JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 26 (2022) 83–87
Received: 5 October 2022
Accepted: 25 January 2023

The analysis of Blender open-source software cloth simulation capabilities

Analiza możliwości programu Blender pod kątem symulowania tkanin

Wojciech Kogut

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

This article explores Blender open-source software capabilities in the area of cloth simulations. Simulation performance with different scene complexity and quality settings is tested with a script that automates the testing. System resource utilization is measured and the appearance of visual artifacts is taken into consideration. Cloth simulation features of Blender are compared to commercial software.

Keywords: Computer graphics; Blender; cloth simulation; physics simulation

Streszczenie

Ten artykuł stanowi analizę otwarto-źródłowego programu Blender w dziedzinie symulacji tkanin. Wydajność przeprowadzania symulacji jest badana dla różnej złożoności scen oraz różnych ustawieniach jakości przy pomocy skryptu automatyzującego pomiary. Użycie zasobów systemu jest mierzone, brane pod uwagę jest również występowanie artefaktów wizualnych. Funkcjonalność Blendera pod kątem symulacji tkanin jest porównywana do oprogramowania komercyjnego.

Słowa kluczowe: Grafika komputerowa; Blender; symulacje tkanin; symulacje fizyki

Email address: **wojciech.kogut@pollub.edu.pl**

1. Introduction

Due to the movie industry shift from practical effects to computer generated imagery (CGI) and video games becoming more and more visually complex, as well as other use cases, e. g. de-sign mockups or architectural visualisation, there is a great focus in the field of 3D computer graphics in chasing what is known as „photo-realism". The goal is to make rendered images indistinguishable from real photographs, or, in the case of moving pictures, real videos. One area that is crucial in achieving that, especially when it comes to motion pictures is physics simulation. There are several types of physics simulation – there are fluid simulation, soft body physics, cloth simulation etc. In recent years, simulations that look sufficiently convincing have been achieved, but it comes at a cost – accurate physics simulations tend to be one of the most computationally demanding tasks in 3D graphics, even for modern hardware.

This article will focus on cloth simulations, more specifically, on how well, the most popular open-source 3D graphics software on the market handles them in terms of performance and how does it fare in comparison to other, commercial solutions. To achieve that, a series of benchmarks will be ran and the results of that tests will be presented throughout this article.

2. Research goals and methods

This article's goal is to examine the functionality of Blender open-source program in cloth simulation area as well as the simulation performance. The goal of the tests performed is to prove the following hypothesis: the more cores the CPU used for simulating cloth in Blender has, the less time those simulations take on average.

For that purpose, three benchmark scenes where created:

- A sheet of fabric hanging in the air from its corners with additional wind force object
- A sheet of fabric falling onto a sphere
- An animated character wearing a simple dress

Every scene has two versions differing in polygon count: less complex scene with simulated object having around 4000 vertexes and more complex one with simulated object having around 15000 vertexes. Each variant will be tested with two sets of simulation quality settings – 5 quality steps, 2 collision quality and 10 quality steps and 5 collision quality. The tests will be automated using a script in Python that runs the animation 100 times, clears memory cache during each run and gets the elapsed time at the end.

The results will be presented in a table, with percent usage of the computer resources during testing and 1-10 grade representing the occurrence of graphical artifacts such as object clipping.

Blender's functionality in cloth simulation will be examined in relation to other, paid programs – Autodesk Maya, Autodesk 3DSMax and Cinema4D. The analysis of functionality will be presented as a table where the rows will be representing cloth simulation features and column will be representing particular programs.

3. Automating tests

Blender does not provide users with ways to control the number of animation loops nor does it show the run time of the animation. The only information the user is

given is current framerate, which is not sufficient information.

In order to address those issues, there was a need for an algorithm that will automatically run the animation and measure the elapsed time.

Blender gives its users the option to run scripts in Python under a dedicated tab. Those scripts can control most aspects of a Blender project. Because of that, the algorithm written for the purpose of this paper was written in Python inside the tab that was mentioned earlier.

