. ეს ინარჩუნებს ორიგინალი ბადის ტოპოლოგიას, რადგან მის სხვა ნაწილებთან ახალი კავშირები არ არსებობს.

4 დანაწევრების პროგრამა

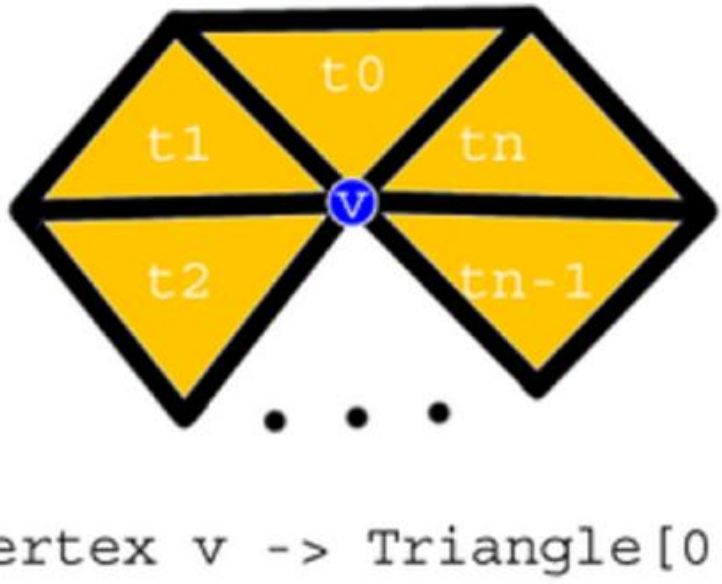
ჩვენი პროგრამა რეალიზებას უკეთებს ზემოთ განხილულ დანაწევრების ალგორითმს. კერძოდ, ის იყენებს სიბრტყემდე მანძილის ფორმულას და არ პოულობს რთულ წვეროებს, რადგან ისინი წასაშლელად არ გვჭირდება. ჩვენს პროგრამას აქვს წიბოსა და კუთხის მარტივი დეტექტორი.

4.1 მონაცემთა სტრუქტურები

ყველაზე იოლი გზა დანაწევრების ალგორითმის იმპლემენტაციისათვის არის წრიული სტრუქტურის შექმნა. ის შედგება მონაცემთა სტრუქტურისგან, რომელშიც სამკუთხედებმა იციან, რომელ წვეროებს იყენებენ (სურათი 4.1) და წვეროებმა იციან, რომელი სამკუთხედების ნაწილნი არიან (სურათი 4.2). ამას მივყავართ წრიულ სტრუქტურამდე, რომელიც საშუალებას აძლევს დანაწევრების ალგორითმს წაშალოს წვეროები და იოლად გაუკეთოს რეკონსტრუირება ტრიანგულირებულ ბადეს.



სურათი 4.1: Triangle-ის მონაცემთა სტრუქტურას აქვს 3 Vertex ობიექტის სია



სურათი 4.2: Vertex-ის მონაცემთა სტრუქტურა აქვს Triangle ობიექტების სია.

4.2 ალგორითმი

ჩვენი დანაწევრების ალგორითმი შლის წვეროებს ბადის სივრციდან, რომლებიც შედარებით ბრტყელნი არიან. წვეროს აქვს მეზობელი წვეროების სია. ჩვენ განვსაზღვრავთ მეზობლებს, როგორც სხვა წვეროებს მთელ სიაში, რომლებიც დაკავშირებულნი არიან წვეროსთან სამკუთხედის წიბოს საშუალებით. საშუალო სიბრტყე შეიძლება გამოითვალოს ამ მეზობლებიდან. ეს არის სიბრტყე, რომელიც წარმოადგენს ყველაზე ახლო მიახლოებას ყველა მეზობელთან წერტილ-ნორმალის ფორმით. წერტილი წარმოადგენს მეზობლების საშუალო წერტილს (P) და ნორმალი არის საშუალო მათ ნორმალებს შორის (N). P მოიძებნება მეზობლების ყოველი კომპონენტის (x, y და z) გასაშუალებით; N მოიძებნება მათი ნორმალების ყოველი კომპონენტის (x, y და z) გასაშუალებით.

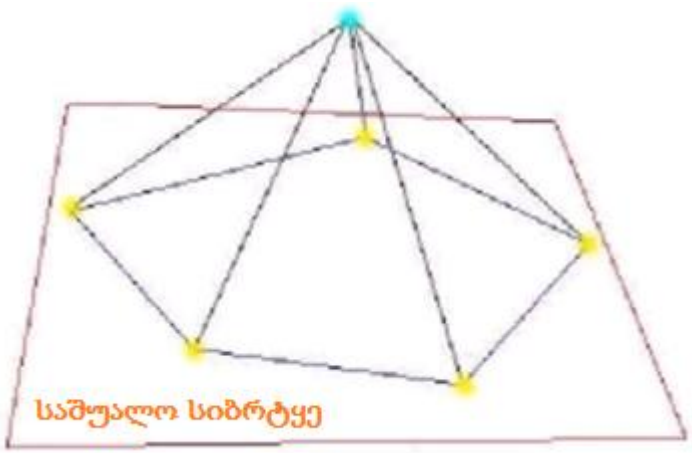
$$d = V \cdot N - p \cdot N \quad (4.1)$$

საშუალო სიბრტყის პოვნის შემდეგ გამოითვლება მანძილი წაშლის კანდიდატ წვეროსა და სიბრტყეს შორის. თუ მანძილი მოთავსებულია მოცემულ ზღვრულ დიაპაზონში, მაშინ წვერო წაიშლება. ზღვრული დიაპაზონი მოიცემა, ან მომხმარებლის მიერ შეტანილი პარამეტრით, ან წვეროებისათვის გამოთვლილი მანძილებს შორის უმცირესით. ჩვენს შემთხვევაში ეს მოიცემა მეორე შემთხვევით. მანძილი გამოითვლება ფორმულით (4.1), სადაც V არის წაშლის კანდიდატი წვერო, P არის სიბრტყეზე პროექცია ამ წერტილში და N არის ზედაპირის ერთეულოვანი ნორმალი.

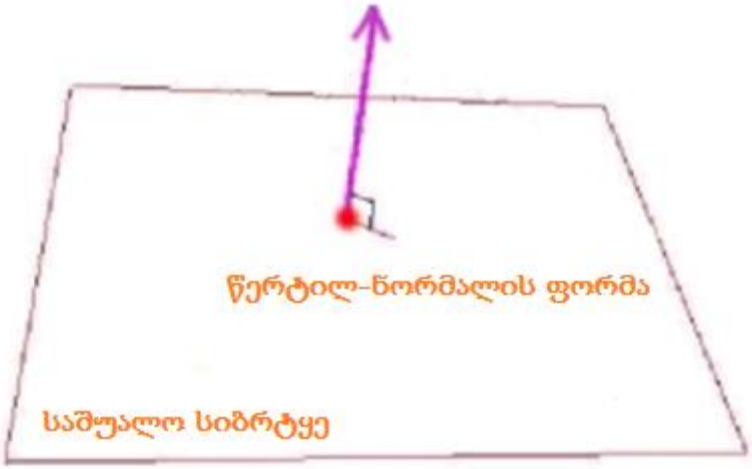
ქვემოთ მოცემულია სურათები, სადაც ნაჩვენებია თუ როგორ გამოითვლება სიბრტყემდე მანძილი.

ბიჯი 1, სურათი 4.3: საშუალო სიბრტყის მიღება, წერტილ-ნორმალის თვალსაზრისით (სურათი 4.4).

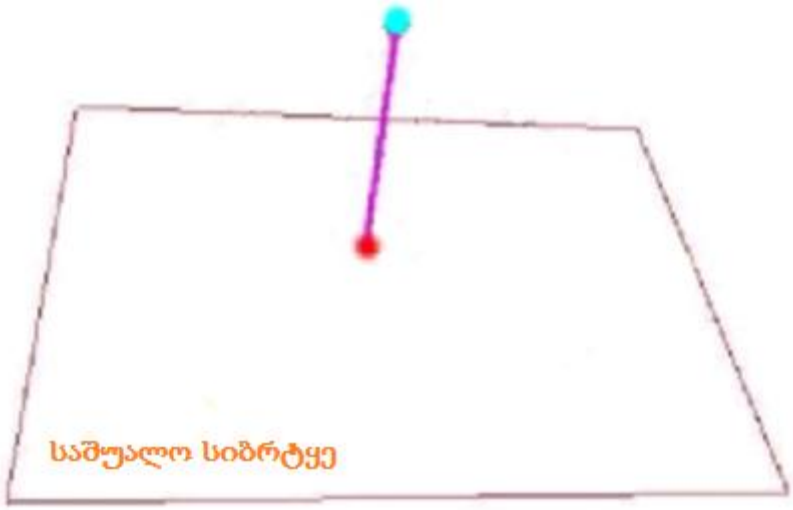
ბიჯი 2, სურათი 4.5: წვეროდან სიბრტყის წერტილამდე მანძილის მიღება. ეს არის სიბრტყემდე მანძილი და შეიძლება გამოისახოს როგორც განტოლება (4.1).



სურათი 4.3 საშუალო სიბრტყის პოვნა



სურათი 4.4: საშუალო სიბრტყე წერტილ-ნორმალის ფორმით



სურათი 4.5: წვეროდან სიბრტყემდე მანძილის გამოთვლა

ჩვენი ალგორითმი ითვლის სიბრტყემდე მანძილს ყველა წვეროსათვის, თავდაპირველად ფაილის ჩატვირთვისას და ასევე მისი მეზობელი წვეროდან ერთ-ერთის წაშლის დროს. ნებისმიერ დროს, როცა მეზობელი წვერო წაიშლება სიიდან, საშუალო სიბრტყეც შეიცვლება და მანძილს ჭირდება თავიდან გამოთვლა, ახალი საშუალო სიბრტყის გამოყენებით. მიზეზი იმისა, რომ ჩვენ თავიდან ვითვლით მანძილს, როდესაც მეზობელი წვერო იშლება არის ის, რომ გვქონდეს ყოველთვის მიმდინარე მანძილი.

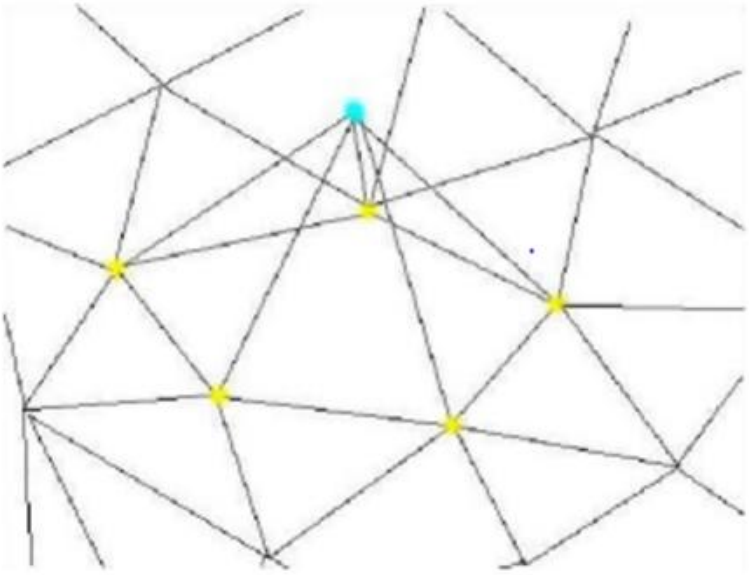
ქვემოთ წარმოდგენილია სურათები, რომლებიც წარმოადგენენ მეთოდს, რომელიც გამოიყენება სიიდან ერთი წვეროს წასაშლელად.

