Sniego deformacija - Ataskaita

Domas Kalinauskas

Aprašas

Dinaminė sniego, bei kitokio paviršiaus deformacija padeda suteikti didesni realistiškumo jausma žaidimuose. Man buvo įdomu išsiaiškinti kaip įmanoma tokį efektą pasiekti.

Figure 1: padangos rastas

Trečios šalies bibliotekos

Įgyvendinimui pagrindinės trečios šalies bibliotekos:

- Three.js - kompiuterinės grafikos biblioteka
- Cannon-es - fizikos biblioteka

Dar kartu buvo naudojamos kelios papildomos dėl paprastumo/debugginimo:

- cannon-es-debugger - rodo cannon-es kūnų wireframe'us
- three-mesh-bvh - paspartina three.js raycast operacijas
- three-to-cannon - sugeneruoja kūnam bounding-box'us pagal three.js mesh'a

Įgyvendinimas

Įgyvendinimas buvo atliekamas keliais etapais, iki kol pasiekiau galutini, optimaliausia varianta.

Tarp variantu pats deformacijos piešimo metodas nesikeitė, tačiau keitėsi kaip deformacija buvo aptinkama.

Deformacijos piešimas

Paprasčiausia dalis viso projekto buvo pačios deformacijos atvaizdavimas. Piešimui pasirašiau ganėtinai paprasta GLSL vertex ir fragment shaderį, kuris buvo paduodamas plokštumos piešimui.

Į shaderius buvo paduodamas deformacijos tekstura, kurioje balta spalva reprezentavo jokios deformacijos, o juoda pilna deformacija. Naudojant šia tekstūra, vertex shaderyje buvo pakeičiamos galutinės vertice pozicijos. Šalia to, teko uždėti limitacija kad aplink plokštumos kampus deformacija visada būtu pilna, kadangi kitu atvėju matytusi tarpas tarp nulinės ir pilnos deformacijos.