Listing 1: Python script for benchmark automation

```
import datetime
import bpy
from bpy.app.handlers import persistent

@persistent
def run_after_frame_change(dummy):
    if bpy.context.scene.frame_current > frame_end:
        bpy.app.handlers.frame_change_post.remove(run_after_frame_change)
        bpy.ops.screen.animation_cancel()
        loop()

def loop():
    global n
    global a, b
    if n==0:
        a = datetime.datetime.now()
    n += 1
    if n == number_of_time_you_want_loop:
        b = datetime.datetime.now()
        print(b - a)
        return

    bpy.context.object.modifiers["Cloth"].point_cache.use_library_path = True
    bpy.context.object.modifiers["Cloth"].point_cache.use_library_path = False
    bpy.ops.ptcache.free_bake_all()
    bpy.context.scene.frame_current = frame_start
    bpy.app.handlers.frame_change_post.append(run_after_frame_change)
    bpy.ops.screen.animation_play()

n = 0
a = 0
b = 0
number_of_time_you_want_loop = 100
frame_start, frame_end = 1, 60
bpy.ops.screen.animation_play()

bpy.app.handlers.frame_change_post.append(run_after_frame_change)
```

The algorithm loops the number of times specified by number_of_time_you_want_loop variable. When the loop ends, it stops the animation. At the start of the execution of the script, variable a is initialized with current date and time and where the script reaches the end of the loop, current date and time is assigned to variable b. To get the elapsed time, variable a is subtracted from b at the after looping the number of times specified earlier. Aside from that, during each loop, the data cached in memory is cleared and using library path is disabled. It has to be re-enabled and disabled again each time because of the possible overlook on developers' part.

4. Performance benchmarks

The tests were performed on two different systems. The first benchmarking platform had the following specification: Intel Core i5-11400F with 6 cores and 12 threads, Nvidia GeForce RTX 3070 and 32 GB of DDR4 RAM. The second platform had the following specification: Intel Xeon E5-2630v3 (8 cores, 16 threads), 64 GB of DDR3 RAM and an integrated Intel GPU. The tests were performed in Blender version 3.1, which was the latest stable release at the time.

All scenes were tested with 2 sets of settings: quality steps set to 5 with collision quality set to 2 and also quality steps set to 10 with collision quality set to 5. The first two scene had collision detection distance set to 0.015 m, whist the third one had it set to 0.001 m. The rest of the settings were left on the default values provided by the software.

4.1. Execution times

Figure 1 presents the average times of simulation for 5 quality steps and 2 collision quality settings on the first benchmarking platform. The graph shows that increasing the polygon count results in significant increase in execution times. The simulation took from 3 to 9 times longer when going from lower to higher mesh density with the same quality settings across all scenes. However, despite not having any collision object, the scene with a sheet of fabric takes more time to calculate than the scene with a sphere when comparing higher polygon count versions of those benchmarks.

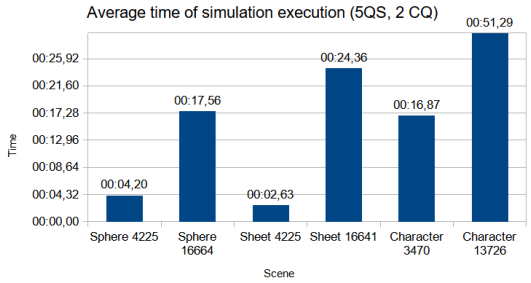

Figure 1: Average time of execution for 5 quality steps and 2 collision quality on the first machine.

Figure 2 showcases the result obtained with the second benchmarking platform with the lower simulation quality settings. The simulation times are slightly higher than on the first machine with the exception for the more complex version of the sheet of fabric benchmark, which is slightly faster on the second machine, although the results follow the similar pattern as with the first platform.

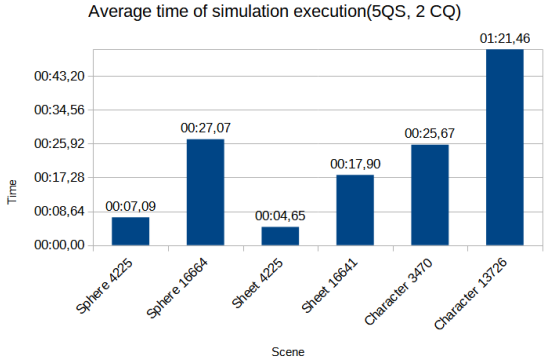

Figure 2: Average time of execution for 5 quality steps and 2 collision quality on the second machine.

Figure 3 shows the average times for quality steps set to 10 and collision quality set to 5 on the first machine. As expected, increasing those values significantly affects execution times – in case of this research there was a 1,5 to 5 times increase compared to the tests shown in figure 1.

