

# Gruppe 6: Steinras Simulering



Markus Løtveit



Markus Pedersen



Runar Straume



Stian Aas Trohaug

Dato: 17.11.2022

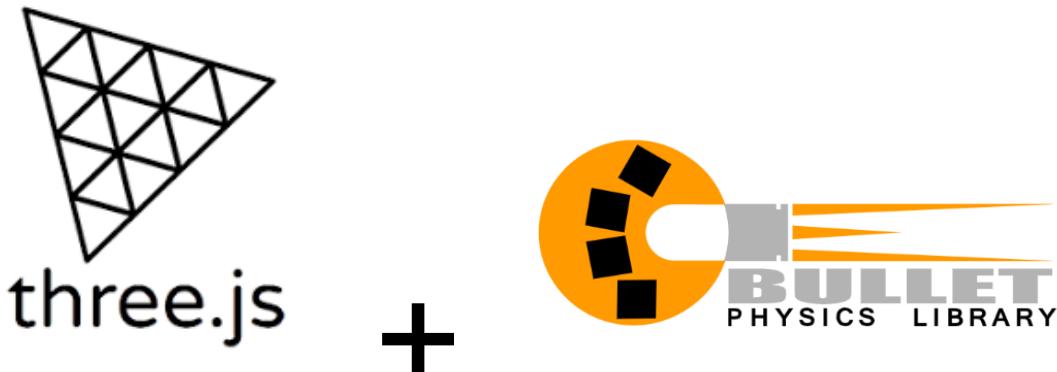
<b>Sammendrag</b>	<b>3</b>
<b>Innledning</b>	<b>3</b>
<b>Oppgave</b>	<b>3</b>
Beskrivelse	3
Scenegraf	4
<b>Løsningsmetode</b>	<b>4</b>
Utviklingsmetode og Verktøy	4
Oversikt over ukentlige sprinter og arbeidsmetode	5
Teknologier og Arkitektur	6
Ammo fysikk	6
Photogrammetry med Agisoft Metashape	6
Terreng	7
Steiner	8
Veien	9
Arkitektur	10
Detaljert klasse oversikt	11
<b>Konklusjon</b>	<b>12</b>
<b>Videre Arbeid</b>	<b>12</b>
Terrenget	12
Shader teknikker	12
Utskrift av data	13
Regulerbar rasvoll og fangnett	13
Strukturen i koden	14
Kjente problem	14
<b>Referanser</b>	<b>15</b>

## Sammendrag

Det er ingen hemmelighet at vestlandet med sine høye fjell, dype daler og store mengder nedbør er meget rasutsatt. Det var denne tanken kombinert med en imponerende fysikk demo av Marius Irgens som gav liv til ideen om simulering av steinras. For å få til en troverdig simulering var det flere ting som måtte på plass. Det første var å implementere det bibliotek for fysikk simulering, der landet gruppen på Ammo etter anbefaling fra Marius. Videre måtte objektene ha noenlunde realistiske former og proporsjoner, ble løst ved at steiner fra naturen ble skannet inn for så å importeres til Three.js prosjektet. Noen utfordringer dukket opp underveis, sånn som for eksempel importering av 3d modeller i gltf formatet og integrering av fysikk biblioteket med terreng generatoren, men gruppen kom i mål til slutt.

## Innledning

Målet med oppgaven var å lage et proof of concept for simulering av steinras. Dette konseptet kan så taes videre og brukes som et ledd i kartlegging av rasfare og rassikring. Denne rapporten vil videre ta for seg konkret hvordan dette proof of concept skulle implementeres, hvilke teknologier som ble brukt, og det endelige sluttresultatet.



# Oppgave

## Beskrivelse

Lage en simulering av steinras med følgende komponenter:

- Fjellside
- Steiner i ulike fasonger og størrelser
- Trær
- Realistisk fysikk for ovennevnte komponenter
- Kjøretøy

Videre skal flere avanserte teknikker fra kurset tas i bruk for å løfte scenen grafisk slik som:

- Shadows
- Photogrammetry
- Heightmap
- Terrain
- Plassering av objekter
- Normalmap eller bumpmap
- Roughmap
- Skybox