ბიჯი 1, სურათი 4.6: საუკეთესო კანდიდატის მოძიება მოდელში. ჩვენი ალგორითმი გაივლის წვეროების მთელ სიას და მოძებნის წვეროს სიბრტყემდე ყველაზე პატარა მანძილით, რომელიც ჩვენ წინათ აღვნიშნეთ როგორც d.

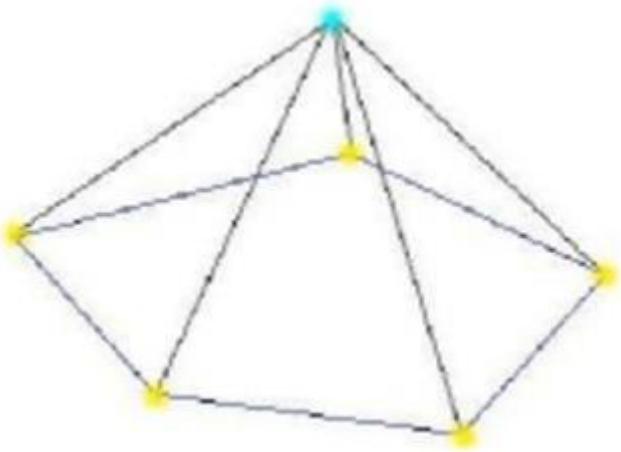
ბიჯი 2, სურათი 4.7: საუკეთესო კანდიდატი წვეროდან მეზობლების სიის მიღება. ეს შეიძლება შესრულდეს იოლად, რადგან ჩვენი წვეროს მონაცემთა სტრუქტურა ინახავს თავისი მეზობლების მიმდინარე სიას, ადვილად განმეორებად STL სიმრავლეში.

ბიჯი 3, სურათი 4.8: ხელახალი ტრიანგულაცია. ჩვენი ხელახალი ტრიანგულაციის ალგორითმი ძირითადია. ის ექსპლუატაციას უკეთებს მოცემულს მრავალგვაროვან მოდელში. მოცემული არის ის, რომ თუ ჩვენ შევცვლით საუკეთესო კანდიდატ წვეროს ერთ-ერთი მისი მეზობლით, ყველა მეზობელში ხდება ორი რამ. მოდელი ინარჩუნებს მრავალფეროვნებას და ორი სამკუთხედი იშლება. წინა სურათზე იყო ხუთი სამკუთხედი. ახლა გვაქვს სამი.

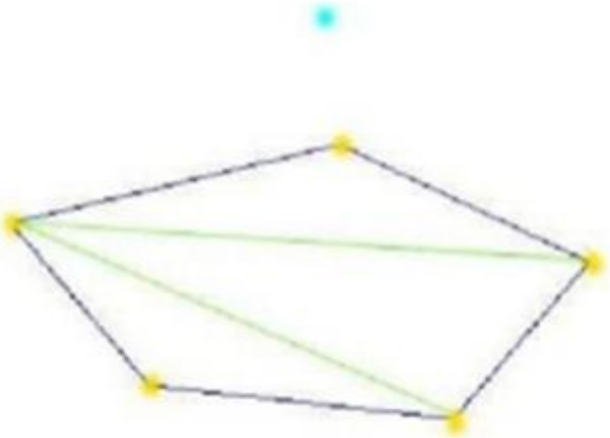
ბიჯი 4, სურათი 4.9: საბოლოოდ ჩვენ ვშლით წვეროს მოდელიდან. მოდელს ახლა აქვს ერთით ნაკლები წვერო და ორით ნაკლები სამკუთხედი.



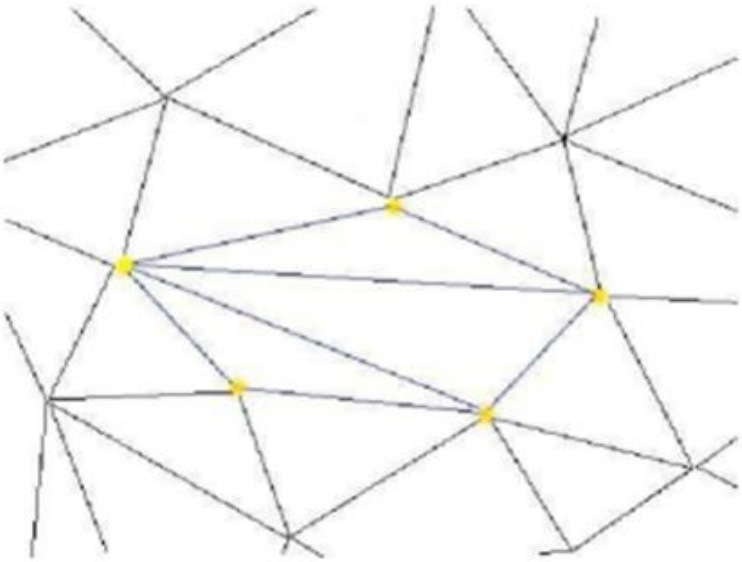
სურათი 4.6: წაშლის კანდიდატის პოვნა



სურათი 4.8: მეზობლების სიის მიღება



სურათი 4.8: ხელახალი ტრიანგულაცია



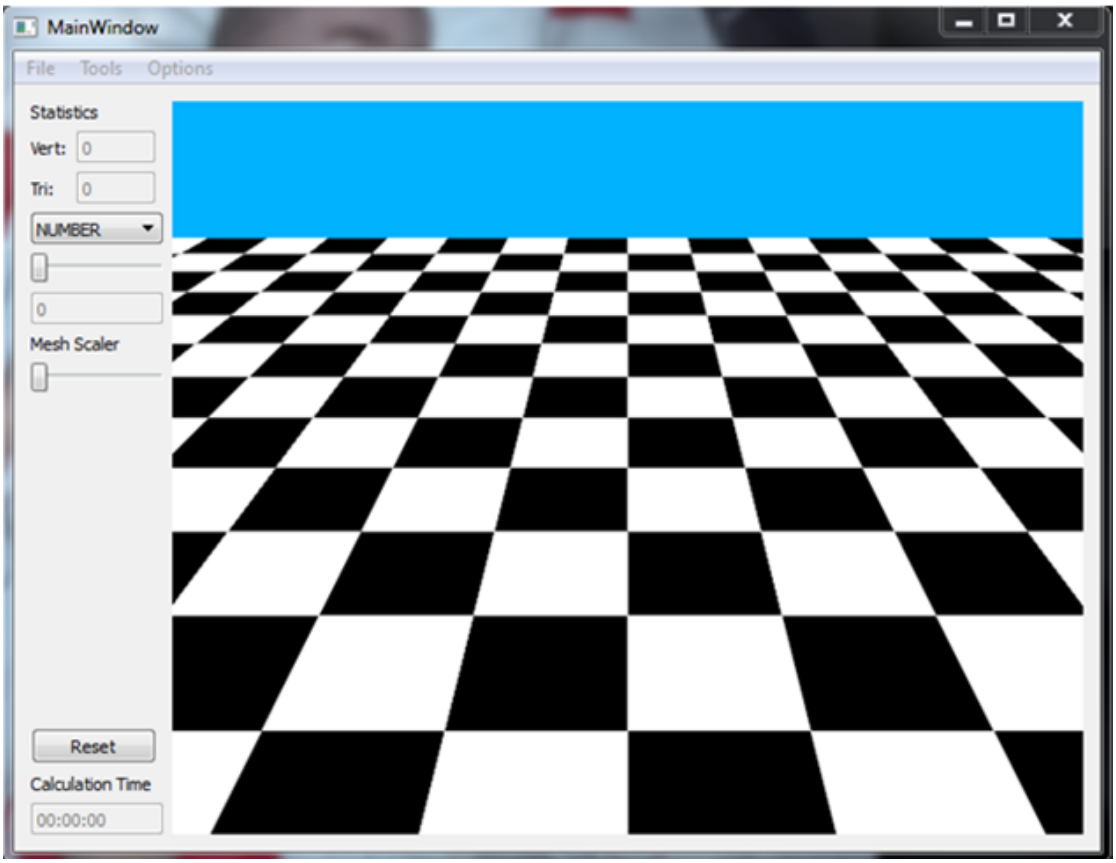
სურათი 4.9: წვეროს წაშლა

ზემოთ აღწერილი ბიჯები წარმოადგენენ ერთ სრულ ციკლს. ეს ციკლი მეორდება იმდენჯერ, რამდენი წვეროს წაშლაც უნდა მომხმარებელს. გამეორება ცნობილია, როგორც დანაწევრება.

