Mappeoppgave Visualisering og Simulering

Anders P. Åsbø | kandidat.nr.: 843
Høgskolen i Innlandet

Sammendrag

Innhold

Introduksjon

Et viktig aspekt av moderne beredskap er forståelsen av ekstremvær og dets effekt på lokal natur. Derfor er det nødvendig å kunne lage digitale representasjoner av virkelige terreng, samt simulere fysikk ved bruk av datamaskin på slike representasjoner. I denne rapporten presenterer jeg en metode for å modellere, samt simulere effekten av nedbør på terreng konstruert fra punktskydata.

Rapporten fokuserer på hvordan lage en regulært indeksert trekantflate som representerer terrenget, samt hvordan bruke B-Splines til å kartlegge vassdrag som dannes ved ekstrem nedbør, og hvordan simulere effekten et slik vassdrag har på løsmateriale.

Metode

Punktsky og triangulering

Vertekser

Moderne målinger av terreng gjøres med LiDAR, og resulterer i rådata i form av en uorganisert samling punkter i $\mathbb{R}^3$ relativt til et valgt koordinatsystem (Berger mfl., 2017). Jeg har valgt å laste ned punktdata fra Kartverket sin database 'hoydedata.no' i '.laz'-format, som jeg så konverterer til en rentekst-fil ved hjelp av programvaren 'LASzip' (Isenburg, 2019). Den resulterende tekstfilen inneholder da n antall punkter fordelt på formen

$$\begin{matrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ \dots & \dots & \dots \\ x_{n-1} & y_{n-1} & z_{n-1} \end{matrix}$$

hvor hver linje svarer til et punkt. For å kunne visualisere punktskyen som en sammenhengende overflate, så konstruerer jeg et regulært rutenett i xy-planet med dimensjoner $\textbf{width} = x_{\max} - x_{\min}$ og $\textbf{height} = y_{\max} - y_{\min}$. Hvor $x_{\max}$, $x_{\min}$, $y_{\max}$ og $y_{\min}$ er henholdsvis største og minste verdi for x- og y-koordinatene til punktene. Jeg velger så at hver rute i det regulære rutenettet er et kvadrat med areal $10\,\mathrm{m}^2$ (steglengde $h = 10\,\mathrm{m}$), slik at $n_x = \lceil \textbf{width}/h \rceil$ og $n_y = \lceil \textbf{height}/h \rceil$ er henholdsvis antall ruter i $x-$ og $y-retning$. Hvert punkt kan så sorteres inn i ruten som dekker x- og y-koordinatene dens ved å regne ut rad- og kolonneindeks i og j i rutenettet som

$$\begin{aligned} i &= \lfloor \frac{x - x_{\min}}{h} \rfloor \\ j &= \lfloor \frac{y - y_{\min}}{h} \rfloor. \end{aligned}$$

For hver rute kan det regnes ut en gjennomsnittlig z-koordinat ved å ta gjennomsnittet av z-koordinatene til alle punktene med tilsvarende rad- og kolonneindeks

$$\bar{z}_{i,j} = \frac{1}{n_{i,j}} \sum_{k=0}^{n_{i,j}} z_{i,j,k},$$

hvor $n_{i,j}$ er antall punkter innenfor ruten som tilsvarer rad i kolonne j. Hvis en rute skulle vise seg å ikke inneholde punkter, så kan gjennomsnittshøyden i den ruten regnes som gjennomsnittet av gjennomsnittshøydene til naborutene

$$\bar{z}_{i,j} = \frac{1}{8} \sum_{k=i-1}^{i+1} \sum_{\substack{l=j-1 \\ k \neq i \vee l \neq j}}^{j+1} \bar{z}_{k,l}.$$

Hvis dette må regnes ut for en rute på kanten av rutenettet, så må alle naboer som ikke eksisterer ekskluderes fra beregningen over. Et sett med regulært fordelte punkter i xy-planet, sentrert rundt origo, kan skrives til fil på formatet

$$\begin{aligned} &n_x \cdot n_y \\ &(i \cdot h - n_x \cdot h/2, \quad \bar{z}_{i,j} - (z_{\max - z_{\min}})/2, \quad j \cdot h - n_y \cdot h/2) \\ &... \end{aligned}$$

for alle $j = 0, 1, 2, ..., n_y - 1$ for alle $i = 0, 1, 2, ..., n_x - 1$. Første linjen i filen inneholder antall punkter. Videre er hver koordinat translatert slik at origo er sentrert, og y- og z- koordinaten er byttet om for innlesning i Unity («Unity Engine», 2023) som bruker et venstrehendt koordinatsystem med positiv y-akse som oppoverretning.