## Scenegraf

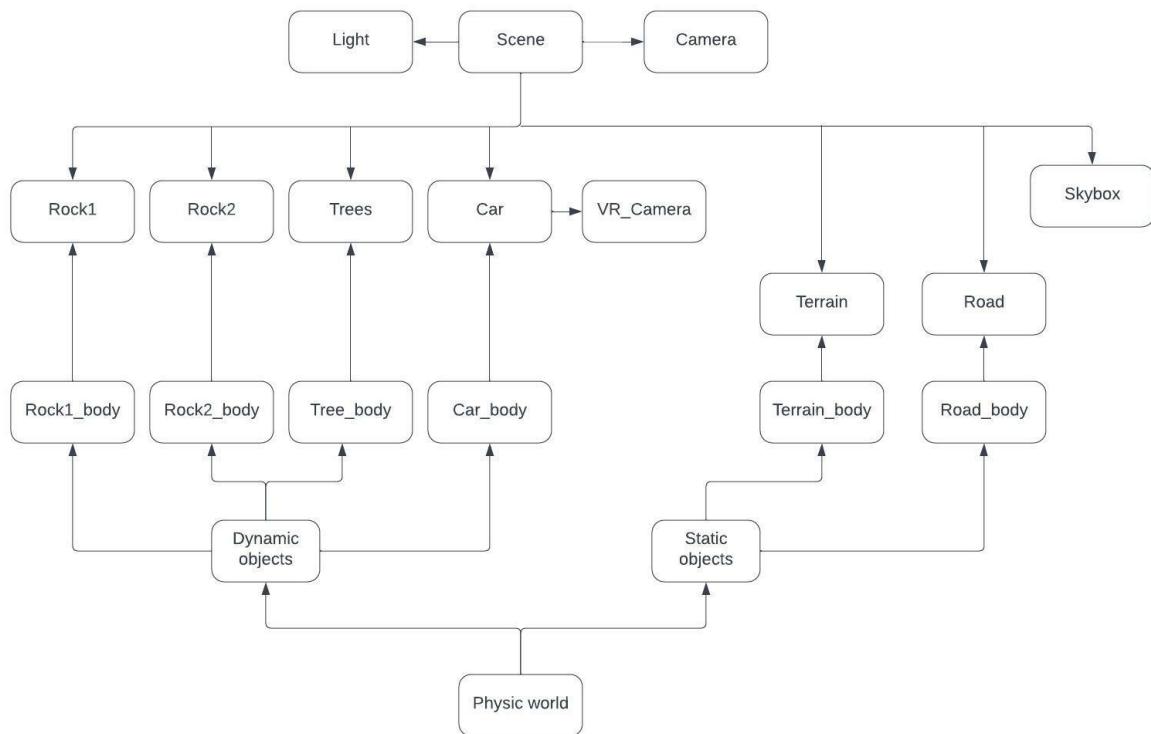


Fig. 1. Scenograf over begge verdener.

# Løsningsmetode

## Utviklingsmetode og Verktøy

Tidlig i prosjektet ble det satt opp et felles git repository som alle jobbet mot. Gruppen har arbeidet iterativt med oppgaven med klare definerte roller.

- Runar Straume: Prosjektleder og fysikk spesialist, hovedansvar for implementering av Ammo og github.
- Markus Løtveit: 3D grafikk spesialist, ansvar for 3d scanning av objekter.
- Markus Pedersen: Terreng ansvarlig og utvikler
- Stian Aas Torhaug: Shader og utvikler

Posisjon for mål er lagt til	origin & Bil	origin & main	Markus Pedersen	37 minutes ago
Vegen har nå en fysisk kropp og bilen starter ikke å kjøre før raset går			Markus Pedersen	Today 12:00
Lagt til vr-kamera på bilen. Bilen stopper nå ved raset			Markus Pedersen	Today 11:06
Lagt til veg og bildummy med en centralForce. Mangler riktig forflyttelse til bilen			Markus Pedersen	Yesterday 16:13
<b>renamed html file</b>	origin & moveIndex		Runar	<b>Yesterday 12:00</b>
<b>reIndexering</b>			Runar	<b>Yesterday 11:35</b>
Adjusted VR camera start pos			Runar Straume	Yesterday 01:29
Removed white border canvas			Runar Straume	Yesterday 01:01
adjusted buttons			Runar Straume	Yesterday 00:44
Added skybox			Runar Straume	Yesterday 00:19
adjusted fade inn/out sound	origin/MergeTest2		Runar Straume	07.11.2022 23:18
Merge branch 'Terreng' into MergeTest2			Runar Straume	07.11.2022 22:27
Added TextureSplatting and played around whit start position	origin & Terreng		Runar Straume	07.11.2022 01:09
Working physic body terrain			Runar Straume	06.11.2022 23:20
Terreng med ikke fungerende fysikk			Markus Løtveit	04.11.2022 14:42
<b>New rock, removed himlight</b>			Runar	<b>04.11.2022 14:13</b>
Importert terreng			Markus Løtveit	04.11.2022 13:06
<b>Merged ConvexHull into main</b>			Runar	<b>04.11.2022 12:57</b>
Merge remote-tracking branch 'origin/ConvexHull' into main			Runar	04.11.2022 12:50
<b>physic body fix</b>	origin & ConvexHull		Runar	<b>04.11.2022 12:49</b>
<b>changed tree shape to box</b>			Runar	<b>02.11.2022 14:02</b>
ConvexHull Rocks			Markus Løtveit	02.11.2022 12:49
<b>added geometry and javaDoc</b>			Runar	<b>02.11.2022 12:28</b>
lower mass for trees and more friction			Runar Straume	01.11.2022 20:30
Changed trees to dynamic	origin/Trees		Runar Straume	01.11.2022 19:51
Added spwan trees dummy			Runar Straume	01.11.2022 19:25
<b>Added sound + fading inn volume</b>	origin & Audio		Runar	<b>01.11.2022 15:52</b>