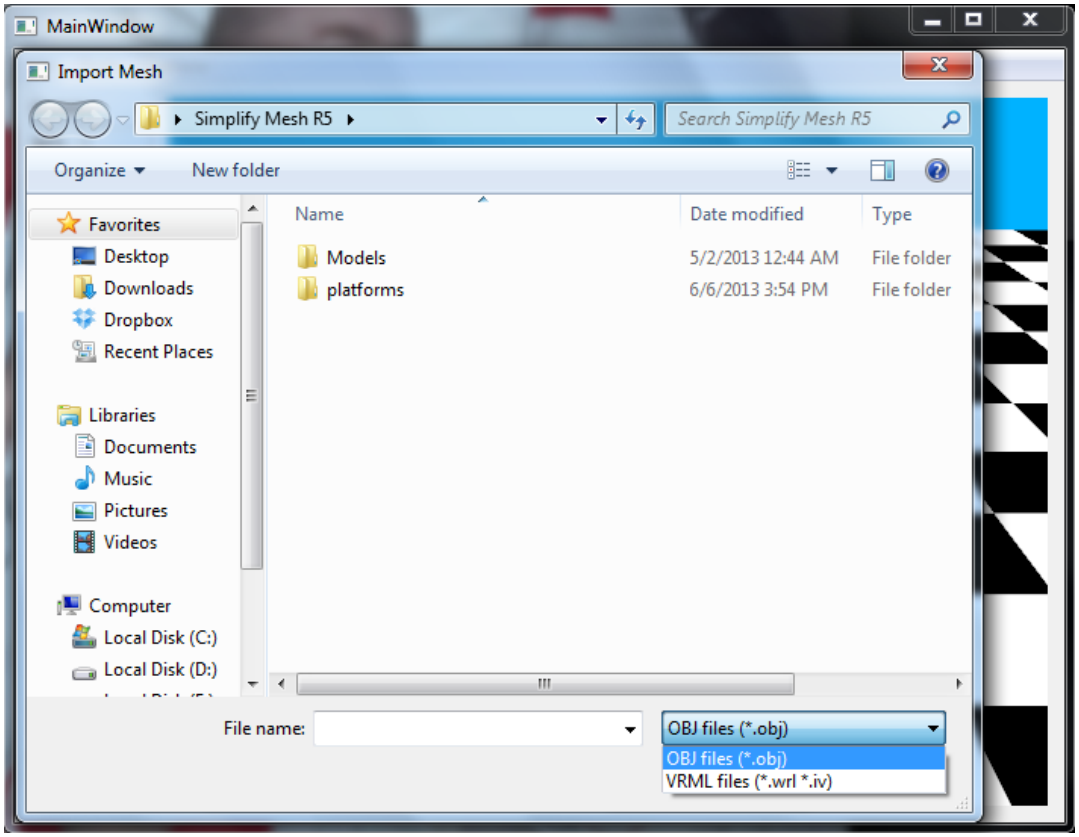
4.3 ინტერფეისი

აღნიშნული პროგრამა წარმოადგენს Windows აპლიკაციას. დიზაინი შექმნილია Qt Designer-ის საშუალებით. ალგორითმი რეალიზებულია C++ - ობიექტზე ორიენტირებულ პროგრამირების ენაზე, სიების შესანახად გამოყენებულია STL ბიბლიოთეკა და გრაფიკული მხარდაჭერისთვის გამოყენებულია OpenGL - გრაფიკული პროგრამირების სტანდარტული API. ეს ყველაფერი კი გაერთიანებულია და რეალიზებულია MS Visual Studio 2010 - Microsoft-ის ინტეგრირებულ განვითარების გარემოში. იმის გათვალისწინებით, რომ ყველა ეს ზემოთ ჩამოთვლილი პროგრამული უზრუნველყოფა მულტიპლათფორმულია, იოლია ამ Windows აპლიკაციის შეცვლა სხვა ოპერაციული სისტემებისათვის (მაგ. Linux). პროგრამა მუშაობს Windows XP, Windows Vista, Windows 7 და Windows 8 ოპერაციულ სისტემებზე.

რეალიზებული პროგრამა თავდაპირველი გაშვებისას ნაჩვენებია შემდეგ სურათზე.

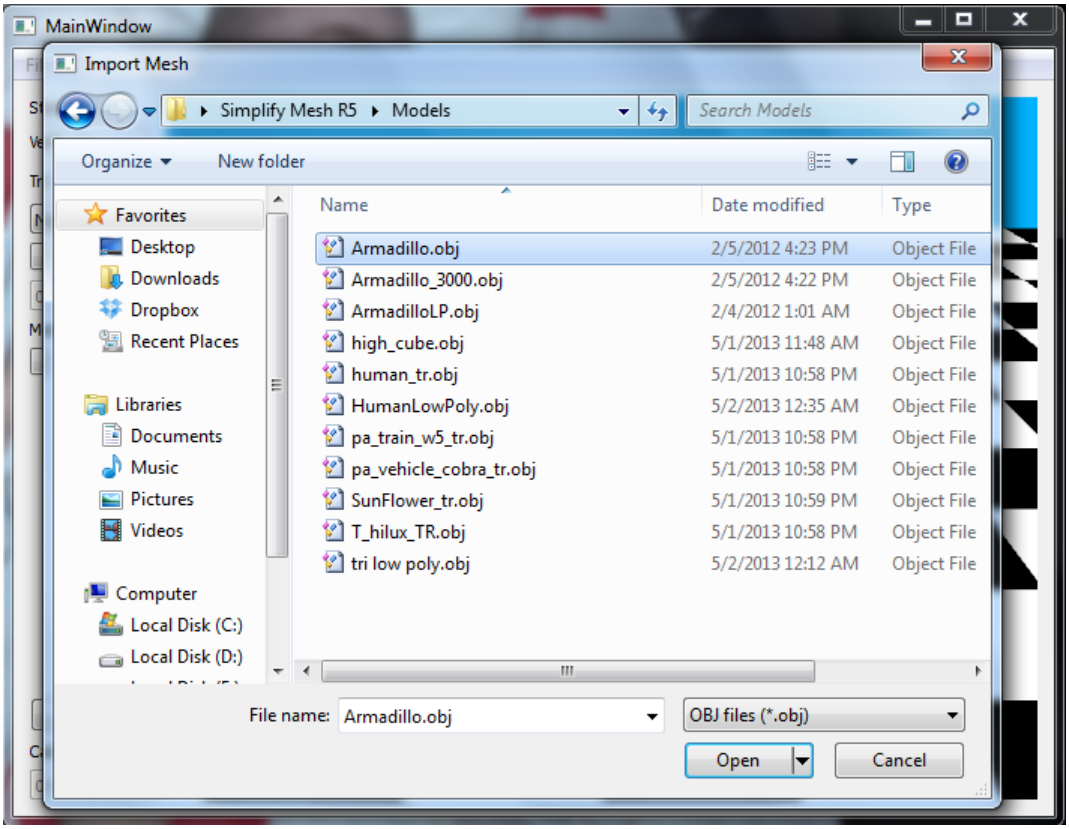


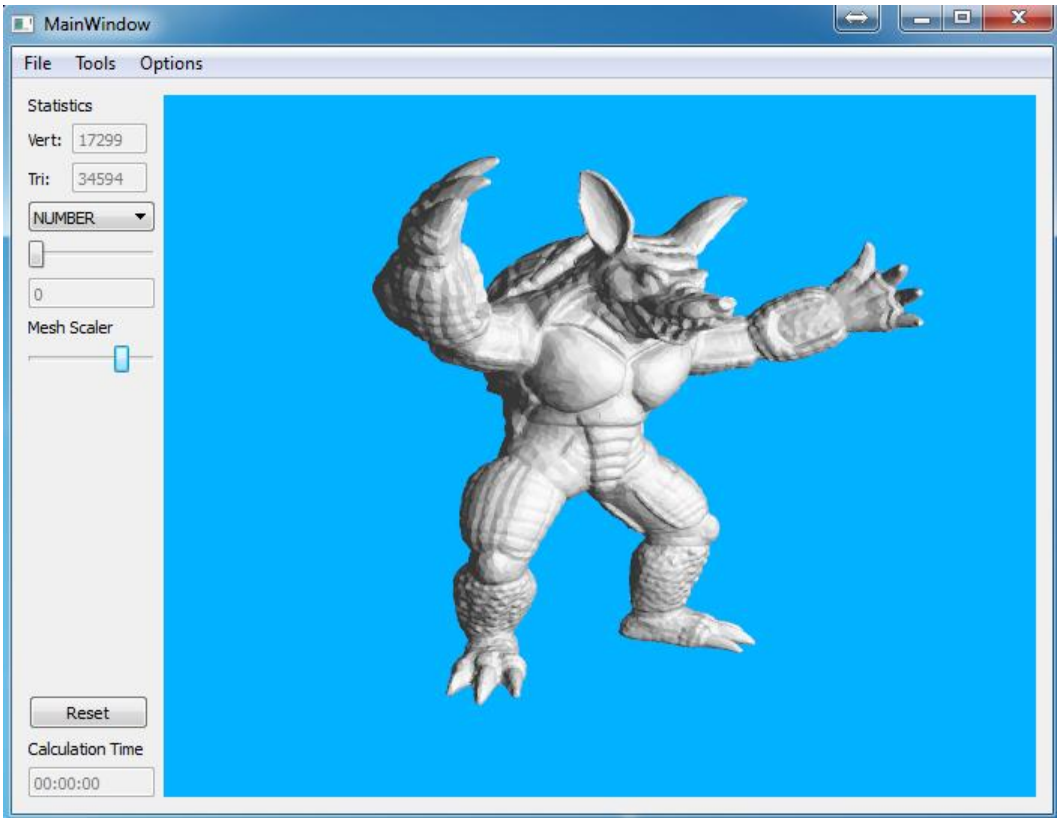
მონაცემების შეტანა ხდება File->Import Model და გამოდის შემდეგი ფანჯარა



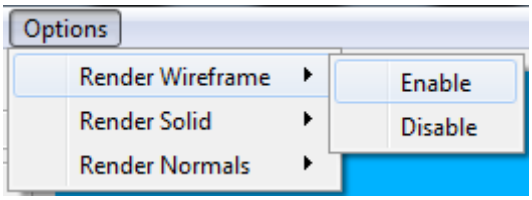
სადაც საშუალება გვეძლევა ავირჩიოთ სხვადასხვა გაფართოების (.obj, .wrl და .iv) ფაილები. ამ გაფართოების ფაილების წასაკითხად ვიყენებთ შესაბამის სინტაქსურ ანალიზატორს - .obj-სათვის OBJParser-ს, ხოლო .wrl-სა და .iv-სათვის VRMLParser-ს, რომლებიც ასევე ჩვენ მიერ რეალიზებულია C++ ენაზე.

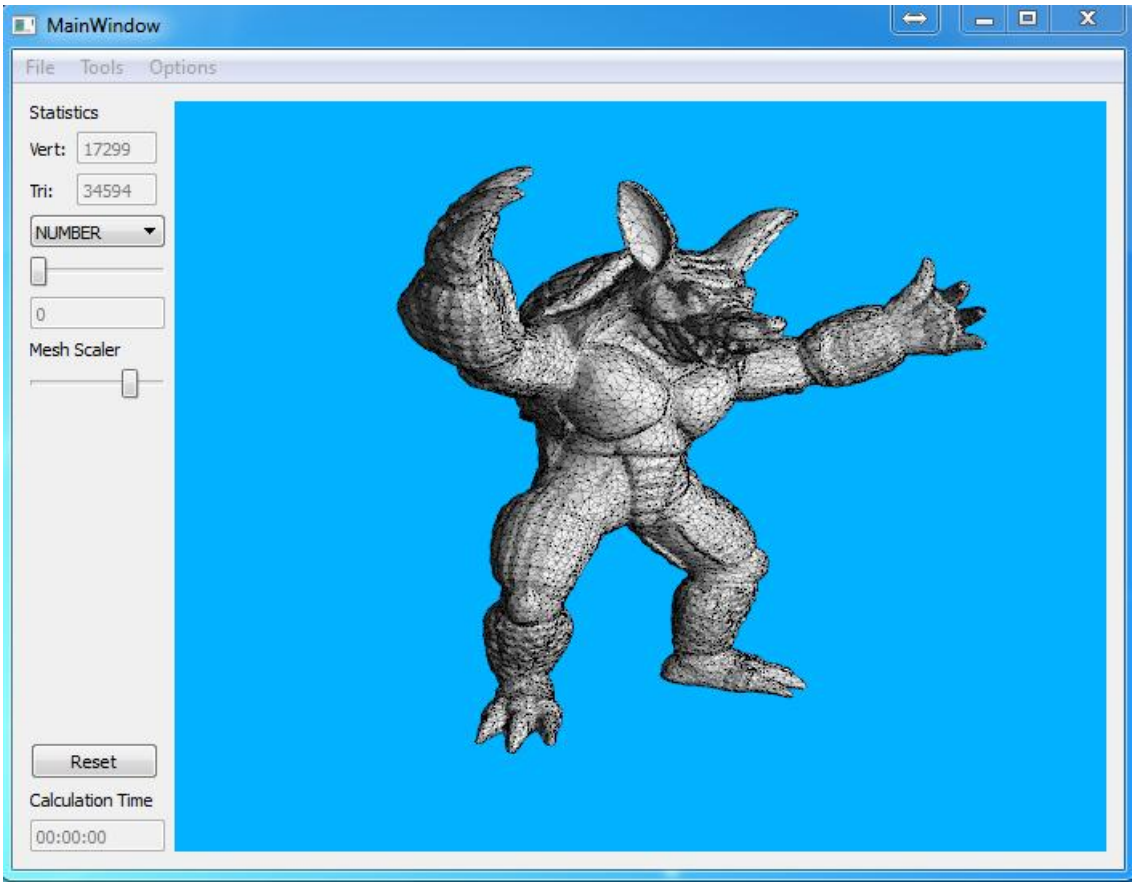
ქვემოთ ნაჩვენებია შესაბამისი ფაილის არჩევისა და იმპორტირების შედეგები.