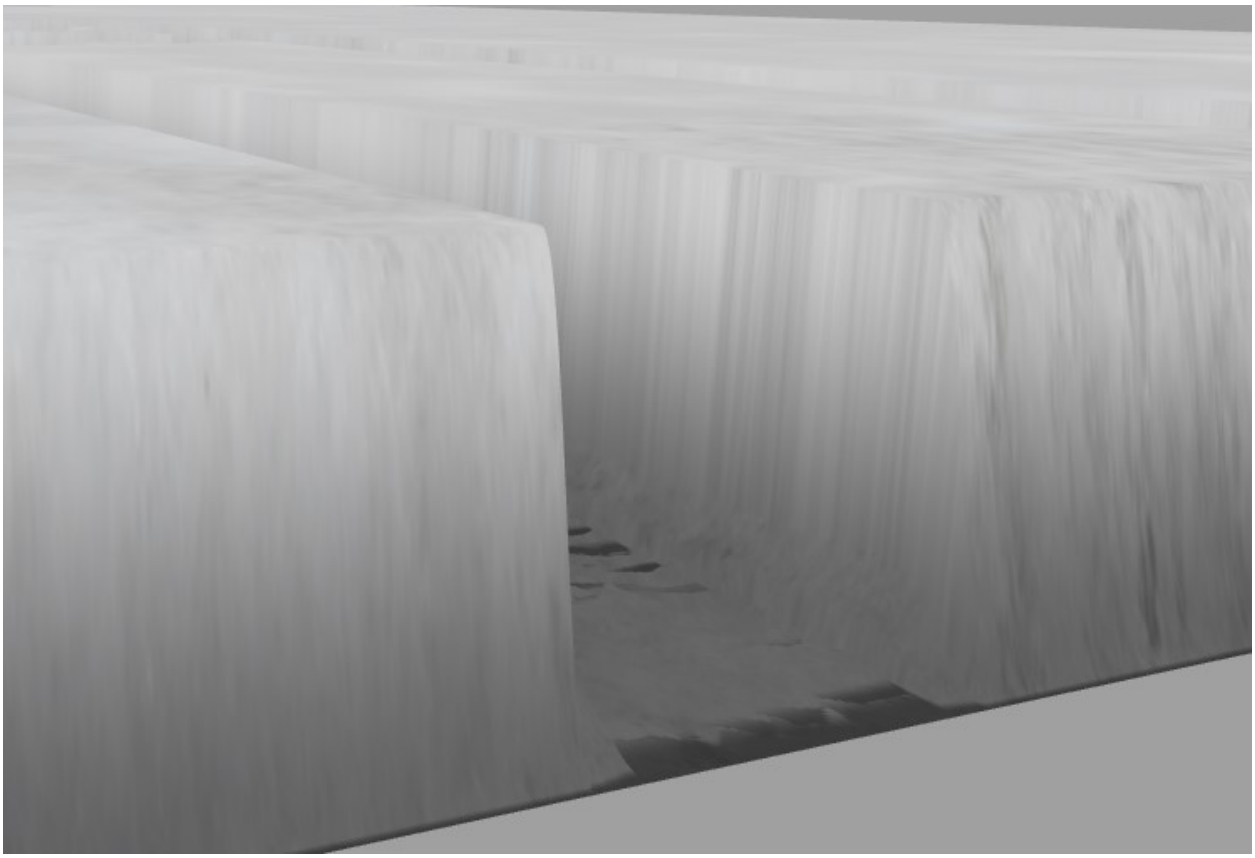

Figure 2: sniegas

Fragment shaderyje, atsižvelgiant į esančia deformacija, priskiriame skirtinga sniegui spalva. Ta spalva uždedama ant naudojamos sniego tekstūros.

Pradinis variantas - fizikos objektais paremta deformacija

Pirmas ir papraščiausias metodas kurį aš bandžiau, buvo turėti, šalia mašinos, dar ir 'heightfield' - iš esmės dinaminio aukščio fizikinis objektas, kuris mašinos neįtakoja, bet tik generuoja susidurimus.

Susidurimo metu, butu išsiunciamas spindulys (raycast) nuo susidurimo taško į plokštumos apačia. Tada spindulio ilgis paemamas kad nuspresti kiek turėtu sniegas būt 'įspaustas'. Dar buvo atsižvelgiama į atstuma nuo objekto, kuo arčiau objekto, tuo didesnis aplinkinis įspaudimas buvo.

Su paprastu mašinos modeliu tai veikė visai gerai, tačiau daug laiko užtruko fizikos apskaičiavimai. Tai ypač jautėsi norint didesnio tikslumo su tankesniais heightfield'ais, arba norint pridėti aukstesnio detalumo modelį.