Fig. 2. Git commits som viser iterativt arbeid.

Som vist i [Fig. 2], er hver runding en commit på en mindre oppgave som en liten justering eller endring til et objekts posisjon eller egenskap. Linjene ut (i grønn og lilla) representerer egne brancher når det skal gjøres større kritiske endringer som krever grundig testing før de kan settes sammen med hovedprosjektet. Noen eksempel på slike oppgaver er å gi steinene fysisk korrekt kropp, generert et terreng med fysikk, legge til nye funksjoner. Dette er en god metode for å sikre at en alltid har en fungerende versjon i hoved (main) branchen.

## Oversikt over ukentlige sprinter og arbeidsmetode

Gruppen har fulgt scrum metoden ved å jobbe agilt og utført korte ukentlige sprinter basert på våre ønsker/endringer og visjoner til prosjektet. Videre har prosjektleder gitt gruppemedlemmene ulike sprinter etter ønske og sikret at det alltid har vært en fremgang i prosjektet. Gruppen har møttes jevnlig på labbene i tillegg til digital kommunikasjon over discord/messenger. Her er en oversikt over det større sprintene som har blitt gjort underveis, mange av dei i egne brancher.

- Idemyldring, inspirasjon og valg av prosjekt.
- Main og start prosjektet ble opprettet på github og startkode ble lastet opp. Gruppen fikk god hjelp og inspirasjon av lab assistenten til å komme tidlig i gang med prosjektet.
- VR -branch ble lagt til i prosjektet, da de krever endringer i render (kritisk operasjon), ble dette som en egen kort sprint. På dette stadiet har prosjektet bare en scene med objekter som ruller ned på to flate bokser som er strukket ut til et plan.
- 3D-Graphics branch, modeller av første Stein etterfulgt av testing og endringer i størrelse, oppløsning, antall punkter var alt for stort i første utkast.
- Instansiering- branch, programmet hadde allerede problemer med at det var tungt å kjøre, så gruppen ble enige om å opprettet en sprint for å optimalisere programmet.
- Audio- branch, legger til ny funksjonalitet med lyd til programmet.
- Trees- branch, funksjon for å legge til ti prøve modeller av trær i et randomisert område blir lagt til.
- ConvexHull- branch blir lagt til som er en større oppgave med å generere korrekt fysisk kropp etter en 3D modell.
- Terren - branch blir opprettet for den større oppgaven med å generere terren og en matchende fysisk kropp, dette delen var utfordrende og tok flere uker.
- Bil - branch blir opprettet som legger til en bil og vei til scenen, samt logikken for bevegelse til kjøretøyet.
- Sluttrapport og videopresentasjon.

# Teknologier og Arkitektur

## Ammo fysikk

Ammojs[1] er et fysikk bibliotek til Threejs direkte portet fra C++ programvaren Bullet physics[2]. Biblioteket fungerer ved at det blir opprettet en egen fysikk verden. Geometrien blir så bundet opp mot den fysiske verden slik at endringer i fysikken påvirker geometrien. En viktig ting å ha i bakhodet er at den fysiske verden ikke nødvendig må samsvare 1:1 med den geometriske verden, for eksempel kan objekter som ikke trenger å simuleres nøyaktig få forenklede fysiske kropper slik som bokser eller kuler. I dette prosjektet ble simulering av steinen prioritert, derfor fikk de fysiske kropper tilsvarende geometrien til meshen, mens andre objekter som trær og bilen fikk bokser som fysisk kropp.

```
physicsWorld = new Ammo.btDiscreteDynamicsWorld(dispatcher, broadphase, solver, collisionConfiguration);
physicsWorld.setGravity(new Ammo.btVector3(0, gravity, 0));
```

Fig. 3. Kode som setter opp fysikk verden.

