Interaktiv simulering av snett kast och fjäderrörelse

Komvuxarbete i fysik

Christoffer Lundell
2022/06/01

Handledare: Christer Svensson

Abstract

This report presents the results from, as well as the information around the creation, usage, and contents of an interactive simulation program that has been designed with the help of Godot. The main purpose is to show the relationship between physics formulas and objects in motion. The program will aid in the explanations of physical phenomena by giving graphical examples on common problems in physics at an upper secondary school level. The source code for the project, as well as the program, is available at the bottom of this document.

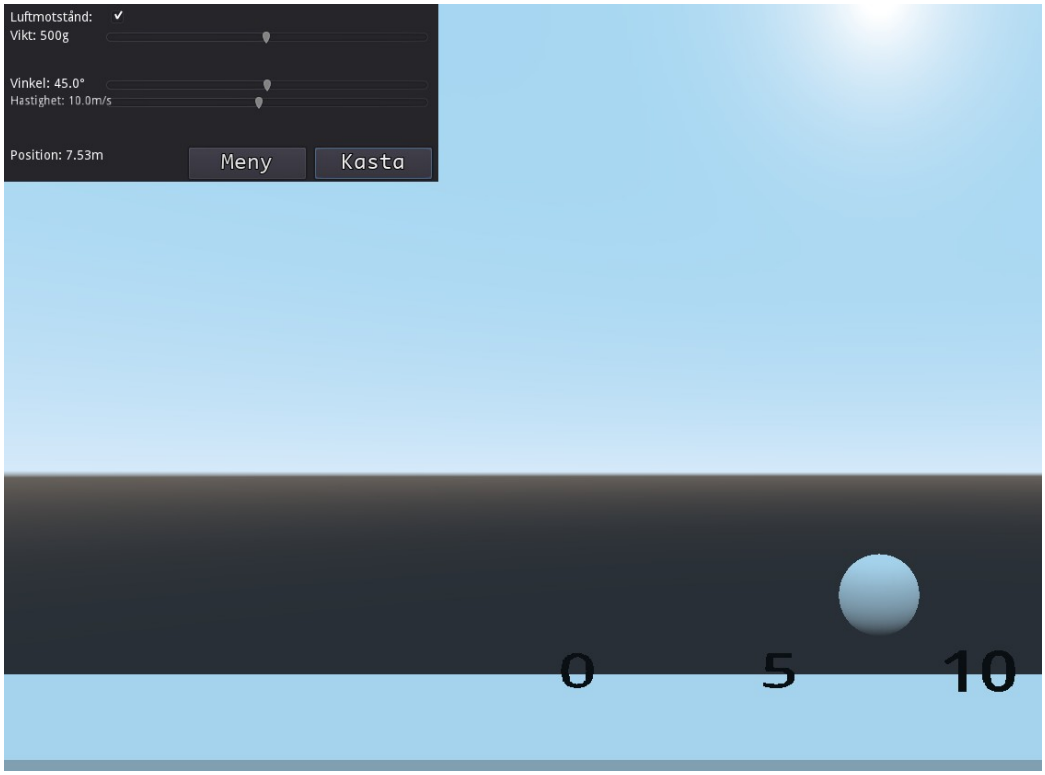
Sammanfattning

Den här rapporten presenterar resultaten från, såväl som informationen kring skapandet, användandet och innehållet av ett interaktivt simuleringsprogram som har utformats med hjälp av Godot. Det huvudsakliga syftet är att visa sambandet mellan fysiska formler och objekt i rörelse. Programmet kommer underlätta i förklaringar på fysiska fenomen genom att ge grafiska exempel på vanliga problem i fysik på gymnasienivå. Källkoden till projektet, såväl som programmet, finns tillgängligt längst ner på detta dokument.