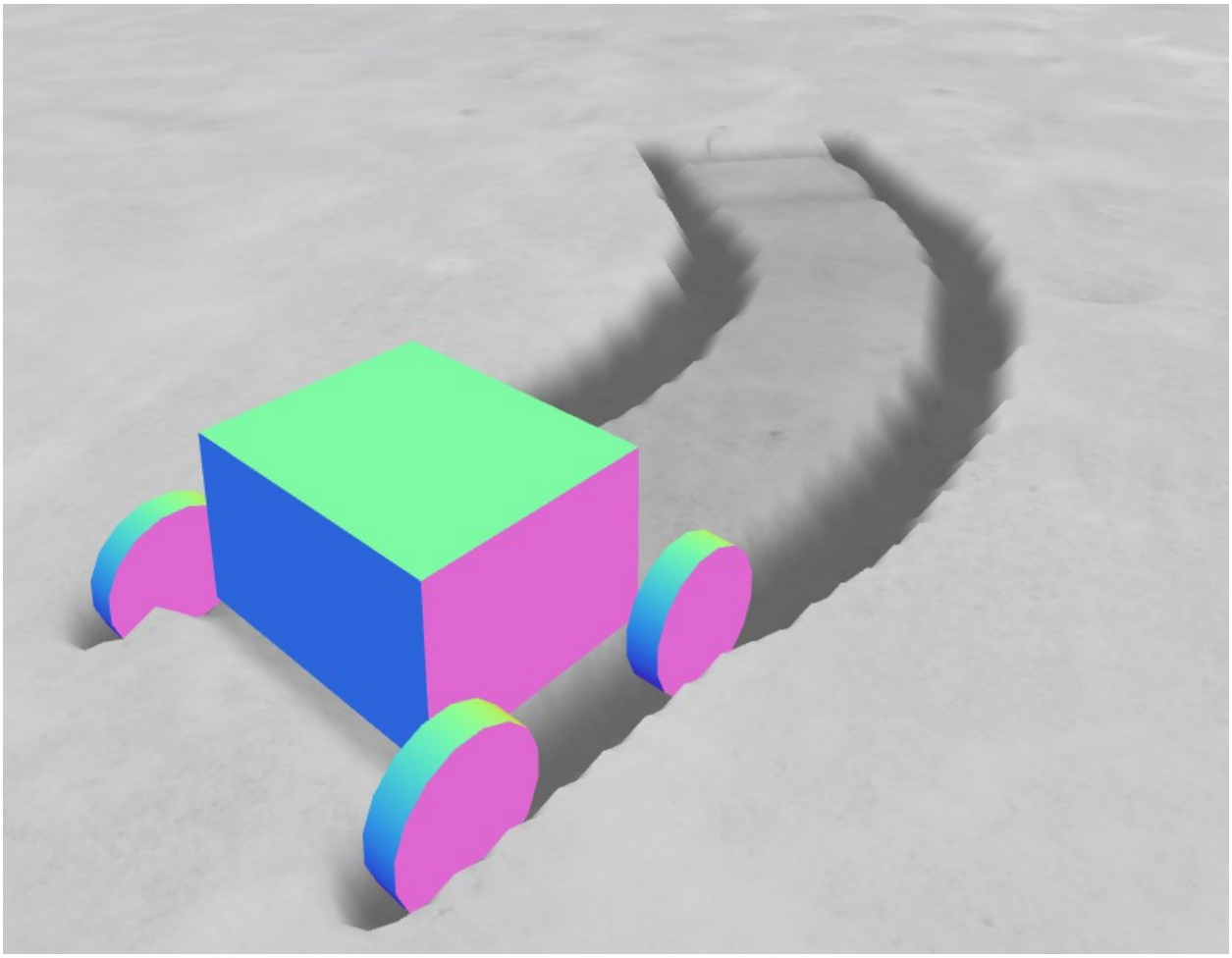

Figure 3: paprastas modelis

Jei į scena idedamas detalesnis modelis, reiktu rinktis tarp labai ilgų apskaičiavimu, kadangi collision mesh'ai būtu labai komplikuoti, arba modeliui priskirti paprastesni collision mesh'a, bet tada deformacija netiksliai atitiktu tai kas vaizduojama ekrane.

Taip pat, šitas metodas nesuteikia pakankamai tikslumo, kad atvaizduoti padangos rašta sniege.

Antrinis variantas - Scenos atstumo buferio paremta deformacija

Kadangi grafinės kortos yra ženkliai spartesnės negu paprasti procesoriai, kodėl nepasirėmus jų sugebėjimais kad paspartinti, bei padaryti deformacija tikslesne?

Grafinės kortos turi galimybe šalia įprasto piešimo, kartu nupiešti scenos ‘gyluma’ (depth).

Kadangi šitas depth bufferis atvaizduoja visa tai ka mato kamera, pačio gylumo raiška labai aukšta - visokie maži įdubimai modeliuose atsispindi juose.

Antras variantas visiskai nenaudoja fizikos bibliotekos deformacija (ji naudojama tik mašinos judėjimui). Vietoj to, mes naudojama kelis papildomus render target’us (tekstūras, į kurias galime piešti scenas).

Pirmas render target’as yra mūsu įprasta scena, tačiau naudojame kamera, kuri yra padėta po plokštuma, žiūrint aukštyn. Jos near/far clipping plane’as nustatytas kad būtu tarp mūsu sniego minimalios ir maksimalios deformacijos (t.y. tolimesni objektai nesimato). Kadangi sniego plokštumai mes piešiame tik išorinius face’us, kameros vaizduo pats sniegas nekliudo (jei kliudytu, yra imanoma išjungti laikinai objektų matomuma scenoje).