## Photogrammetry med Agisoft Metashape

For å lage realistiske steiner ble photogrammetry programvaren Metashape[3] tatt i bruk. Den fungerer ved at man stegvis bygger opp punkter ut fra en serie med bilder, for så å skape en mesh basert på de tidligere genererte punktene. Mer konkret er arbeidsflyten til Metashape følgende:

1. Align Photos(Her beregner programmet posisjonen de ulike bildene ble tatt fra)
2. Create dense cloud (Lag punkter ut fra bildene)
3. Create mesh (lag en mesh basert på punktene fra dense clouden)
4. Fjern unødvendige bakgrunnslementer om du ikke har gjort det i tidligere steg
5. Create texture (lag tekstur basert på farge informasjonen i bildene)

Prosessen i seg selv er ganske rett frem, men avhengig av objektet man skal skanne er det flere utfordringer som kan dukke opp underveis. Det første er å få alle sider av objektet. Hvordan skanner man undersiden av et objekt? Det finnes flere metoder, men en av de er å lage 2 separate 3d objekter, en av undersiden, og en av oppsiden, for så å slå de sammen etterpå. Dette ble gjort på følgende måte:

1. Ta bilder av objektet
2. Snu objektet og gjenta steg 1
3. Legg bildene i 2 ulike mapper og hiv de inn i Metashape
4. Gjør steg 1-4 i workflow beskrivelsen ovenfor for hvert objekt
5. Opprett mask for bildene basert på 3d objektene(må gjøres for begge objektene)
6. Align Chunks(prøver å sette sammen bildedata fra begge bildemappene)
7. Merge Chunks
8. Gjør steg 3-5 av workflowen beskrevet ovenfor for den sammensatte modellen.

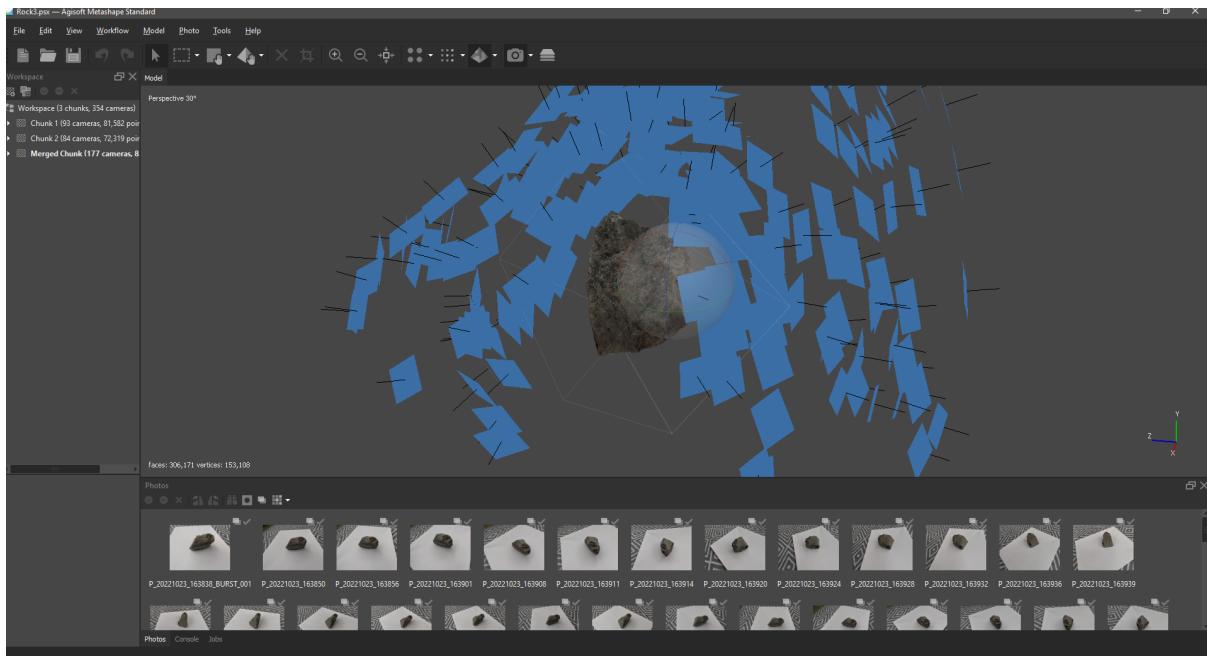


Fig. 4. De blå firkantene er posisjonen til hvert bilde og den svarte streken indikerer retning.