MainWindow
Import Mesh
Simplify Mesh R5 ▸ Models
Search Models
Organize
New folder
Favorites
Desktop
Downloads
Dropbox
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
Local Disk (C:)
Local Disk (D:)
Name
Date modified
Type
Armadillo.obj 2/5/2012 4:23 PM Object File
Armadillo_3000.obj 2/5/2012 4:22 PM Object File
ArmadilloLP.obj 2/4/2012 1:01 AM Object File
high_cube.obj 5/1/2013 11:48 AM Object File
human_tr.obj 5/1/2013 10:58 PM Object File
HumanLowPoly.obj 5/2/2013 12:35 AM Object File
pa_train_w5_tr.obj 5/1/2013 10:58 PM Object File
pa_vehicle_cobra_tr.obj 5/1/2013 10:58 PM Object File
SunFlower_tr.obj 5/1/2013 10:59 PM Object File
T_hilux_TR.obj 5/1/2013 10:58 PM Object File
tri low poly.obj 5/2/2013 12:12 AM Object File
File name: Armadillo.obj
OBJ files (*.obj)
Open
Cancel


MainWindow
File
Tools
Options
Statistics
Vert: 17299
Tri: 34594
NUMBER
0
Mesh Scaler
Reset
Calculation Time
00:00:00

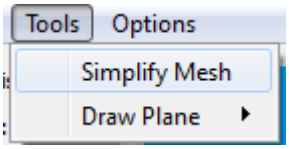
ასევე გვაქვს შესაბამისი თვისებები (Wireframe, Solide, Normals), რომლებიც გვეხმარება სხვადასხვა სახის ვიზუალურ წარმოდგენაში. მაგალითად, გამარტივების პროცესი რომ უკეთ შევამჩნიოთ, ვირჩევთ Wireframe-ს და ვრთავთ (Enable-ის მეშვეობით).



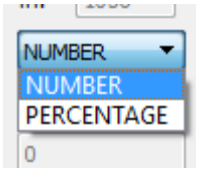


რის შემდეგაც საშუალება გვეძლევა კონკრეტულად დავინახოთ სამკუთხედები, რაც აიოლებს საწყის ეტაპსა და შესაბამისი სამკუთხედების წაშლის შემდეგ მიღებული შედეგს შორის სხვაობის ვიზუალურ აღქმას.

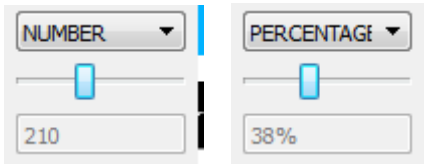
გამარტივების ფუნქცია შეგვიძლია გამოვიძახოთ Tools->Simplify Mesh,



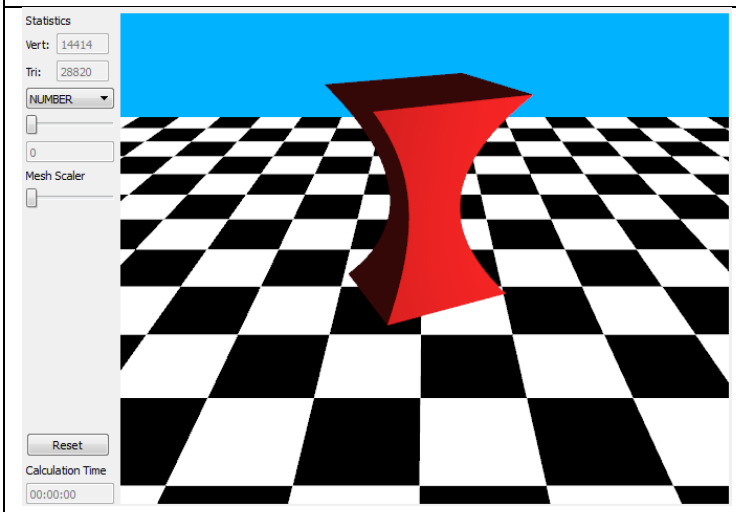
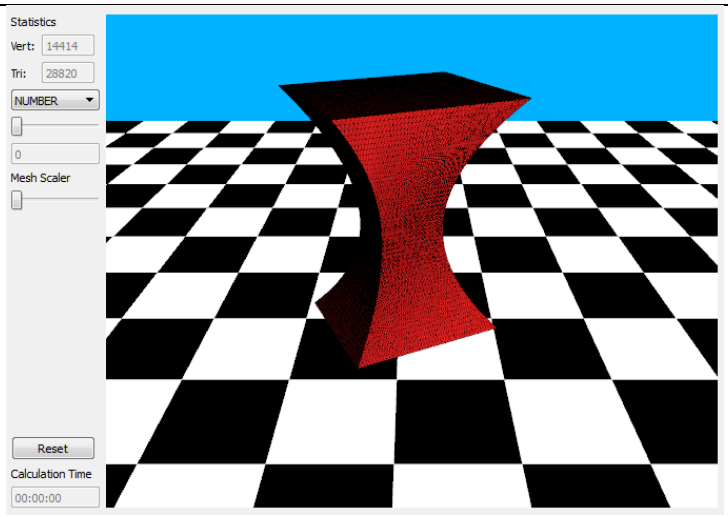
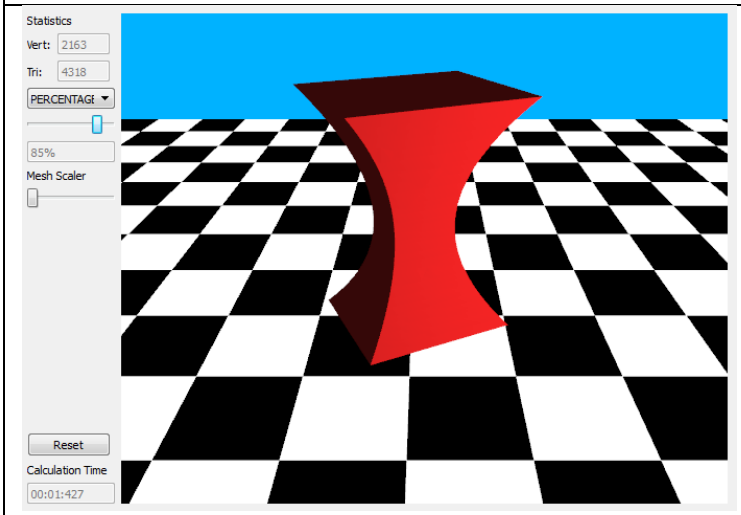
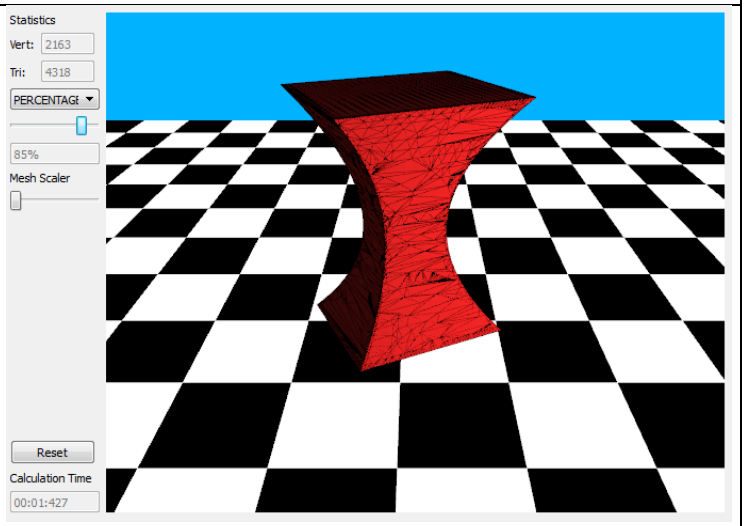
მაგრამ რაიმე კონკრეტული შედეგის მისაღებად უნდა ავირჩიოთ რა სახის გამარტივება გვინდა - რაოდენობრივი თუ პროცენტული.

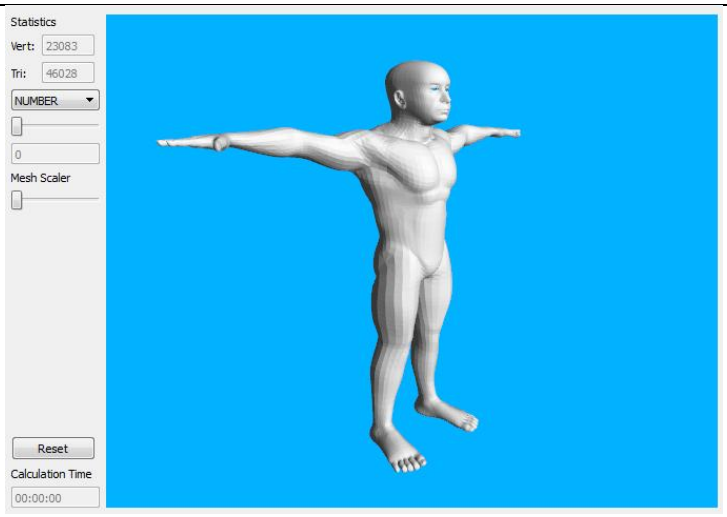
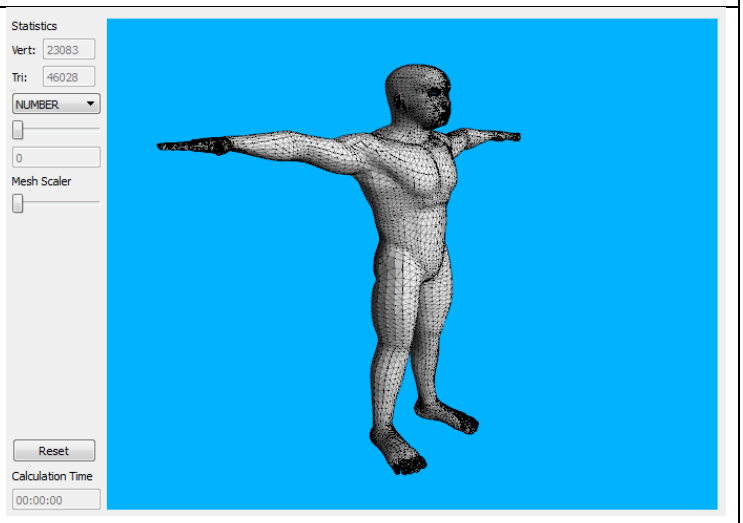
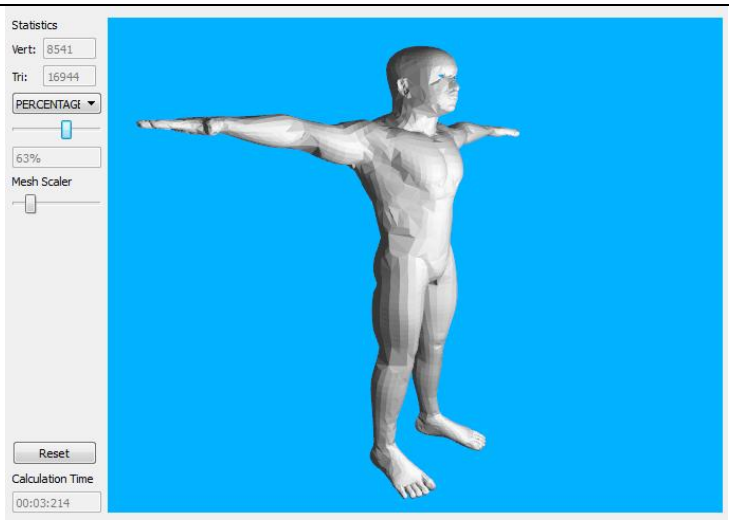
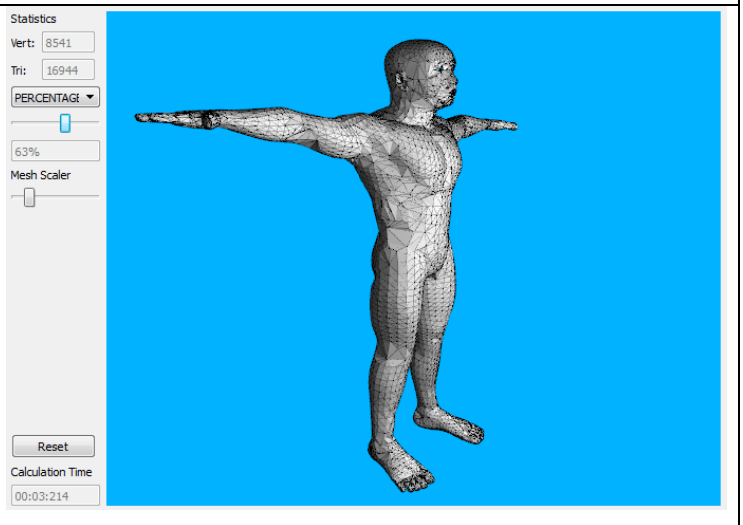