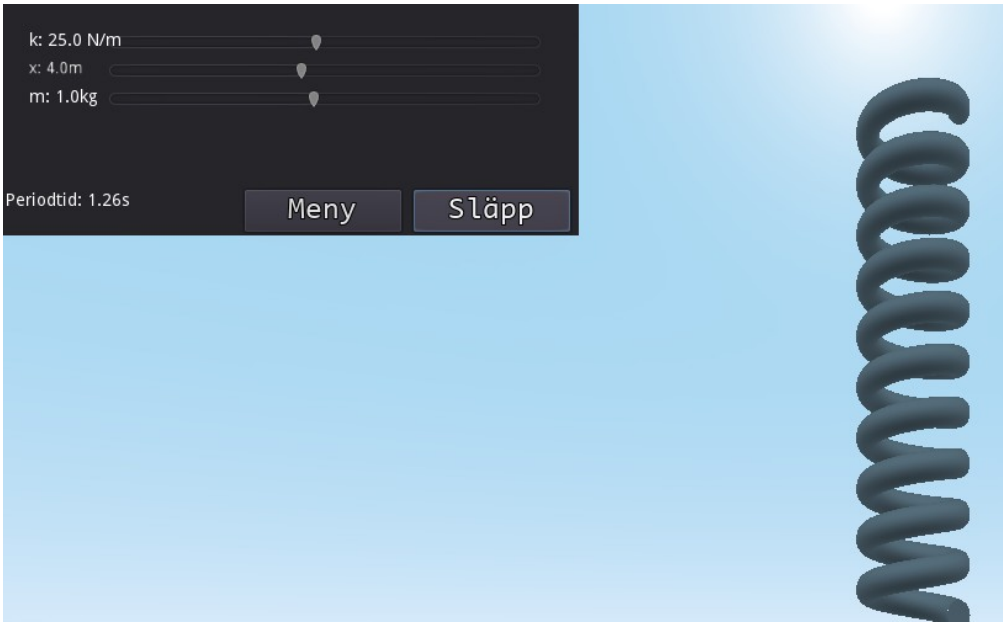
Innehållsförteckning

1. Inledning

Programmet som har skapats till detta projekt utför två olika simulationer, den ena simulerar ett snett kast där det går att ändra olika parametrar som hastighet, vinkel, luftmotstånd och vikt.



Den andra simulerar en fjäderrörelse, alltså hur en fjäders resonans får den att dra ihop sig och förlänga sig över tid. Där lämpliga parametrar som fjäderkonstant, utsträckning och massa går att ändra.



1.1. Frågeställning

Till det här projektet vill jag undersöka kopplingen mellan det teoretiska och det praktiska i fysik närmre. Går det att programmera ett visuellt instrument för att lättare påvisa kopplingen mellan fysikteori och fysiska fenomen med ett grafiskt hjälpmedel?

1.2. Bakgrund

Samband mellan fysiska formler och visuella hjälpmedel har alltid varit viktigt för förståelsen av olika områden i fysik samt för undervisningen av dem. Bara genom att ha en bra förståelse mellan det teoretiska och det praktiska kan vi göra större framsteg inom tillämpad fysik.

1.3. Syfte

Målet med det här arbetet är att använda olika grundläggande verktyg och fysikteori för att sammanställa ett fungerande program som utför olika fysiska simulationer, att testa dessa simulationer så att de stämmer så bra som möjligt.

2. Metod

Godot är en spelmotor där program skapas som är formade med hjälp av byggstenar som kallas objekt. Dessa objekt kopplas samman med hjälp av kod i programmeringsspråket GDScript. I simulationerna i Godot körs koden som är relevant för denna rapport kontinuerligt varje tidsintervall, vilket gör det lämpligt att använda stegmetoder som tar hänsyn till äldre värden under simulationens gång för att räkna ut de nya värdena.

Ett snett kast utan luftmotstånd fungerar så att ett föremål kastas, i det här fallet från markhöjd, med en vinkel som får föremålet att röra sig delvis uppåt. Gravitationen motverkar sedan denna uppåtrörelse och får bollen att röra sig längs en parabel. För att räkna rörelsen för snett kast behöver vi veta hur vinkeln påverkar rörelsen i x-led, och rörelsen i y-led. Här används cos respektive sin för detta ändamål. Se 3.1.

Ett snett kast med luftmotstånd fungerar på liknande sätt som ett snett kast utan luftmotstånd, men föremålet påverkas inte bara av en förändring i hastighet längs y-ledet. Föremålets acceleration i både x-, och y-led beror istället på dess hastighet och luftmotståndskoefficient, som utgör moståndet för en specifik form, och blir det följande.

$a_x = -\frac{k \cdot v}{m} \cdot v_x$ och $a_y = (-g) - \frac{k \cdot v}{m} \cdot v_y$. Se 3.1.