## Terreng

For terrenget bruker applikasjonen terregn generatoren som ble brukt tidligere i dette kurset. Generatoren baserer seg på heightmap som forskyver individuelle verticies langs y aksen. I tillegg valgte gruppen å bruke heightmappet fra kurset, siden dette inneholdt flere fjell kropper og hadde en flat base, passet det bra til en enkel demonstrasjon. Noe av grunnene til at dette passer er at en da kan plassere ut dynamiske fysiske objekt, som trærne, i stede for statiske objekter. Gruppen har til nå ikke funnet en metode å fysisk forankre dynamiske objekter til et skråplan, som da fører til at objekten bare vil skli/rulle nedover. I tillegg når trærne er dynamiske objekter, er de mer realistisk og ikke minst kjekt å se på.

Til å generere den fysiske kroppen, baserte gruppen seg på koden fra kripken/ammo.js [4], linje 238 til 297, med et par egne modifikasjoner, slik at det fungerte med et heightmap, og ikke med en matematisk sinus kurve. vår modifikasjon er vist i [Fig. 5].

```
terrainData = getHeightmapData(image, resolution);
for (let i = 0; i < terrainData.length; i++) {
    this.attributes.position.setY(i, terrainData[i] * height);
}
```

Fig 5: Kode for å hente ut høydedata fra et bilde.

Videre i koden ender en tilsutt opp med en fysisk kropp i 2D, så en må da strekke ut y-aksen, slik at en får høyden til det grafiske bilde en ser på skjermen. Dette løses med denne lille kodesnudden vist i [Fig. 6].

```
//we want to scale y to our height
heightFieldShape.setLocalScaling( new Ammo.btVector3( 1, height, 1 ) ); //X,Y,Z
```

Fig. 6. Skalering av heightFieldShape fra 2D til 3D

## Steiner

Applikasjonen har to 3D modeller (svart ruglete og rund lys) som lastes inn i simulatoren som glb filer. Desse lastes inn kun en gang med GTLF lasteren. Da denne lasteren er ganske ressurskrevende, er dette noe en ønsker å bare gjøre en gang og ikke for hver eneste stein. Deretter plasserer en hver av steinen modellene inn i en array hvor indexen bestemmer hvilken modell de er samt at en også får mappet de etter indexen de har. Slik kan en videre behandle de individuelt med genereringen av randomiserte størrelser.

Til å skalere steinene korrekt, ble det nødvendig å lage en egen metode for å skalere hvert eneste punkt til 3D modellen. Dette kunne ikke gjøres så enkelt som å bruke ferdige metoder som Rocks.scale.set(x,y,z) fordi dette vil kun skalere det synlige 3D objektet på skjermen, og ikke dens fysiske kropp. For å løse dette problemet slik at en får en fysisk kropp som matchet den synlige, henter programmet ut alle posisjonene til alle vertex (punkt) til 3D objektet. Deretter skaleres de etter en randomisert størrelse (size) som vist i kodesnuttten i [Fig. 7].

```
//Manually scale new physic body from rocks
let geo;
if (isOdd(counter) == 0) {
    geo = new Float32Array (Rocks.geometry.getAttribute('position').array);
    for (let i = 0; i < geo.length; i++){
        geo[i] = geo[i]*(size);
    }
} else{
    geo = new Float32Array (Rocks.geometry.getAttribute('position').array);
    for (let i = 0; i < geo.length; i++){
        geo[i] = geo[i]*(size*4)
    }
}
```

Fig. 7. Viser hvordan man kan hente alle punkt til en 3D modell. Note, 3D modellen til de runde steinene, som er partalls steiner, har en mindre modell som er grunnen til size \* 4

## Veien



Fig. 8. Viser et nærbilde av veien for å vise detaljene.

En typisk vestlands vei er ikke en flat vei, så her har gruppen tatt i bruk flere avanserte grafikk metoder. Først ble de valgt å bruke MeshStandardMaterial [5] fra three.js. Som dokumentasjonen viser er dette et materiale som gir bedre realistisk resultat enn phong og andre materialer. Også bruker dette materialet shading per fragment som gir finest resultat mot en pris i ytelse. Men her snakker en om en kort vei stub, så det er innenfor.

[Fig. 8] viser veien med:

- roadBaseColor, som er teksturen.
- normalMap, som justerer normalen til hver piksel fragment og påvirker hvordan lyset lyser opp fargene i teksturen. Kartet er i RGB, hvor blåfargen er den sterkeste og rød lavest.