ამის შემდეგ ვირჩევთ რაოდენობას/პროცენტობას, თუ რამდენი სამკუთხედი წაიშალოს მოდელიდან.



ქვემოთ ნაჩვენებია პროგრამის მუშაობის რამდენიმე მაგალითი. ნათლად ჩანს, რომ საკმაოდ სწრაფად ვღებულობთ კარგ მიახლოებას (ასევე რეგულირდება „ხმაური“ ანუ დამახინჯება) დიდი რაოდენობით წვეროებისა და სამკუთხედების წაშლის შემდეგაც კი, რაც მიუთითებს ჩვენი ალგორითმის წარმატებულ მუშაობას.

Solide	Wireframe
	
საწყისი ეტაპი წვეროები: 14 414, სამკუთხედები: 28 820	
	
გამარტივების შემდეგ წვეროები: 2 163, სამკუთხედები: 4 318, გამარტივების დრო: 1.427 წამი	

Solide	Wireframe
	
საწყისი ეტაპი წვეროები: 23 083, სამკუთხედები: 46 028	
	
გამარტივების შემდეგ წვეროები: 8 541, სამკუთხედები: 16 944, გამარტივების დრო: 3.214 წამი	

5 დასკვნა და შედეგები

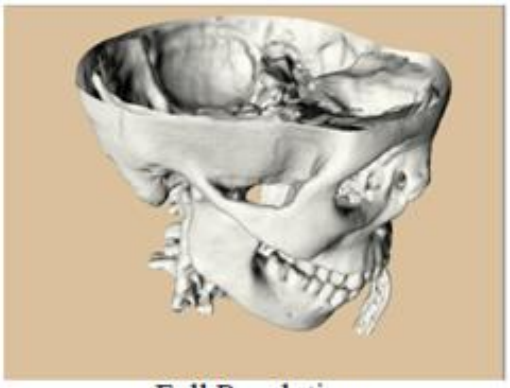
ამ თავში ჩვენ განვიხილავთ ორი განსხვავებული გეომეტრიული მოდელირების აპლიკაციის შედეგებს, რომლებიც ილუსტრირებას უკეთებენ დანაწევრების ალგორითმის სიძლიერეს. ასევე ვნახავთ, რამდენიმე განსხვავებულ კომპიუტერზე, ჩვენი პროგრამის მუშაობის შედეგებს. საბოლოო დასკვნის სახით კი ვაჩვენებთ ამ ნაშრომის მიზანს.

5.1 სხვა აპლიკაციების შედეგები

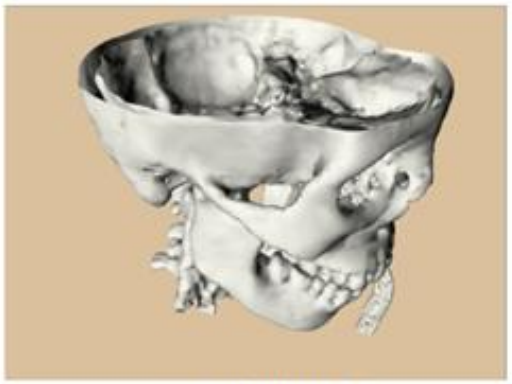
ორი განსხვავებული აპლიკაცია უკეთებს ილუსტრაციას სამკუთხედის დანაწევრების ალგორითმს. თუმცა, თითოეული აპლიკაცია იყენებს განსხვავებულ სქემას საწყისი ბადის შესაქმნელად, ყველა შედეგი მიღებულ იქნა ერთი და იგივე დანაწევრების ალგორითმით.

5.1.1 მოცულობითი მოდელირება

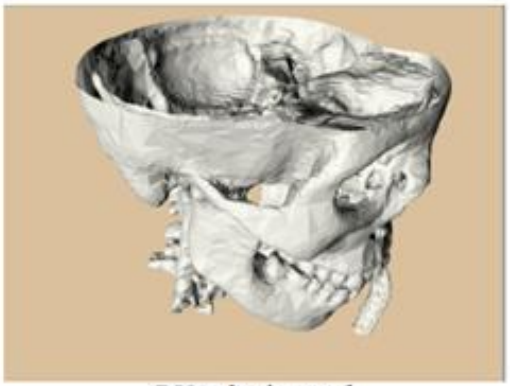
პირველი აპლიკაცია დანაწევრების ალგორითმს იყენებს, სამედიცინო და ინდუსტრიული სკანერების კომპიუტერული ტომოგრაფიიდან, დონის ზედაპირების შესაქმნელად. *მოძრავი კუბების* ალგორითმი გაუშვეს 256x256 პიქსელზე 93 ნაჭრის შესასწავლად. 560 000 სამკუთხედზე მეტი დასჭირდა ძვლის ზედაპირის მოდელირებას. უფრო ადრინდელმა კვლევებმა აჩვენა სამკუთხედის შემცირების სტრატეგია, რომელიც იყენებს გასაშუალებას სამკუთხედების რაოდენობის შესამცირებლად იგივე მონაცემთა სიმრავლეში. სამწუხაროდ, გასაშუალება აისახება მთლიან მონაცემთა სიმრავლეზე და ამით შთანთქავს მაღალი სიხშირის მახასიათებლებს. ფიგურების პირველი სიმრავლე გვიჩვენებს ძვლის დაგენერირებულ დონის ზედაპირს 0%, 75% და 90% დანაწევრებისათვის, სადაც დანაწევრების ხარისხი არის 1/5 ვოქსელის განზომილებასთან. შემდეგი ფორმების წყვილი გვიჩვენებს დანაწევრების შედეგს ინდუსტრიულ კომპიუტერული ტომოგრაფიის მონაცემთა სიმრავლისათვის, რომლებიც შედგებიან 300 ნაჭრისაგან, 512x512 გარჩევადობით.



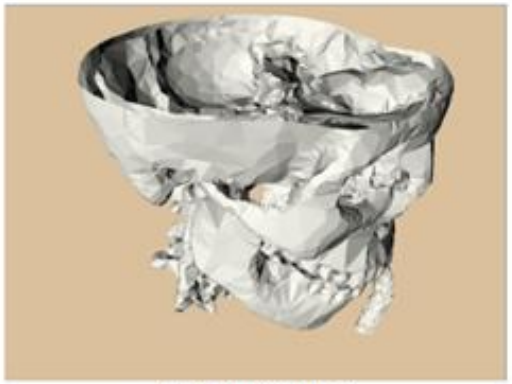
Full Resolution
(569K Gouraud shaded triangles)



75% decimated
(142K Gouraud shaded triangles)

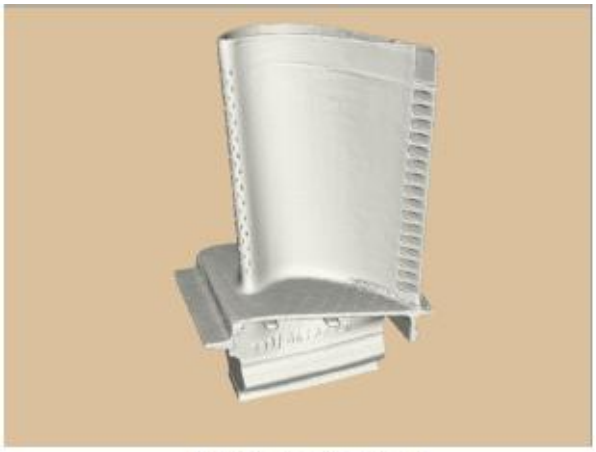


75% decimated
(142K flat shaded triangles)

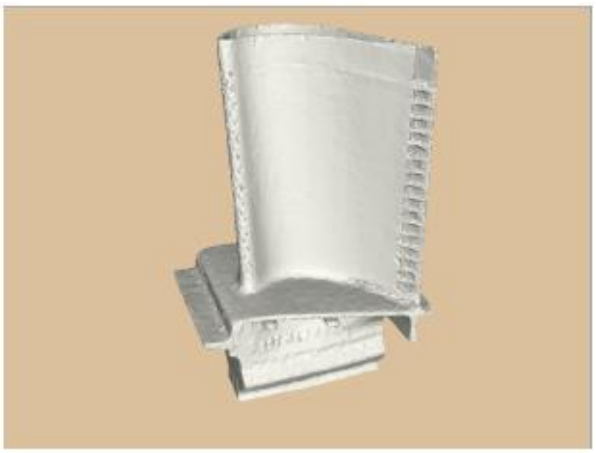


90% decimated
(57K flat shaded triangles)

შემდეგი დონის ზედაპირი შექმნილია 1.7 მილიონი სამკუთხედისგან შემდგარი ორიგინალი მონაცემთა წყაროდან. გამოსახულების სახით ჩვენ არ გამოგვაქვს ორიგინალი მოდელი, რადგან გადავაჭარბეთ ჩვენი გრაფიკული აპარატურის მონაცემთა გაცვლის მოცულობას. ვხედავთმ, რომ 90%-იანი სამკუთხედების დანაწევრების შემდეგაც კი, სერიული ნომერი ქვემოთ ნაჩვენებ მოდელზე ისევ ნათლად ჩანს.