Fjäderrörelsen fungerar så att den ena änden på fjädern fästs fast vid en punkt och en utsträckning på fjädern, eller en annan kraft som påverkar fjädern, får den att sträcka ut sig och dra ihop sig i något som kallas en resonansrörelse. För att räkna ut denna rörelse används ”Hookes lag” som lyder $F = k \cdot x$ vilket betyder att kraften från fjädern är lika med fjäderkonstanten (k , som utgör fjäderns elastiska egenskap) gånger utsträckningen på fjädern från jämviktsläget (x). Se 3.2.

3. Teori

3.1. Snett kast

Uträkningen av sneda kast är uppbyggd av stegmetoden nedan. Utan luftmotstånd ändras hastigheten enligt följande formel, med ett varierande tidsintervall Δu . Lägg märke till att variablernas index är sekventiella, vilket kan vara missvisande eftersom tiden mellan dem *inte* är konstant.

$v_{x0}=v_0\cdot\cos(\alpha)$ och $v_{y0}=v_0\cdot\sin(\alpha)$

$v_{y1}=v_0+\Delta u\cdot(-g)$
$v_{y2}=v_0+\Delta u\cdot(-g)$
…
$v_{y(n)}=v_{y(n-1)}+\Delta u\cdot(-g)$

Med luftmotstånd ändras hastigheten/accelerationen enligt följande, med tidsintervallet Δu som också varierar.

$(t_0=0)$

$v_{x0}=v_0\cdot\cos(\alpha)$ $\quad$ $v_{y0}=v_0\cdot\sin(\alpha)$

$a_{x0}=\frac{-(k\cdot v_0)}{m}\cdot v_{x0}$ $\quad$ $a_{y0}=(-g)-\frac{k\cdot v_0}{m}\cdot v_{y0}$

$(t_1=t_0+\Delta u_0)$

$v_{x1}=v_{x0}+a_{x0}\cdot\Delta u$ $\quad$ $v_{y1}=v_{y0}+a_{y0}\cdot\Delta u$

$v_1=\sqrt{v_{x1}^2\cdot v_{y1}^2}$

$a_{x1}=\frac{-(k\cdot v_1)}{m}\cdot v_{x1}$ $\quad$ $a_{y1}=(-g)-\frac{k\cdot v_1}{m}\cdot v_{y1}$

…

$(t_n=t_{n-1}+\Delta u_{n-1})$

$v_{x(n)}=v_{x(n-1)}+a_{x(n-1)}\cdot\Delta u$ $\quad$ $v_{y(n)}=v_{y(n-1)}+a_{y(n-1)}\cdot\Delta u$

$v_n=\sqrt{v_{x(n)}+v_{y(n)}}$

$a_{x(n)}=\frac{-(k\cdot v_n)}{m}\cdot v_{x(n)}$ $\quad$ $a_{y(n)}=(-g)-\frac{k\cdot v_n}{m}\cdot v_{y(n)}$

där k är den följande konstanten.

$k=½\cdot C\cdot\rho\cdot A$ varav

C utgör luftmotståndskoefficienten
ρ utgör luftens densitet
A är föremålets tvärsnittsarea

Vid varje steg ändras positionen med

$x_0=0$ $\quad$ $y_0=0$
$x_n=x_{n-1}+v_{x(n-1)}\cdot\Delta u$
$y_n=y_{n-1}+v_{y(n-1)}\cdot\Delta u$

Motsvarigheten i funktionen som körs i varje tidsintervall blir då det följande.
Utan luftmotstånd:

```
func _physics_process(delta):
    var a_y = -g
    velocity.y += a_y * delta
    translation += velocity * delta
```

Där `translation` benämner objektets position och `+=` betyder att värdet till höger läggs till i variabeln till vänster.

Med luftmotstånd:

C för en sfär är en konstant $0{,}47$.

ρ har valts till att vara konstanten $\frac{1{,}2\,kg}{m^3}$.

Tvärsnittsarean på bollen är $A = r^2 \cdot \pi = 0{,}5^2 \cdot \pi \approx 0{,}785\,m$.