For example, the animation for the character scene with 13736 polygon cloth object is 3,04 times slower than 3470 version with 5QS/2CQ settings, but 6,22 times slower for 10QS/5CQ. The distribution of values is comparable to graph seen on figure 1, but, in case of the most demanding of the scenes, the one with a character, the simulation time is far longer and the difference between less complex version of it and more complex one is also far greater.

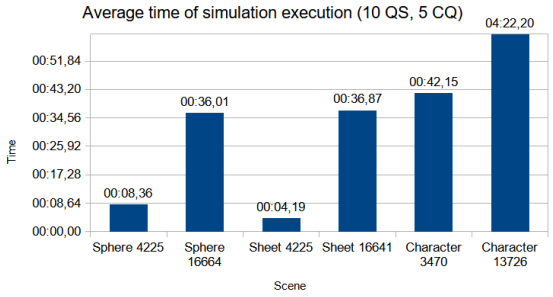

Figure 3: Average times of execution for 10 quality steps and 5 collision quality on the first machine.

Figure 4 shows the execution times for the scenes with higher simulation quality settings obtained on the second platform. Similarly to the corresponding graph for the first test bench, the results are mostly slightly higher than those acquired on the first computer, although the last scene, the character benchmark with higher polygon count ran faster by almost exactly 1 minute on the second platform.

This may be because of utilizing 2 more cores and, as a result, 4 more threads of the CPU that the second platform had available to it, albeit at lower clock speed.

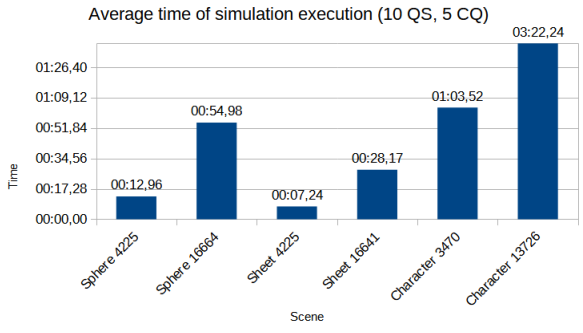

Figure 4: Average times of execution for 10 quality steps and 5 collision quality on the second machine.

In all of the tests performed, the character benchmark took the most time to complete due to having the most complex collision object, that was also animated. This causes the simulation engine to take the movement of that object into consideration and increases the time of the calculations.

4.2. Resource usage and artifacts

Table 1 presents the use of computer resources during the testing and perceived level of graphical arti-facts such as inter-object clipping.

As expected, more complexity equals more strain on the computer – e.g., sheet scene that has no interactions between objects uses less processing power even in the heaviest version than other scenes that have collision objects thrown into the mix. However, it can be noticed that in some cases the more complex scene may use less CPU power than the less complex one. This is likely due to the engine utilizing more cores and spreading the load more evenly for more demanding scenes. For example – the sheet scene with 4225 polygons with 10 QS/5CQ used 1 thread at 100% with others being idle, whilst the sheet scene with the same QS and CQ, but 16641 polygons used one thread at 80% and another at 78%. Sheet and sphere scenes used only one or two threads entirely or close to entirely, leaving the rest at idle and the character scene utilized all CPU threads above 50% with at least one exceeding 80%, however 13726 version of the character benchmark actually used 8 threads at close to 100%.

The simulation impact on RAM usage was negligible, since the simulation caching was disabled for the purpose of this research. Any variation in this category was most likely due to other programs running in the background and other components of Blender.

It can be observed that the polygon count of the simulated object impacts the cache size. The quality settings, however, do not impact the cache size, since there was no difference in the cached memory between higher settings and lower settings versions of scenes that had the same number of vertexes.

Measuring the level of graphical artifacts observed is a challenging task, because of it being subjective and depending on one's own perception which varies from person to person, however it is important when considering the outcome of the simulation. For that reason, the group of 20 anonymous respondents aged 17-33, mostly male, have graded all the scenes via a form. The respondents assigned the score from 1 to 10 to each scene – 1 meaning there are none or barely any visible artifacts and 10 meaning that there are a lot of noticeable artifacts. The value in Table 1 is the average grade given to a scene by the respondents.