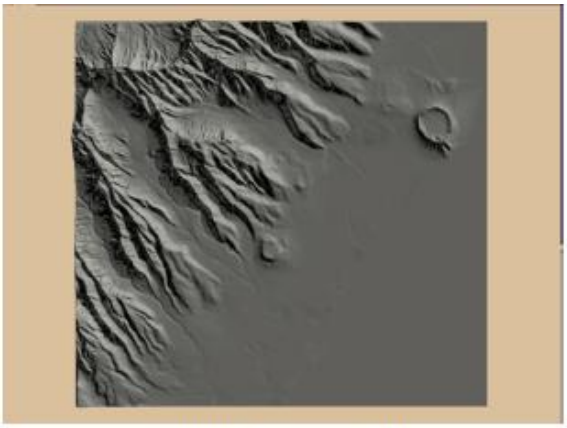
75% decimated
(425K flat shaded triangles)



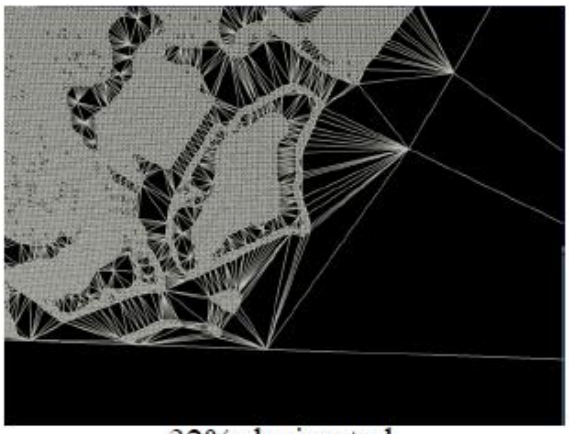
90% decimated
(170K flat shaded triangles)

5.1.2 რელიეფური მოდელირება

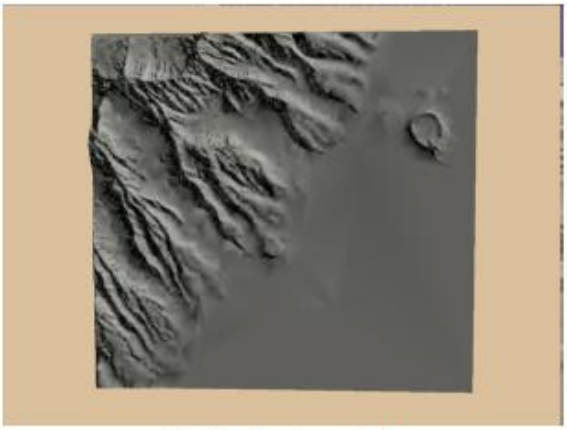
ამ შემთხვევაში დანაწევრების ალგორითმი გამოიყენება ორი რელიეფური მოდელის ციფრული მონაცემთა სიმრავლისათვის: ჰავაის შტატის დედაქალაქი ჰონოლულუ და მარინერის ხეობა მარსზე. ორივე მაგალითში საწყისი ბადეები მიღებულია ყოველი ერთგვაროვანი ოთხკუთხა ელემენტისათვის ორი სამკუთხედის აგების გზით. ჰონოლულუს მაგალითი ილუსტრირებას უკეთებს მრავალკუთხედების ეკონომიურ გამოყენებას ისეთ მოდელებში, სადაც დიდი ბრტყელი არეებია. თავდაპირველად დანაწევრების ბარიერად აღებულია 0 და ვაშორებთ თანასიბრტყეებზე მდებარე სამკუთხედების 30%. ბარიერის გაზრდამ მოაშორა სამკუთხედების 90%. შემდეგი ოთხი სურათი გვიჩვენებს 30% და 90%-იანი ტრიანგულაციების შედეგებს. ყურადღება მივაქციოთ, სანაპირო ზოლის ირგვლივ, დიდი ბრტყელი არეებიდან მცირე დეტალებზე გადასვლას.



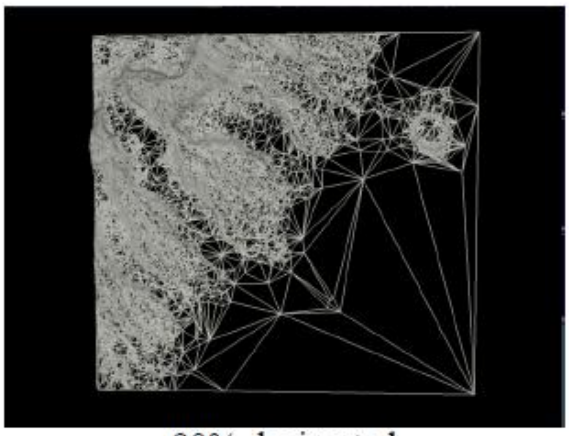
32% decimated
(276K flat shaded triangles)



32% decimated
(shore line detail, wireframe)

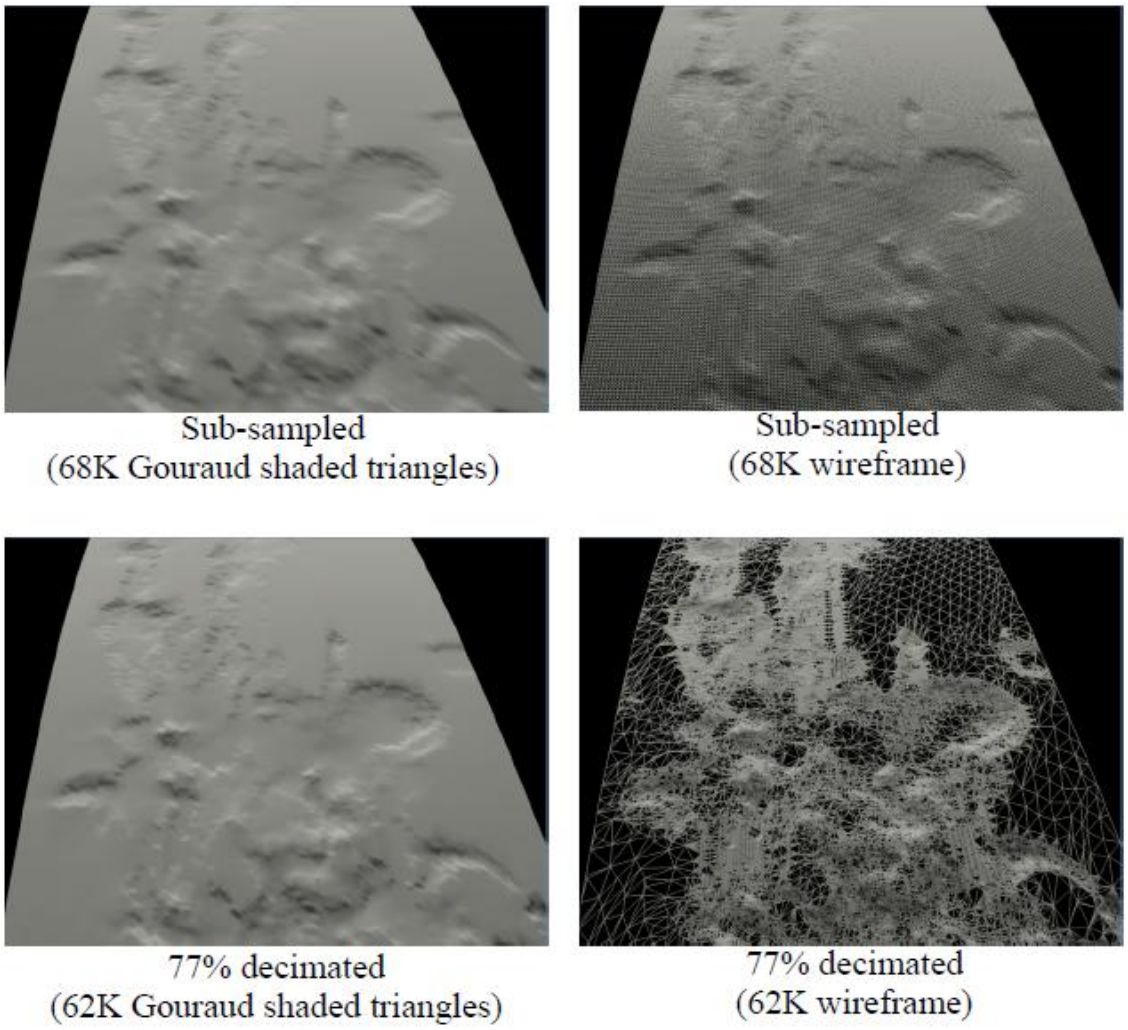


90% decimated
(40K Gouraud shaded triangles)



90% decimated
(40K wireframe)

მარსის მაგალითი არის სათანადო ტესტი, რადგან ჩვენ გვქონდა წვდომა გარჩევადობის დისკრეტიზებულ მონაცემებთან, რომელიც შეიძლება შევადაროთ დანაწევრებულ მოდელებს. მონაცემები წარმოადგენს მარინერის ხეობის(Mariner Valley) დასავლეთ ნაწილს, რომელიც არის 1000x500 კმ-ზე. ბოლო სურათი ადარებს shaded და wireframe მოდელებს, რომლების მიღებული იქნა დისკრეტიზაციისა და დანაწევრების საშუალებით. საწყისი მოდელი იყო 480x288. დისკრეტიზირებული მონაცემები იყო 240x144. 77%-იანი შემცირებით დისკრეტიზირებული მოდელი შეიცავს უფრო ნაკლებ სამკუთხედს, თუმცა გვიჩვენებს უკეთეს დეტალიზაციას.



5.2 ჩვენი პროგრამის შედეგები

შემდეგ ქვეთავებში განვიხილავთ ჩვენი პროგრამის, სხვადასხვა სისტემაზე, ტესტირების შედეგებს და ვნახავთ, რომ საკმაოდ სწრაფად და კარგად ახერხებს სხვადასხვა სიძლიერის სიტემებზე რთული გეომეტრიების გამარტივებას.