Indeksering

For å kunne tegne en trekantflate av de regulært fordelte punktene i underseksjonen 'Vertekser', så må det konstrueres en regulær triangulering med punktene som vertekser i trekanter. Dette gjøres ved å ta for seg alle kvadrater hjørner $\vec{v}_{j+i\cdot n_y}$, $\vec{v}_{(j+1)+i\cdot n_y}$, $\vec{v}_{j+(i+1)\cdot n_y}$ og $\vec{v}_{(j+1)+(i+1)\cdot n_y}$, hvor $\vec{v}_k$ er de regulært fordelte punktene, og $j = 0, 1, 2, ..., n_y - 1$ for alle $i = 0, 1, 2, ..., n_x - 1$. Videre er det to trekanter T_{even} og T_{odd} per kvadrat med henholdsvis partallsindeks og oddetallsindeks.

Altså kan området deles opp i $(n_x - 1) \cdot (n_y - 1)$ antall kvadrater med en partallstrekant

$$T_{\text{even}} = 2(j + i \cdot (n_y - 1))$$

med verteks-indekser

$$j + i \cdot n_y, \quad (j + 1) + i \cdot n_y, \quad j + (i + 1) \cdot n_y,$$

og nabotrekanter

$$2(j + i \cdot (n_y - 1)) + 1, \quad 2(j + i \cdot (n_y - 1)) - 1, \quad 2(j + i \cdot (n_y - 1)) + 1 - 2(n_y - 1),$$

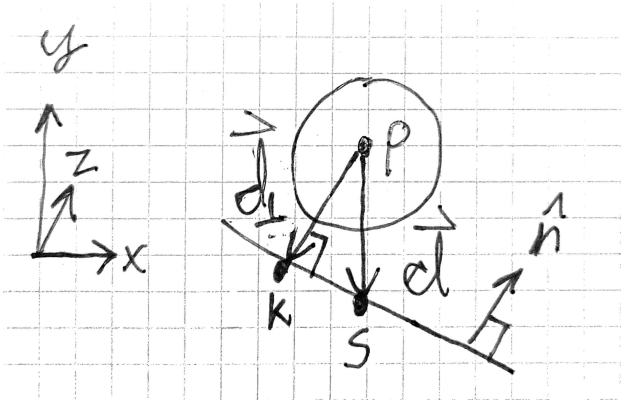
Hvor første verteks-indeks er motstående første naboindeks, også videre.

Ball på trekantflate

For å simulere baller lagde jeg klassen `BallPhysics` som legges til som komponent på en instance av Unitys innebygde sfære-primitiv. Klassen har en referanse til `TriangleSurface`-objektet den skal rulle på. Videre definerer jeg parametere for ballens masse [kg], radius [m], friksjon [dimensjonsløs] og bounciness [dimensjonsløs].

Kollisjonsdeteksjon

For å dettektere kollisjon med trekantflaten, så vil ballen kalle `GetCollision`-metoden i `TriangleSurface` objektet i hver `FixedUpdate`. Ballen gir da posisjonen til senteret sitt p til `TriangleSurface`, som bruker barysentriske koordinater til å finne trekanten ballen er på. Videre finner `TriangleSurface` punktet s på trekanten som er på linje med senteret i ballen langs y-aksen.



Figur 1

Diagram av kollisjonsdeteksjon. p er posisjonen til ballens senter, s er punktet på flaten med samme x- og z-koordinat som ballens senter, og k er punktet på flaten nærmest ballens senter.