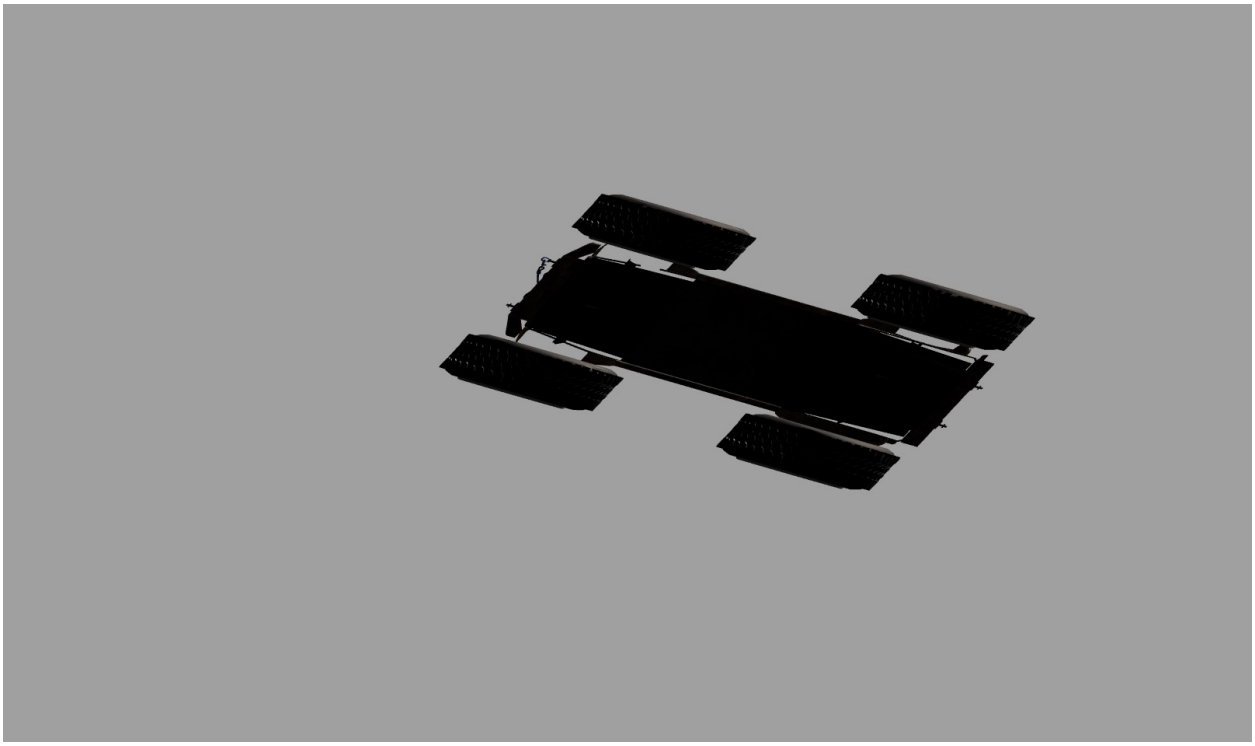

Figure 4: pirmas render pass

Iš pirmo render target’o, mes paemema detalu depth bufferi - jame mes matysime kur šiuo metu yra objektai, kurie turėtu deformuoti sniega.

Antrame render target’e, mes paemame pirmo render target’o tekstūra, bei kartu mūsų dabartine išorės tekstūra. Rezultate, mes sujungiame dvi tekstūras taip, kad ten kur buvo deformacija, ji liktu, o ten kur nebuvo - atsirastu (t.y. `spalva = min(dabartinio_gylio, praeito_gylio)`).

Tada, šia tekstūra mes su CPU perpiešiam ant canvas tekstūros, kuri naudojama mūsu sniegos deformacijos plokštumoje.

Galū gale naudojame trečia render target’a kad nupiešti įprasta sceną.

Šis metodas leidžia mum turėti ženkliai didesnia deformacijos raiška, naudojant ženkliai detalesnius modelius, ir vistiek būti ženkliai greičiau negu praeitas sprendimas - be jokių brangių fizikos objektų susidurimo apskaičiavimu - deformacijai fizikos sistemos išvis nereikia.