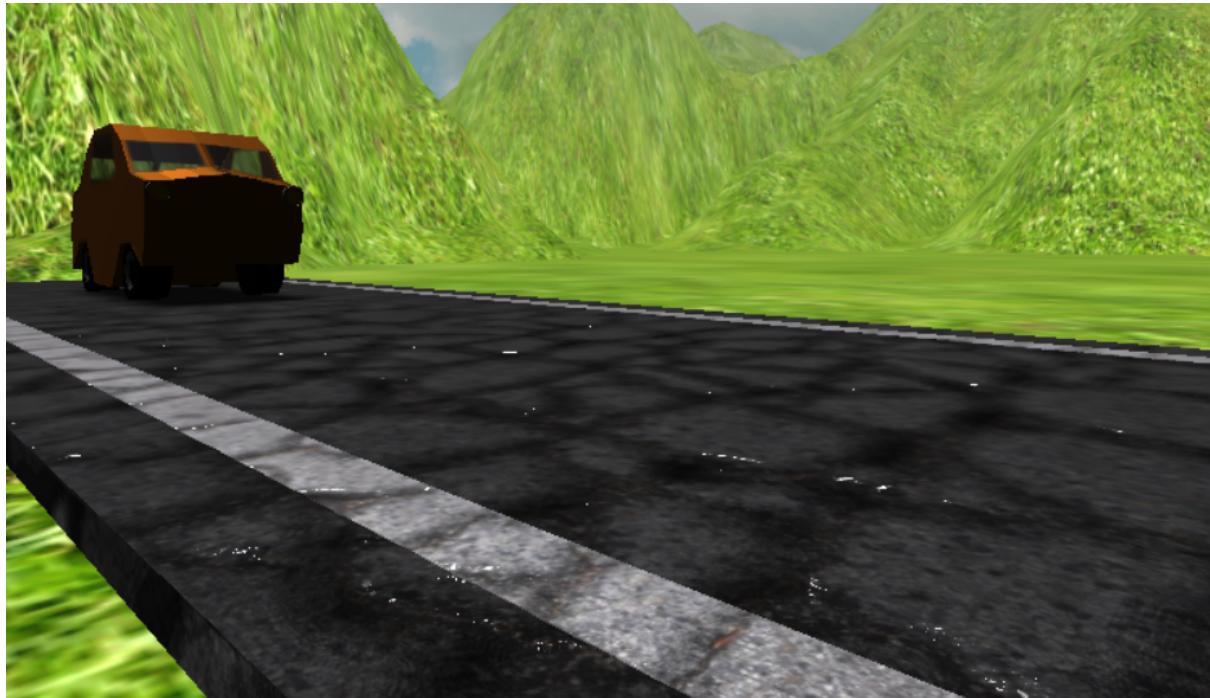


Fig. 9. Viser en annen teknikk som er brukt i prosjektet

[Fig. 9] viser veien med:

- road Base Color, som er tekturen.
- bump Map, brukt gråtonene i displacement map der hvit er det høyeste punktene og sort de laveste. Videre er effekten skalert ned (til 0.2) for å ha en lavere innvirkning.
- Roughness Map, bruker den grønne biten (G) i kartet til å gi en refleksjon av lyset, som gir en illusjon av våt vei i vårt tilfelle.

## Arkitektur

Eksterne biblioteker er lagt i egne mapper og importert som moduler. Resten av koden, inkludert alle funksjonene som er laget er lagt inn i main.js. Dette viste seg etterhvert å være et dårlig valg ettersom det ble mye merge konflikter mellom ulike greiner i git repositoriet. Ved en annen anledning ville en selvsagt flyttet flere av funksjonene ut i egne klasser, men grunnet tid og mulige komplikasjoner ved samspillet mellom three.js og fysikk motoren, ble ikke dette etterarbeidet utført i prototypen. Prosjektstrukturen vises i [Fig. 10].