For å sjekke om ballen er i kontakt med flaten beregner jeg først punktet k på flaten som er nærmest ballens senter p

$$k = p + \left((s - p) \cdot \hat{n}\right) \hat{n},$$

og sjekker om differansen $|p - k|$ er mindre eller lik radiusen r til ballen. I tilfellet hvor $|p - k| \leq r$, så er det kollisjon. For å unngå at ballen beveger seg gjennom kollisjonsflaten setter jeg så ballens posisjon til å være i radius avstand fra kontaktpunktet langs enhetsnormalen

$$p_{\text{korrigert}} = k + r \cdot \hat{n}.$$

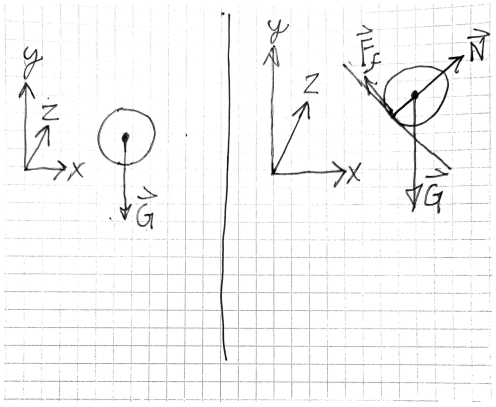
For å simulere spretting endrer jeg ballens hastighet til å bli

$$\vec{v}_{\text{etter}} = \vec{v}_{\text{før}} - (1 + b)\left(\vec{v}_{\text{før}} \cdot \hat{n}\right) \hat{n},$$

hvor $b \in [0, 1]$ er bounciness-parameteren til ballen, og representerer hvor stor brøkdel av normalkomponenten av hastigheten som skal reflekteres ved kollisjon. For $b = 0$ så mister ballen all hastighet normalt på planet. Hvis $b = 1$ så vil ballen få like stor hastighet normalt på planet som før kollisjonen, men i motsatt retning.

Glidefysikk

For å simulere fysikken på ballen, så finner jeg først alle kreftene på ballen i både tilfellet når den er og ikke er i kontakt med trekantflaten.



Figur 2

Frilegmediagram av ballen i både fritt fall, og på trekantflaten, med Unity sitt venstrehendte koordinatsystem tegnet inn.

I begge tilfeller er tyngdekraften på ballen lik

$$\begin{aligned}\vec{G} &= m\vec{g} \\ &= m\begin{pmatrix} 0 \\ -9.81\,\mathrm{m/s^2} \\ 0 \end{pmatrix},\end{aligned}$$

hvor m er ballens masse i kilogram.

I tilfellet hvor ballen er i kontakt med trekantflaten, så opplever ballen en normalkraft

$$\vec{N} = -\left(\vec{G}\cdot\hat{n}\right)\hat{n},$$

hvor $\hat{n}$ er enhetsnormalvektoren til trekanten ballen er i kontakt med. Videre modelerer jeg også en glidefriksjonskraft

$$\vec{F}_f = -\mu|\vec{N}|\hat{v}_\parallel,$$

hvor μ er rollingResistance-parameteren til `BallPhysics`-klassen, og $\hat{v}_\parallel$ er en dimensjonsløs enhetsvektor parallel med hastigheten til ballen projisert på trekanten ballen befinner seg på, slik at

$$\hat{v}_\parallel = \frac{\vec{v}_\parallel}{|\vec{v}_\parallel|} = \frac{\vec{v} - (\vec{v}\cdot\hat{n})\,\hat{n}}{|\vec{v} - (\vec{v}\cdot\hat{n})\,\hat{n}|}.$$

Her er $\vec{v}$ hastighetsvektoren til ballen. Friksjonskraften virker da alltid mot bevegelsesretningen langs flaten ballen er i kontakt med.

Jeg finner så akselerasjonen til ballen ved Newton's andre lov, slik at

$$\vec{a} = \frac{\vec{G} + \vec{N} + \vec{F}_f}{m},$$

og integrerer hastighet $\vec{v}$ og posisjon $\vec{p}$ med Forward-Euler algoritmen

$$\begin{aligned} \vec{v}_{t+\Delta t} &= \vec{v}_t + \vec{a} \cdot \Delta t, \\ p_{t+\Delta t} &= p_t + \vec{v}_{t+\Delta t} \cdot \Delta t. \end{aligned}$$

Rulling på flere trekanter

I tilfellet hvor ballen krysser fra en trekant til neste, så har jeg implementert to separate måter å håndtere det på. I tilfellet hvor ballen har en bounciness som er større enn 0, så håndteres all kollisjon med den delvise refleksjonen beskrevet i underseksjon 'Kollisjonsdeteksjon'

$$\vec{v}_{\text{etter}} = \vec{v}_{\text{før}} - (1 + b)\,(\vec{v}_{\text{før}} \cdot \hat{n})\,\hat{n}.$$