Siekiant didesnio tikslumo smarkiai padidinau tekstūros dydi, tačiau naudojant 2K (2048x2048) dydžio tekstūras - pastebėjau kad FPS’ai smarkiai sumažėjo.

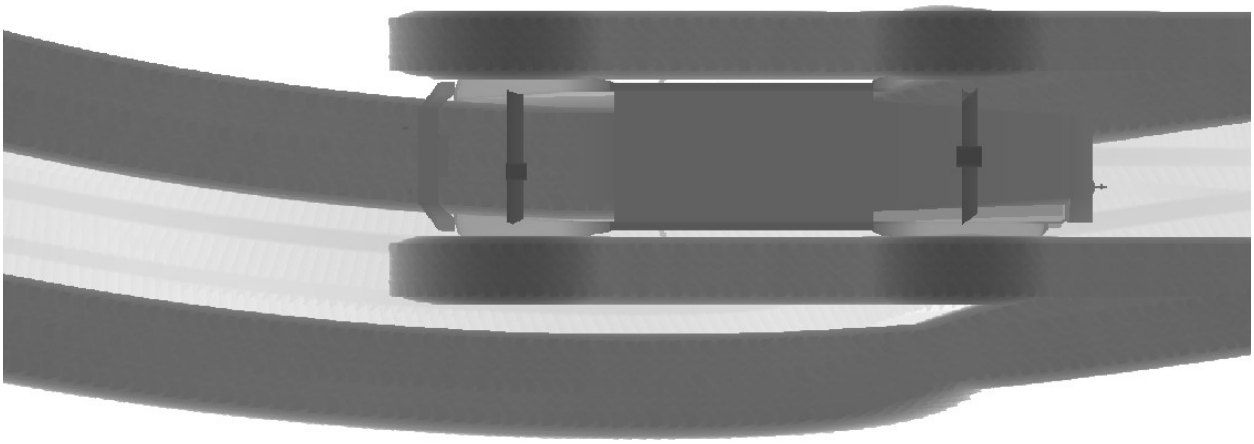

Figure 5: antras render pass

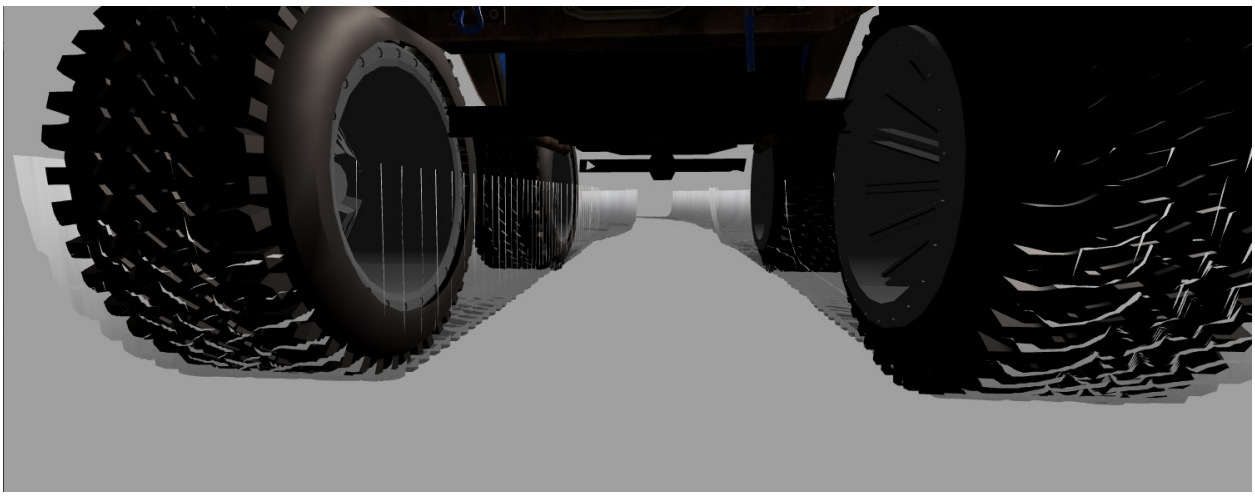

Figure 6: trecias render pass

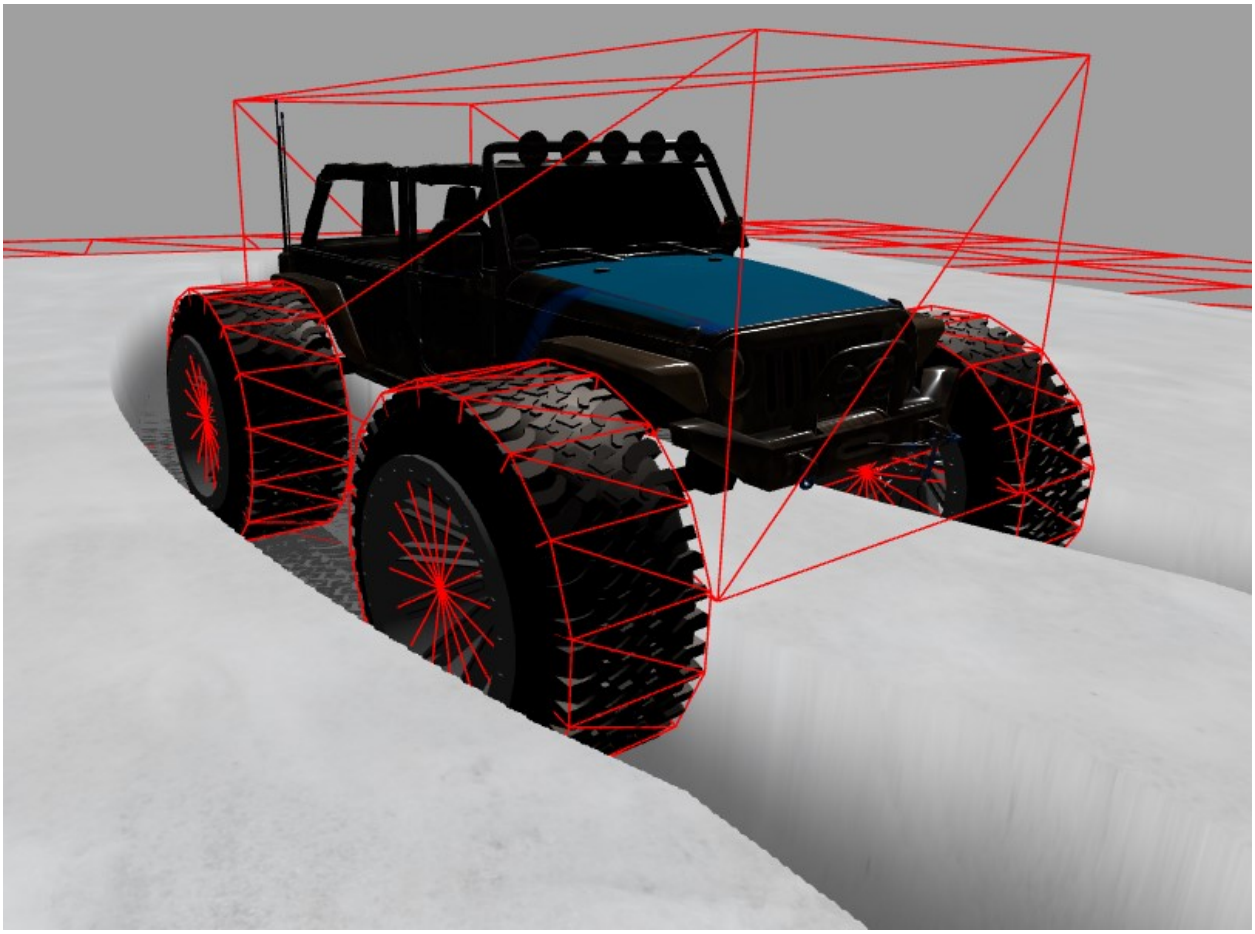

Figure 7: fizikos wireframe

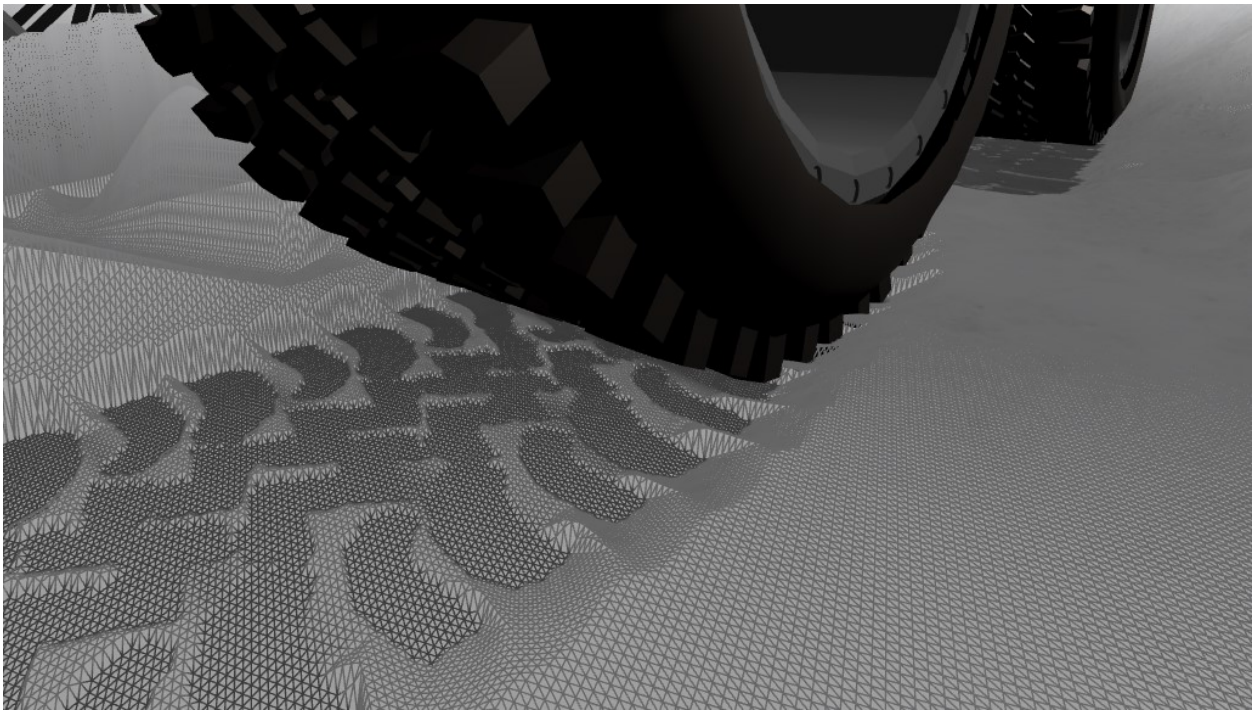

Figure 8: daug vertexu

Atlikdamas truputi profilinimo supratau, kad lečiausia dalis yra tas perpiešimas iš antro render target'o į CPU atmintyje esančia tekstūra.