5.2.1 გამოყენებული კომპიუტერები

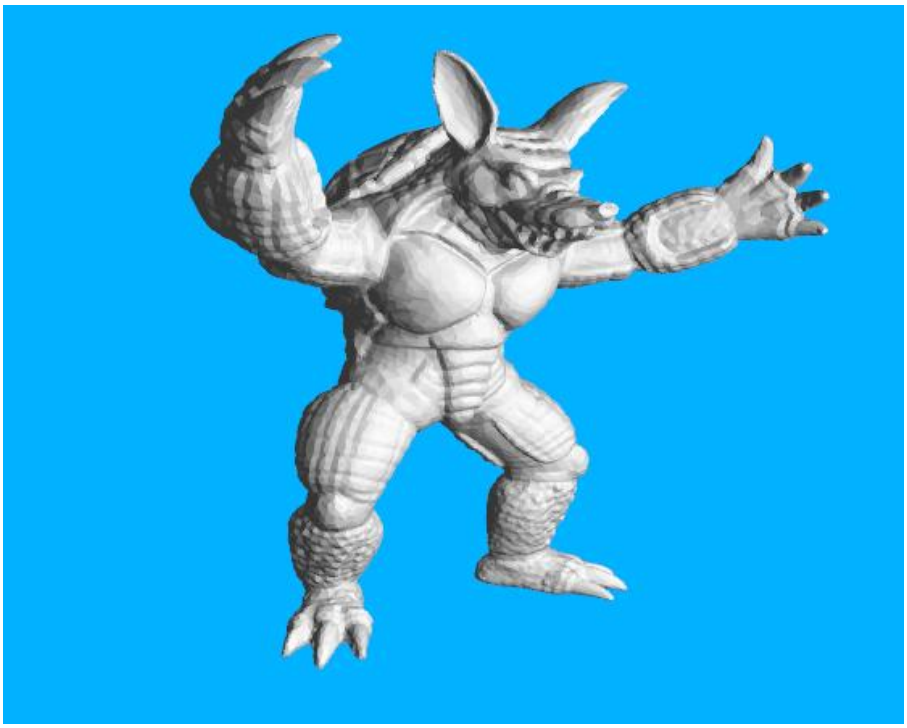
ტესტირებისათვის ჩვენ გამოვიყენეთ ოთხი კონპიუტერი. 1) პერსონალური კომპიუტერი(PC): Intel Core 2 Duo CPU E7400 2.80GHz, 4GB RAM 800 MHz, Windows 7 64-bit Operationg System. 2) პორტატული პერსონალური კომპიუტერი (DELL): Intel Core i3 CPU M350 2.27 GHz, 3GB RAM 1333 MHz, Windows 7 64-bit Operationg System. 3) პორტატული პერსონალური კომპიუტერი (Siemens Amilo): Intel Pentium Dual Core T 3200 2.0 GHz, 3GB RAM 800 MHz, Windows 7 32-bit Operation System. 4) პორტატული პერსონალური კომპიუტერი (Acer): Intel Core i7 3610QM 2.3 GHz(turbo boost up to 3.3 GHz), 6GB RAM 1333 MHz, Windows 8 64-bit Operation System.

5.2.2 გამარტივებული მოდელები

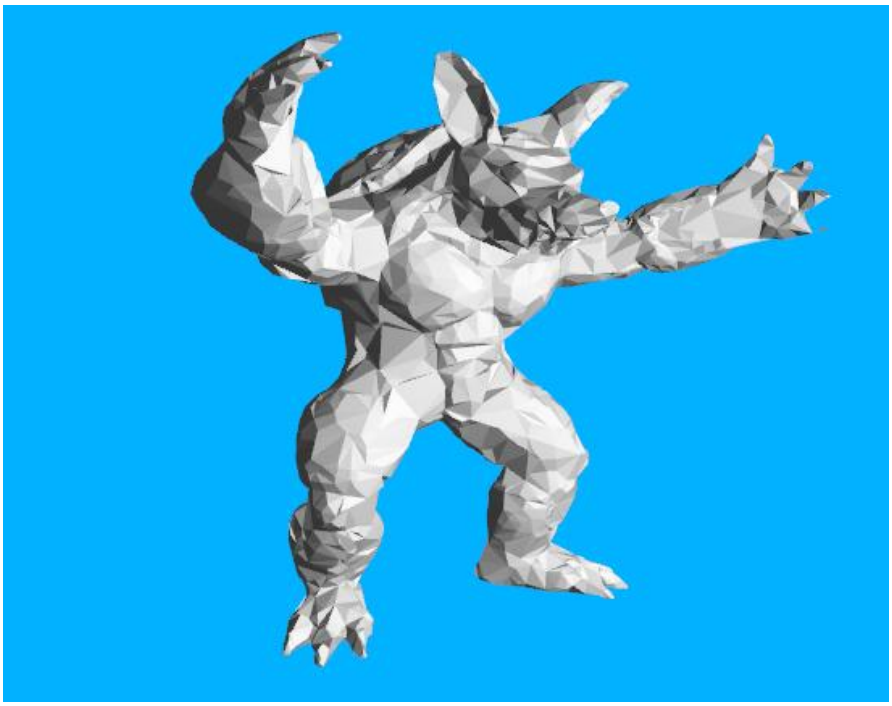
ჩვენ გამოვიყენეთ საკმაოდ ცნობილი მოდელები, მათ შორის სტენფორდის უნივერსიტეტის ცნობილი მოდელები http://graphics.stanford.edu/data/3Dscanrep/ (Armadillo და Bunny).

ქვემოთ ნაჩვენებია ყველა იმ მოდელის პირვანდელი და გამარტივების შემდგომი მდგომარეობები, რომლებზეც ჩავატარეთ ტესტირება, ზემოთ აღნიშნული კომპიუტერების მეშვეობით.

1) მოდელი - Armadillo.

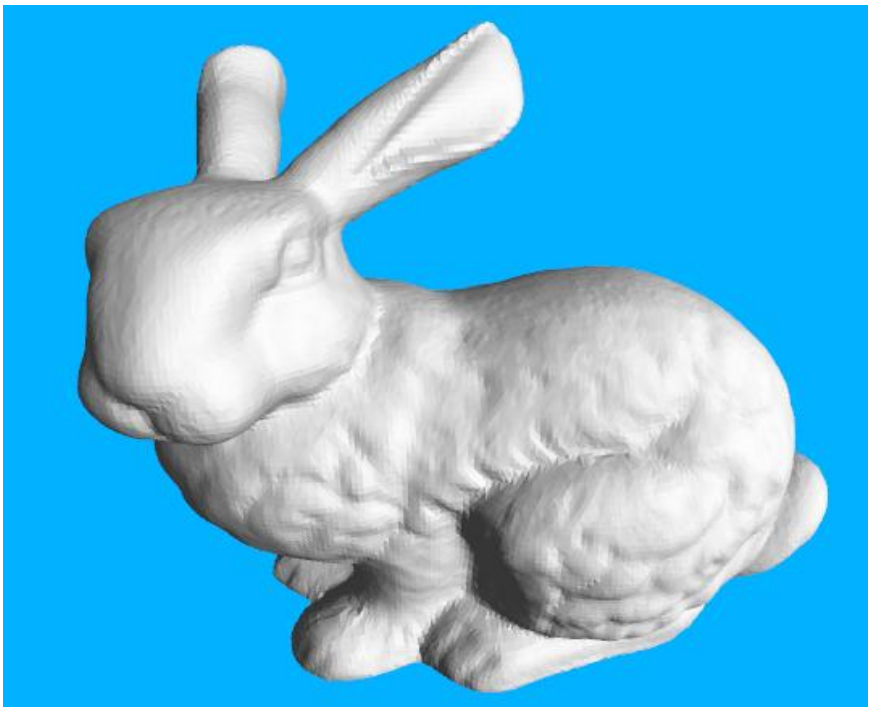


საწყის ეტაპზე 17 299 წვეროთი და 34 594 სამკუთხედით

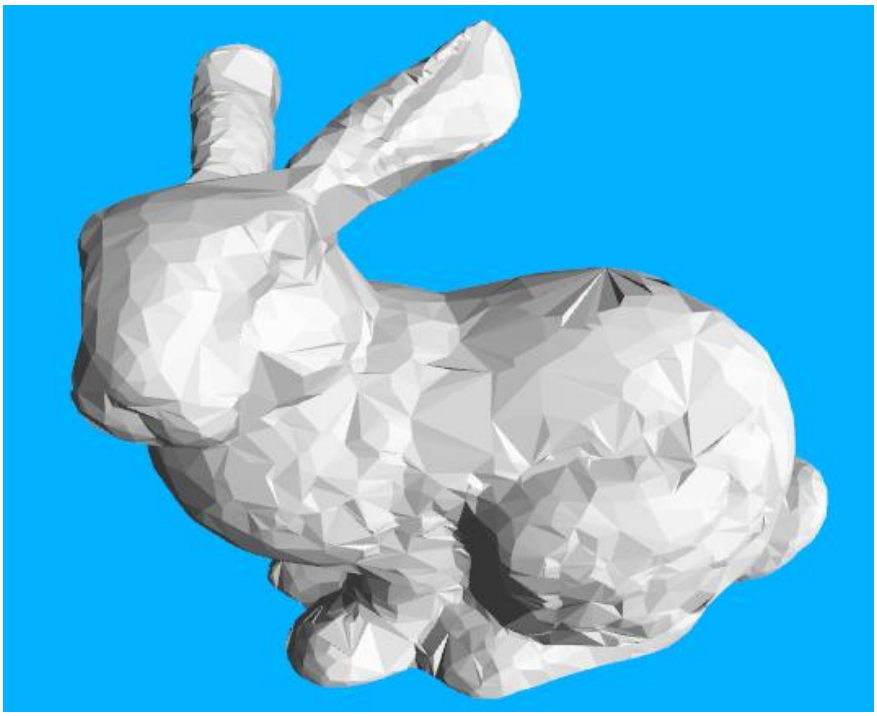


80%-იანი გამარტივების შემდგომ 3 460 წვეროთი და 6 916 სამკუთხედით

2) მოდელი - Bunny.

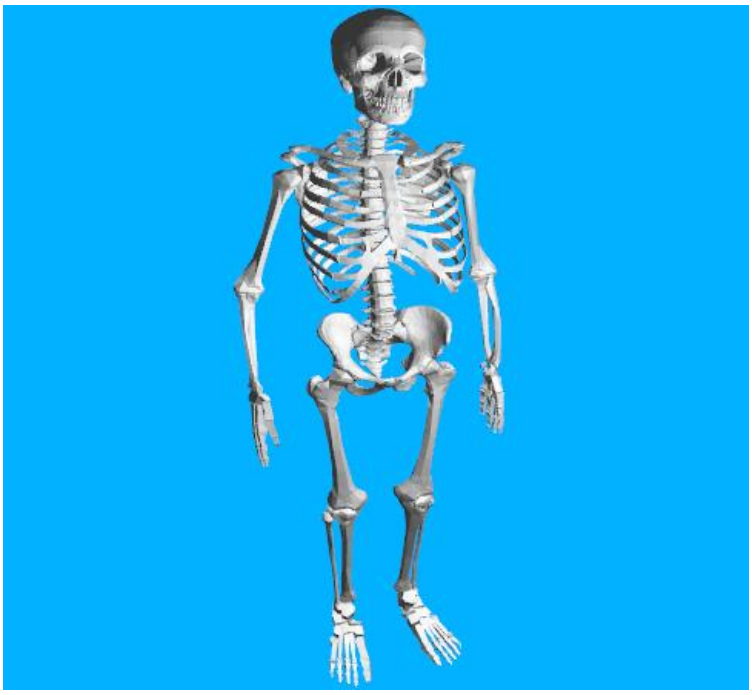


საწყის ეტაპზე 35 947 წვეროთი და 69 451 სამკუთხედით



90%-იანი გამარტივების შემდეგ 3 595 წვეროთი და 6 973 სამკუთხედით

3) მოდელი - Skeleton.



საწყის ეტაპზე 37 941 წვეროთი და 75 381 სამკუთხედით

70%-იანი გამარტივების შემდეგ 11 383 წვეროთი და 22 146 სამკუთხედით

4) მოდელი - SunFlower.