I tilfellet hvor ballens bounciness er satt til å være 0, så bruker jeg i stedet en refleksjon av planet med normal lik gjennomsnittsnormalen til de to involverte trekantene.

$$\vec{v}_{\text{etter}} = \vec{v}_{\text{før}} - 2\,(\vec{v}_{\text{før}} \cdot \hat{n})\,\hat{n},$$

hvor

$$\hat{n} = \frac{\hat{n}_{\text{før}} + \hat{n}_{\text{etter}}}{|\hat{n}_{\text{før}} + \hat{n}_{\text{etter}}|}.$$

Dette bevarer farten til ballen over trekantbyttet. Jeg har også valgt å kun gjøre denne refleksjonen i tilfellet hvor

$$(\hat{n}_{\text{før}} \times \hat{n}_{\text{etter}}) \cdot \left(\hat{n}_{\text{før}} \times \hat{v}_{\parallel}\right) < 0,$$

hvor $\hat{v}_{\parallel}$ er enhetsvektoren i samme retning som komponenten av ballens hastighet som er parallel med trekanten. Betingelsen over, sørger for at refleksjonen kun skjer hvis ballen beveger seg over et konkavt trekantbytte. Derimot, hvis trekantbyttet er konvekst, så vil ballens hastighet ikke reflekteres. Dette gjør jeg slik at for en vilkårlig trekantflate, så vil ballen oppføre seg som om flaten er glatt for konvekse skjøter, men være i stand til å forlate flaten og komme i fritt fall ved konvekse skjøter.

B-Splines og simulering av vassdrag

For å simulere vassdrag som produseres under ekstremnedbør, modelerer jeg et sett med enkeltregndråper i form av baller som følger fysikken beskrevet i forrige sekjson. Videre får hver ball en tilfeldig generert startposisjon innenfor et valgt volum over terrenget. Jeg velger så å logge posisjonen til hver av regndråpene et gitt antall ganger per tid, over en gitt tidsperiode. De lagrede posisjonene blir så brukt som kontrollpunkter i xz-planet for en $2D$, kvadratisk B-Spline som skal representere enkeltdråpenes bevegelse horisontalt langs terrenget. Splinen kan så tegnes ved å evaluere punkter for n_t uniformt fordelte verdier av parameteren t, som jeg tegner rette linjer mellom. De tegnede punktene løftes opp på terrengoverflaten ved å beregne høyde med barysentriske koordinater.

For å simulere effekten av et slik vassdrag på en kampestein eller annet løsmateriale, så bruker jeg likningen for motstandskraft i fluid

$$\vec{F}_D = \frac{1}{2} C_D \rho A |\vec{v}| \vec{v},$$

hvor C_D er dragkoeffisienten til legemet som påvirkes, ρ er tettheten til fluidet, A er tversnittarealet som står normalt på fluidets bevegelsesretning, og $\vec{v}$ er fluidets hastighet relativt til legemet (Alexander & Cooker, 2016, s.7). I min simulasjon bruker jeg en ball med masse $m_b = 50\,\text{kg}$ og radius $r_b = 1\,\text{m}$ plasert på terrenget. Jeg har valgt en dragkoeffisient på $C_D = 0.5$ siden det gjelder for sfærer i mange forskjellige typer flyt (Hall, 2023). Videre har ballen et tversnittareal $A = \pi r_b^2$.

Resultater

Diskusjon

Konklusjon

Referanser

Alexander, J., & Cooker, M. J. (2016). Moving boulders in flash floods and estimating flow conditions using boulders in ancient deposits (V. Manville, Red.). *Sedimentology*, *63*(6), 1582–1595. https://doi.org/10.1111/sed.12274

Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Guennebaud, G., Levine, J. A., Sharf, A., & Silva, C. T. (2017). A Survey of Surface Reconstruction from Point Clouds. *Computer Graphics Forum*, *36*(1), 301–329. https://doi.org/10.1111/cgf.12802

Hall, N. (2023 november). Drag of a Sphere. Hentet 27. november 2023, fra https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/drag-of-a-sphere/

Isenburg, M. (2019 november). LASzip. https://laszip.org/

Unity Engine. (2023 mars). Hentet 20. april 2023, fra https://unity.com