Galutinis variantas - Optimizuotas variantas

CPU atmintyje esanti tekstūra buvo ganėtinai svarbi - jos deka galėjome matyti šone ekrano dabartinia sniegos deformacijos tekstūra, bei piešti ant jos su pele. Tačiau norėjau pasižiurėti ar imanoma būtu kaip nors panaikinti šita žingsnį.

Kad jį panaikinti, pasinaudojau dar papildomu render target'u (alternatyvus sprendimas būtu sukurti dar viena tekstūrą). Su šiuo variantu, pavyko pasiekti 180fps (max ekrano refresh rate, matuota su RTX 3080).

Potencialus optimalesnis variantas ant mažiau galingu grafiniu kortų

Dėja, ant letėsnių kompiuteriu, FPS'ai vistiek nėra itin geri. Su M1 macbook pro - tik 60fps su 2048x2048 tekstūra + plokšuma.

Spėju, kad tai dėl to kad plokštuma turi labai daug verteksu. Šita galima apeiti naudojant 'tesellation', kad vietose kur deformuota suteiktu daugiau verteksu, o kitose vietose nedaug, tačiau GPU accelerated variantas iš WebGL nėra pasiekiamas, tai tektu daryti on-CPU, kas greičiausiai vistiek būtu lėčiau, negu dabar yra.