Table 1: Computer resources used and graphical artifacts observed by the respondents

Scene	Vertex count	Quality Steps	Collision Quality	CPU	RAM	Cache (MiB)	Artifacts
Sphere	4225	5	2	33,00%	12,00%	8,9	4,6
Sphere	4225	10	5	41,00%	12,00%	8,9	4
Sphere	16664	5	2	30,00%	13,00%	34,9	5
Sphere	16664	10	5	50,00%	12,00%	34,9	4,85
Sheet	4225	5	2	27,00%	14,00%	8,9	3,65
Sheet	4225	10	5	25,00%	14,00%	8,9	3,45
Sheet	16641	5	2	23,00%	15,00%	34,9	3,45
Sheet	16641	10	5	21,00%	15,00%	34,9	3,25
Character	3470	5	2	60,00%	17,00%	7,3	5,75
Character	3470	10	5	75,00%	17,00%	7,3	5,75
Character	13726	5	2	55,00%	17,00%	28,9	5,55
Character	13726	10	5	67,00%	14,00%	28,9	4,95

5. Feature analysis

Blender as an open-source software may be falsely perceived by professionals that use commercial solutions as a basic tool that lack functionality and flexibility those programs have, when in reality more and more features are coming to Blender in each new release and it is now one of the most complete 3D tool available.

In this chapter, the features of Blender will be compared to other, paid programs, namely Autodesk 3DS Max, Autodesk Maya and Cinema4D. The results will be shown in a table.

Cloth simulation options in Blender are accessible under the „Cloth" section of the „Physics" tab, while in case of 3DSMax it is available from within the modifiers section as a cloth modifier [1]. It should be noted that you can also find cloth simulation in the modifiers stack in blender as well [2]. Additionally, 3DSMax only allow interactions between objects that are under one instance of cloth modifier. In case of Maya, all of cloth simulation features belong to nCloth plugin [3] and in Cinema4D they can be found under simulation tags section [4].

One of the most important factors that determines both quality and performance of the simulation is choosing a numerical integration method. There are four main categories of integration methods: explicit, implicit, high-order and low-order methods. The authors of the paper [5] concluded that implicit methods like Backward Euler method result in faster calculation, while explicit ones like Runge-Kutta produce the most accurate results. However, none of the software tested offer an option to choose the integration method.

Another important thing to consider is the number of steps of the simulation. As pointed out in paper [6], traditionally small time steps had to be used to avoid numerical instability. The authors of this publication proposed an algorithm that can use larger steps while mitigating instability, but generally the more time steps, the more accurate the result will end up being, at the cost of a slower calculation. All of the tested software enable the user to set the number of steps (in case of Maya the options are called „iterations" and „subsamples", but ultimately it is the same as choosing time steps).

As the paper [7] concludes, running the simulation on a GPU can be up to 60 times faster than on the CPU, but despite the benefits, none of the tested programs allow choosing to simulate on a graphics card outside of rendering and use a CPU computing instead.

All of the selected programs offer an option to control internal characteristics of a cloth object like stiffness, bending, damping, compression and pressure, while Blender and both of Autodesk products also have pre-defined presets for common material types.

Additionally, Blender and Maya offer an option to choose a bending model – angular and linear in the case of Blender.

All of the software tested allow to enable or disable both inter-object and internal collisions and all except 3DS Max also have an option to control the parameters of those collisions, like the distance at which the software detects collisions, friction between colliding geometry etc.

All of the programs enable the user to constrain certain parts of the mesh to either stay in their place or stick to another part of the mesh or a different mesh, while only Blender has an option that enables the simulated object to dynamically react to deformations of the meshes it collides with, here called dynamic mesh.

One way of creating clothing for virtual characters is a method called sewing – which like the real-world sewing involves designing the clothing item in parts that are later joined together. This technique in 3D graphics comes from a well-known tool for creating garment for 3D characters, Marvelous Designer and it is present in all of selected programs except for Maya.

Tearing cloth, although possible in Blender with a use of pin groups is not explicitly available from the simulation options as opposed to 3DSMax, where you can also select individual seams that will be teared and Cinema4D.

All of the programs have a functionality to cache the simulation results into memory and onto a storage disk (which, for distinction will be referred to as baking). It should be pointed out that Maya, for example, require the user to manually save cache each time there is anything changed, while Blender recalculates automatically after every change to the object or its properties.