## Detaljert klasse oversikt

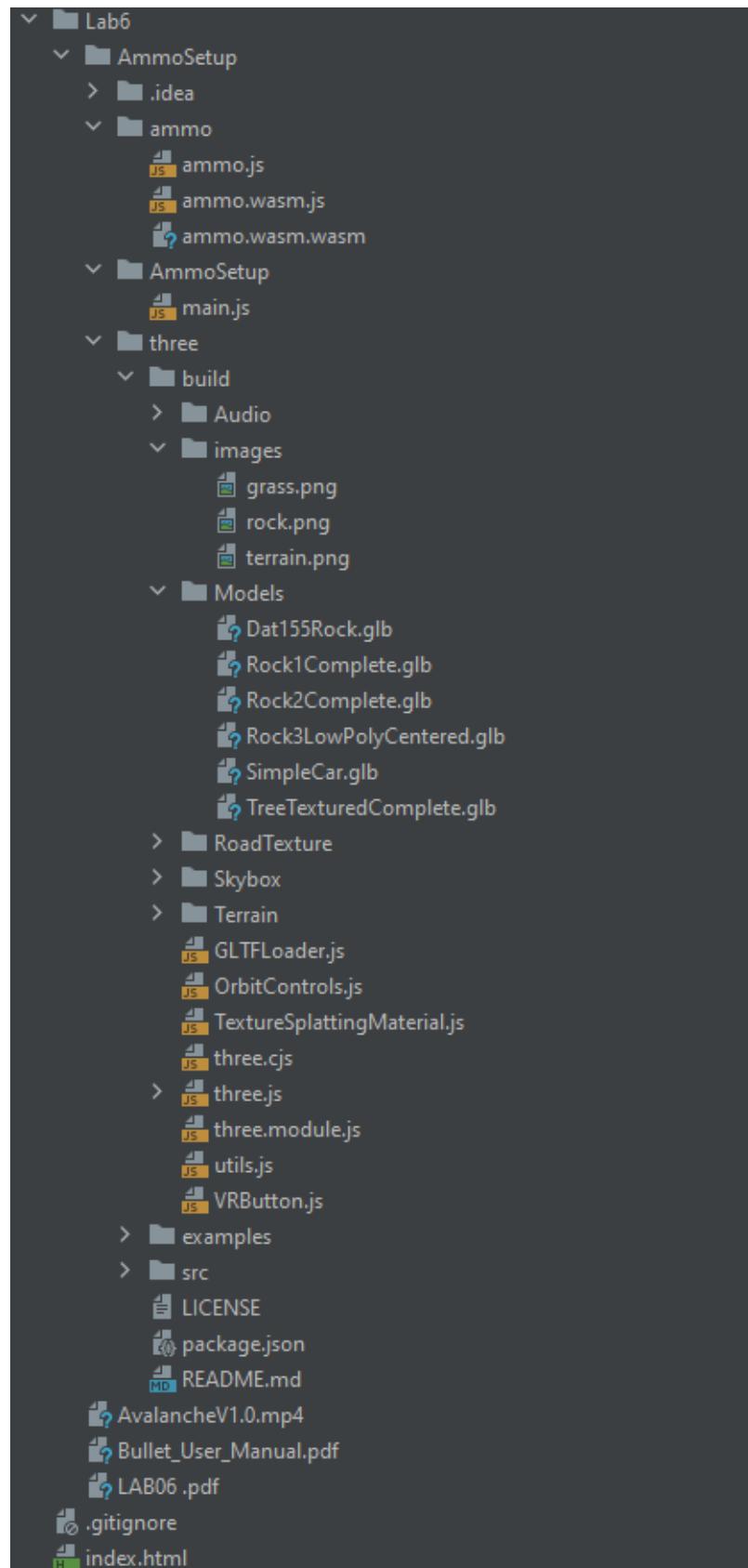


Fig. 10. Oversikt over prosjektstrukturen.

# Konklusjon

Tross noen utfordringer underveis kom gruppen i mål med de satte målene. Alle objekter i scenen, inkludert terrenget er integrert med fysikk biblioteket. Skanning og importering av 3d objekter i scenen gikk også fint, om enn med noen små problemer underveis. Også var det kanskje litt dristig og gå for en simulator da dette har gitt betraktelig mer arbeid i prosjektet i form av testing, samspill mellom objekter, sette seg inn i bullet api og mye teknisk arbeid. Også var det ikke så lurt å tenke at en bare flytter koden ut til egne klasser i ettertid da dette muligens kan bryte samspillet mellom objektene eller skape nye bugs. Derfor ble dette satt til etterarbeid så en ikke risikerte og potensielt bryte heile simulatoren på slutten av tiden satt av til prosjektet.

## Videre Arbeid

### Terrenget

Per dags dato bruker prosjektet heightmappet fra terrenget kurset som ble holdt tidligere i semesteret. Med mer tid ville dette vært byttet ut med et heightmap importert fra det norske kartverket slik at man fikk gjengitt et realistisk terrenget. Videre hadde det vært gunstig å ha støtte for at brukere kunne importere sitt eget heightmap slik at man kunne simulert ras på den aktuelle strekningen.

Naturlig nok var det ikke mulig å lage en realistisk simulering i løpet av de ukene som var tilsidesatt for prosjektet. Den grunnleggende fysikken i seg selv er ikke et stort problem ettersom bibliotek for dette allerede eksisterer, men å lage et realistisk terrenget med alle de objektene som man naturlig vil finne langs en fjellside er både tidkrevende og krevende å prosessere. Friksjonen til underlaget steinene ruller på vil også variere avhengig av faktorer som underlaget, været og andre faktorer. Det ville derfor også være naturlig gitt mer tid å 3d skanne flere objekter samt arbeide med optimalisering av scenen.