საწყის ეტაპზე 134 697 წვეროთი და 261 210 სამკუთხედით

70%-იანი გამარტივების შემდეგ 40 410 წვეროთი და 66 896 სამკუთხედით

5.2.3 გამოთვლის შედეგები

1) PC-ზე გამოთვლის შემდეგები:

	Armadillo	Bunny	Skeleton	SunFlower
გამარტივების მაჩვენებელი	80%	90%	70%	70%
საწყისი წვეროები	17 299	35 947	37 941	134 69
საწყისი სამკუთხედები	34 594	69 451	75 381	261 210
წაშლილი წვეროები	13 839	32 352	26 558	94 287
წაშლილი სამკუთხედები	27 678	62 478	53 235	194 314
დარჩენილი წვეროები	3 460	3 595	11 383	40 410
დარჩენილი სამკუთხედები	6 916	6 973	22 146	66 896
გამარტივების დრო	4.155 წმ	33.649 წმ	31.445 წმ	7 წთ 33.994 წმ

2) DELL-ზე გამოთვლის შედეგები:

	Armadillo	Bunny	Skeleton	SunFlower
გამარტივების მაჩვენებელი	80%	90%	70%	70%
საწყისი წვეროები	17 299	35 947	37 941	134 69
საწყისი სამკუთხედები	34 594	69 451	75 381	261 210
წაშლილი წვეროები	13 839	32 352	26 558	94 287
წაშლილი სამკუთხედები	27 678	62 478	53 235	194 314
დარჩენილი წვეროები	3 460	3 595	11 383	40 410
დარჩენილი სამკუთხედები	6 916	6 973	22 146	66 896
გამარტივების დრო	5.685 წმ	31.341 წმ	29.121 წმ	6 წთ 20.113 წმ

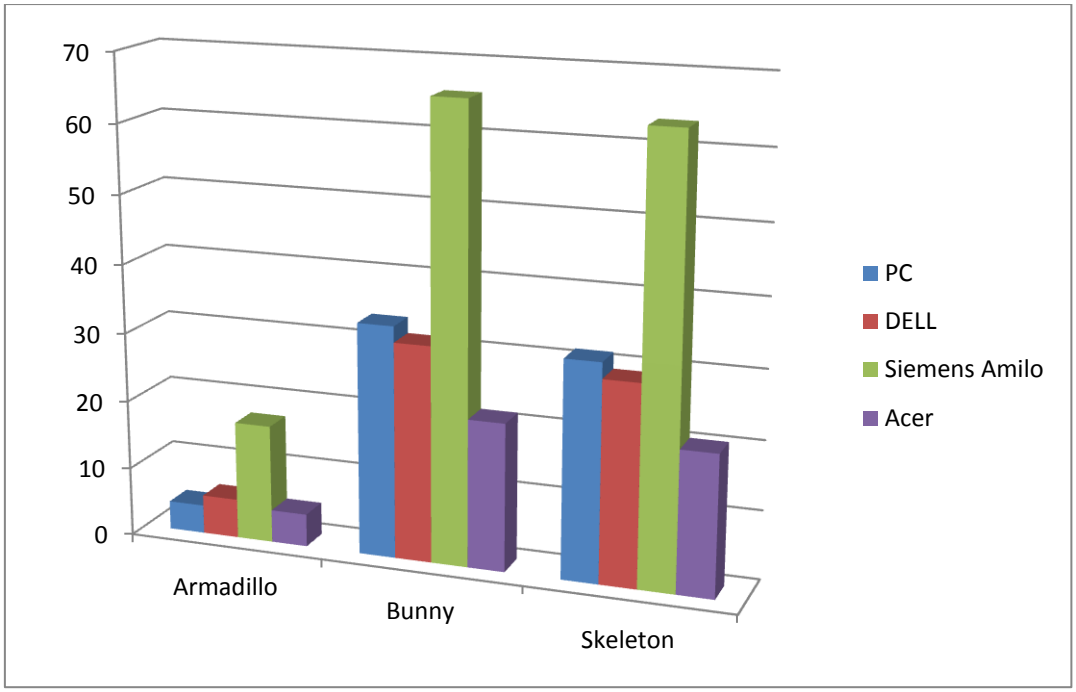
3) Siemens Amilo-ზე გამოთვლის შედეგები:

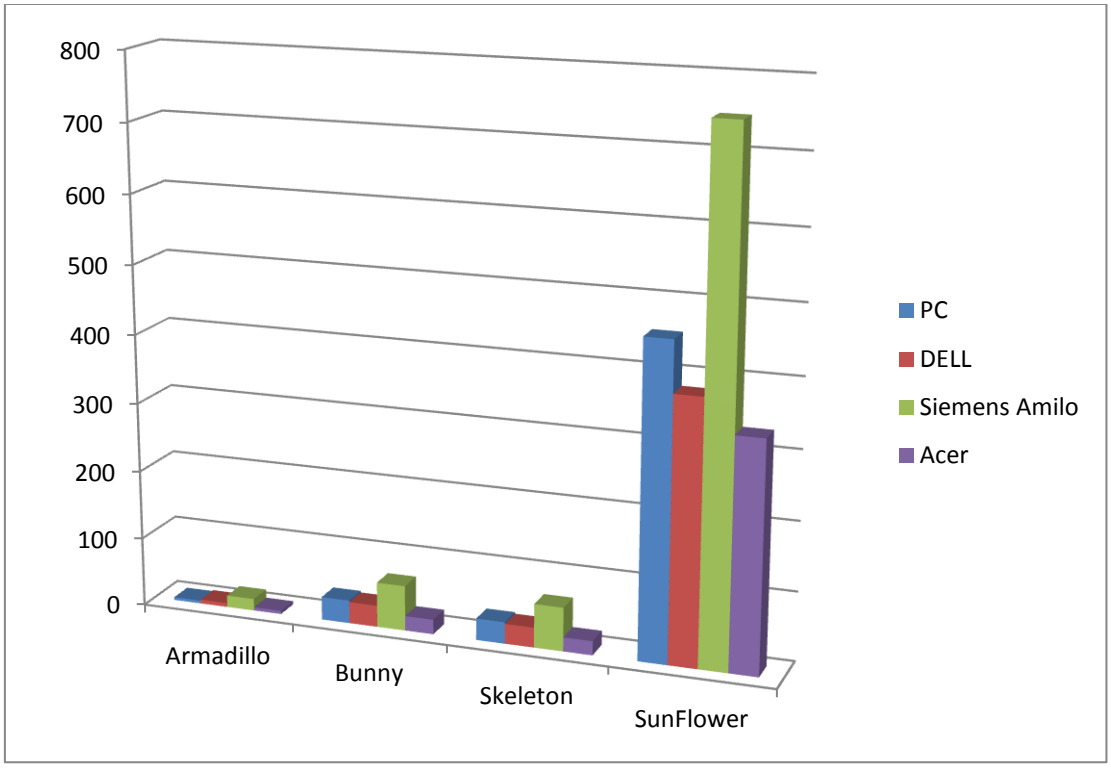
	Armadillo	Bunny	Skeleton	SunFlower
გამარტივების მაჩვენებელი	80%	90%	70%	70%
საწყისი წვეროები	17 299	35 947	37 941	134 69
საწყისი სამკუთხედები	34 594	69 451	75 381	261 210
წაშლილი წვეროები	13 839	32 352	26 558	94 287
წაშლილი სამკუთხედები	27 678	62 478	53 235	194 314
დარჩენილი წვეროები	3 460	3 595	11 383	40 410
დარჩენილი სამკუთხედები	6 916	6 973	22 146	66 896
გამარტივების დრო	17.233 წმ	1 წთ 5.632 წმ	1 წთ 3.499 წმ	12 წთ 27.193 წმ

4) Acer-ზე გამოთვლის შედეგები:

	Armadillo	Bunny	Skeleton	SunFlower
გამარტივების მაჩვენებელი	80%	90%	70%	70%
საწყისი წვეროები	17 299	35 947	37 941	134 69
საწყისი სამკუთხედები	34 594	69 451	75 381	261 210
წაშლილი წვეროები	13 839	32 352	26 558	94 287
წაშლილი სამკუთხედები	27 678	62 478	53 235	194 314
დარჩენილი წვეროები	3 460	3 595	11 383	40 410
დარჩენილი სამკუთხედები	6 916	6 973	22 146	66 896
გამარტივების დრო	4.745 წმ	21.463 წმ	20.524 წმ	4 წთ 32.115 წმ

შედეგის უკეთ სანახავად, ქვემოთ წარმოდგენილია ორი დიაგრამა. პირველზე გამოსახულია იმ სამი მოდელის შედეგი, რომელზეც ყველაზე ნაკლები დრო დაიხარჯა, ხოლო მეორეზე, კი ყველა ერთად. დიაგრამებზე ციფრებით მოცემულია დრო წამებში.


70
60
50
40
30
20
10
0
Armadillo
Bunny
Skeleton
PC
DELL
Siemens Amilo
Acer


800
700
600
500
400
300
200
100
0
Armadillo
Bunny
Skeleton
SunFlower
PC
DELL
Siemens Amilo
Acer

5.3 მიზანი

გრაფიკული აპლიკაციების უმეტესობა საჭიროებს, როგორც მაღალი გარჩევადობის ობიექტებს, ისე დაბალი გარჩევადობისაც. ამის გამო გრაფიკის დიზაინერებს უწევთ შექმნან ჯერ მაღალი ხარისხის მოდელი, ხოლო შემდეგ კი ის ხელით გაამარტივონ და შექმნან უფრო დაბალი ხარისხის, ერთი ან რამდენიმე, მსგავსი მოდელი.

ჩვენი მთავარი მიზანია, შევქმნათ ისეთი პროგრამული უზრუნველყოფა, რომელიც გრაფიკასთან მომუშავეთა უმეტესობას დაეხმარება და ზედმეტი დროისა და ენერგიის ხარჯვის გარეშე სულ რამდენიმე წამში/წუთში მიიღებს სასურველ შედეგს. გარდა განსაკუთრებული შემთხვევებისა, როდესაც აუცილებელია გრაფიკოსმა თვითონ გააკეთოს ყველა ხარისხის მოდელი, აღნიშნული პროგრამული უზრუნველყოფა ხელსაყრელია.

ლიტერატურა

1. OpenGL Architecture Review Board (ARB). Opengl. http://opengl.org.
2. Web3D Consortium. Virtual reality markup language. http://web3d.org.
3. Parsing a OBJ file for opengl APIs. http://opengl.org.
4. Jarek Rossignac, Geometric Simplification and Compression, GVU Center and College of Computing Georgia Institute of Technology (1997)
5. Luebke, D.P. A Developer's Survey of Polygonal Simplification Algorithms, IEEE Computer Graphics and Applications, Vol. 21 No. 3 (2001), 24-35.
6. Schroeder,W.J., Zarge, J.A., Lorensen,W.E. Decimation of Triangle Meshes, Computer Graphics, Volume 25, No. 3, (Proc. SIGGRAPH '92), July, 1992.
7. Zuckerberger, E.; Tal, A., Shlafman, S.: Polyhedral surface decomposition with applications, Computers Graphics, 26(5), 2002, 733-743.
8. Euler's Theorem, Tom Davis, July 14, 2009. http://www.geometer.org/mathcircles/euler.pdf.