Table 2 presents the comparison of the cloth-simulation functionality between all of the chosen programs in detail.

Table 2: Cloth simulation functionalities and adjustable properties in Blender, 3DSMax, Maya and Cinema4D

Feature	Blender	3DSMax	Maya	Cinema4D
Time steps	Y	Y	Y	Y
Subsampling	N	Y	Y	Y
Speed control	Y	N	Y	N
Mass	Y	Y	Y	Y
Bending Model	Y	N	Y	N
Integration alogorythm	N	N	N	N
Bending parameters	Y	Y	Y	Y
Damping parameters	Y	Y	Y	Y
Internal springs	Y	Y	Y	Y
Friction	Y	Y	Y	Y
Pressure	Y	Y	Y	Y
Compression	Y	Y	Y	N
Stiffness	Y	Y	Y	Y
Tearing	N	Y	N	Y
Tear strength	N	Y	N	Y
Keep shape	N	Y	Y	N
Seam parameters	N	Y	N	Y
Order of interactions	N	Y	N	N
Cloth type presets	Y	Y	Y	N
Simulation cache'ing	Y	Y	Y	Y
Simulation baking	Y	Y	Y	Y
Cloth constraints	Y	Y	Y	Y
Sewing	Y	Y	N	Y
Advanced pinching	N	Y	N	N
Weld group	N	Y	N	N
Dynamic mesh	Y	N	N	N
Inter-object collisions	Y	Y	Y	Y
Self-collisions	Y	Y	Y	Y
Collision quality	Y	N	Y	Y
Colission distance	Y	N	Y	Y
Collision friction	Y	N	Y	N
Collision clamping	Y	N	N	Y
Edge collisions	N	N	N	Y
GPU Compute	N	N	N	N
Weights	Y	N	Y	Y

This comparison shows that the cloth simulation functionalities provided by Blender out of the box are on par with those offered by commercial soft-ware and considering the open-source nature of this program it will likely offer even more features in years to come.

6. Conclusion

In this article, the performance of cloth simulations in different scenarios was measured. The research confirmed that more vertexes and higher quality settings significantly impact the execution times – the tests shown up to 6 times increase in execution time when going from 5 quality steps and 2 collision quality to 10 quality steps with the same polygon count and 5 collision quality and up to 9 times increase when going from 4000 polygons to 16000 polygons with the same quality settings. At the same time increasing those mesh density and quality setting, doesn't remove all visual artifacts during the animation.

The initial hypothesis could not be proven as the results turned out ambiguous. Most benchmarks were faster on the system with less CPU cores that had higher clock-speed and only the most demanding of the benchmarks, the character scene with 14000 vertices cloth object was faster on the system with more CPU cores with less clock speed.

Although a fast modern CPU was used during the testing, it is still impossible to run complex cloth simulation in real time on a personal computer with a use of a CPU. The next step in this research could be testing how simulating using the GPU compares to our results, but as of now, Blender only allows changing the graphics processor for rendering.

The second part of research shown that Blender's functionality in the field of cloth simulations is comparable to commercial solutions. Most of the programs tested provided similar functionalities with mostly minor differences.

References

[1] Autodesk, 3DS Max Documentation, https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/3DSMax/files/GUID-80CAE782-8EFD-4E9E-ABA2-C53127D6659A-htm.html, [30.08.2022]

[2] Blender Foundation, Blender Documentation, https://docs.blender.org/manual/en/latest/physics/cloth/introduction.html, [30.08.2022]

[3] Autodesk, Maya Documentation, https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2020/ENU/Maya-CharEffEnvBuild/files/GUID-ED791F1C-8412-4785-829F-9925F2604E8A-htm.html, [30.08.2022]

[4] Maxon, Cinema4D Documentation, https://help.maxon.net/c4d/en-us/#html/5953.html?Highlight=cloth, [30.08.2022]

[5] P. Volino, N. Magnenat-Thalmann, Comparing efficiency of integration methods for cloth simulation, Proceedings of Computer Graphics International (2001) 265-272.

[6] D. Baraff, A. Witkin, Large steps in cloth simulation, SIGGRAPH '98 (1998) 43-54.

[7] T. Min, T. Ruofeng, R. Narain, M. Chang, D. Manocha, A GPU-based Streaming Algorithm for High-Resolution Cloth Simulation, Computer Graphics Forum 32(7) (2013) 21-30.