### Shader teknikker

Applikasjonen har et ganske rett terrenget på bakkenivået hvor bilen kjører. Noe som ikke minner om en typisk vei på Vestlandet. Her har gruppen eksperimentert litt med en egen shader som lager svingninger i terrenget, selv om det egentlig er flatt. Men siden flere av materialene ikke virker med dette, samt at fysikken selvsagt blir helt feil, har ikke dette blitt implementert i prototypen. Men viser her til et bilde [Fig. 11] som viser mulighetene som finnes ved bruk av egne shadere. Modifikasjonene og koden ligger i CustomTextureSplattingMaterial.js filen, og denne versjonen er tilgjengelig i branch custom\_shaders på github.



Fig. 11. Viser at terrenget og veien er vrid med en sinus kurve ved hjelp av shader kode.  
Veien er egentlig fortsatt en rett linje.

## Utskrift av data

Visning av kollisjons data over for eksempel, hvilke krefter har bilen blitt utsatt for etter at raset er over. En må da også angi korrekt masse, friksjon og andre realistiske parameter fra den virkelige verden.

## Regulerbar rasvoll og fangnett

For å gi applikasjonen større bruksområde bør det implementeres mulighet for en rasvoll mellom fjell og vei som man kunne regulere høyden på. Også bør simulatoren ha mulighet til å plassere ut fangnett, se fig. 12, i terrenget. Med dette kan simulatoren brukes i plannlegging av rassikring langs rasutsatte veier.



Fig. 12, [6]. Viser fangnett, designet til å stoppe steiner på opp til 8 tonn i en fart på 90 km/t

## Strukturen i koden

Rydde opp i ubrukt kode, se over potensielle problemer i importer på ulike nettsesere, bedre klasse struktur og mer optimalisering.

## Kjente problem

Videre ettersom programmet vokser har gruppen notert seg følgende feil i programmet som det ikke har vært tid og arbeidskapasitet til å løse, da de ikke er kritiske for prototypens funksjonalitet.

- En liten høyde feil i det fysiske terrenget, man kan se at trær og steiner svever litt over bakken på det laveste nivået i terrenget. Problemet kan muligens ligge i det som er i koden fra kripen/ammo/.js [4] på linje 289 og 290.
- Vinduene på bilen som av og til ikke laster inn som gjennomsiktige. Dette henger nok sammen med at innlasting ikke skjer i korrekt rekkefølge. Denne feilen er litt kritisk da en ikke ser noe ut av bilen i VR, når den forekommer.
- Noen få steiner, særlig de minste, faller av og til gjennom terrenget sin fysiske kropp. Henger muligens sammen av at de er for små, og ingen kollisjon med terrenget blir fanget opp ved stor hastighet, eller at terrenget er for strekt ut i y- aksen og det er et faktisk lite hull der de faller igjennom. Feilen er av liten betydning da det er nok av steiner i raset.

## Referanser

- [1] Ammojs.(2022). kripken. Lastet ned: Oct. 20, 2022. [Online]. Available: <https://github.com/kripken/ammo.js/>
- [2] Bullet Physics SDK Manual. (2.83). Lastet ned: Nov. 17. [Online] Available: [https://github.com/bulletphysics/bullet3/blob/master/docs/Bullet\\_User\\_Manual.pdf](https://github.com/bulletphysics/bullet3/blob/master/docs/Bullet_User_Manual.pdf)
- [3] Agisoft.(1.7.5). Metashape. Lastet ned: Oct. 20, 2022 [Online]. <https://www.agisoft.com/>
- [4] kripken, “webgl\_demo\_terrain.” Github.com Hentet fra: [https://github.com/kripken/ammo.js/blob/main/examples/webgl\\_demo\\_terrain/index.html](https://github.com/kripken/ammo.js/blob/main/examples/webgl_demo_terrain/index.html) (Lastet ned: Nov. 17, 2022).
- [5] Threejs, “MeshStandardMaterial.” Threejs.org. Hentet fra: <https://threejs.org/docs/#api/en/materials/MeshStandardMaterial> (Lastet ned: Nov. 17, 2022).
- [6] Aftenbladet. Lastet ned Nov. 17, 2022. [Online]. Available: <https://www.aftenbladet.no/lokalt/i/5LgBX/dette-nettet-skal-stanse-8-tonn-stein-i-90-km-t>