Eftersom radien på bollen är 0,5m blir det följande motsvarigheten i funktionen som körs varje tidsintervall så här.

```
func _physics_process(delta):
    var k = 0.5 * 0.47 * 1.2 * 0.785 * pi
    var v = sqrt(pow(velocity.x, 2) + pow(velocity.y, 2))
    var m = $"Knappar/ViktGlidare".value
    var a_x = -(k * v) / m * velocity.x
    var a_y = -g - (k * v) / m * velocity.y
    velocity.x += a_x * delta
    velocity.y += a_y * delta
    translation += velocity * delta
```

3.2. Fjäderrörelse

Fjäderrörelsen är uppbyggd på Hookes lag, som lyder $F=-k\cdot x$, där F är sträckkraften, k är fjäderkonstanten och x är fjäderns förlängning. Totalkraften är sträckkraften plus gravitationen på fjädern och ändras enligt följande formel, med ett varierande tidsintervall Δu . Den generella formeln blir då på följande sätt. Lägg märke till att variablernas index är sekventiella, vilket kan vara missvisande eftersom tiden mellan dem *inte* är konstant.

$(t_0=0)$
$x_0=0$
$v_0=0$

$(t_1=t_0+\Delta u_0)$
$F_1=-k\cdot x_0+g\cdot m$
$a_1=\frac{F_1}{m}$
$v_1=v_0+a_1\cdot\Delta u_0$
$x_1=x_0+v_1\cdot\Delta u_0$

$(t_2=t_1+\Delta u_1)$
$F_2=-k\cdot x_1+g\cdot m$
$a_2=\frac{F_2}{m}$
$v_2=v_1+a_2\cdot\Delta u_1$
$x_2=x_1+v_2\cdot\Delta u_1$

…

$(t_n=t_{n-1}+\Delta u_{n-1})$
$F_n=-k\cdot x_{n-1}+g\cdot m$
$a_n=\frac{F_n}{m}$
$v_n=v_{n-1}+a_n\cdot\Delta u_{n-1}$
$x_n=x_{n-1}+v_n\cdot\Delta u_{n-1}$

Där k är fjäderkonstanten med enheten $\frac{N}{m}$, x är utsträckningslängden på fjädern ifrån dess jämviktsläge och m är dess konstanta massa. Minustecknet framför definitionen av F beror på att kraften från fjädern alltid är i motsatt riktning till dess utsträckning/ihopdragning från jämviktsläget.

I GDScript blir detta det motsvariga.

```
func _physics_process(delta):
    var force = -k * (x-equilibrium) + g * m
    acceleration = force / m
    velocity += acceleration * delta
    x += velocity * delta
```

Perioden på fjädern blir $T = 2\pi\sqrt{\frac{m}{k}}$. Detta även efter att gravitationen läggs till i ekvationen.

Anledningen till detta är att det enda gravitationen gör är att ändra jämviktsläget för fjäderns svängningar, eftersom kraften på fjädern måste då uppnå g i motsatta riktning för att vända håll uppåt, och g mindre för att vända nedåt. Därför beror perioden bara på m och k .

4. Material

4.1. Godot

För arbetet behövdes en grafikmotor där själva fysiken gick att kontrollera själv. Godot var en bra kandidat eftersom den är en färdig spelmotor där det går att utesluta den inbyggda bearbetningen av fysik. Detta går genom att använda vissa typer av objekt i Godot där simulationen av fysik är avstängd. Syftet med detta är för att sedan kunna implementera fysikuträkningarna manuellt.

4.2. Blender

För vissa simulationer behövdes det tillgång till 3d-modeller som är mer komplexa än vanliga geometriska figurer. Till detta användes Blender för att utforma dessa modeller.

5. Resultat

Den slutgiltiga produkten är ett minimalistiskt exempel på hur fysiska fenomen utspelas och kontrolleras med hjälp av kod och fysikformler som gör det lätt att koppla ihop dessa fenomen med en mer abstrakt fysikteori.

Programmets simulationer har några få brister, huvudsakligen i simulationen för snett kast där det ofta förekommer en felmarginal på upp till 0,15m, detta kan bero på flera olika saker. Bland annat kan det bero på att i Godot körs den användardefinierade `_physics_process()` funktionen i varje objekt enligt intervallen $T = \frac{1}{bilder/sekund}$. Det betyder att när bildhastigheten ändras blir uträkningen odeterministisk eftersom positionen bara kan kontrolleras var Δt sekund. En annan anledning kan vara att skillnaden ser större ut än vad den är på grund av att det reella talet som visas i uträkningsrutan avrundas till att bara visa två decimaltal. Det kan få felmarginalen att verka större än den egentligen är i det fallet att de två talen avrundas åt olika håll. Det vill säga att förutom vid extremt onormala bildhastigheter kommer bollen i simulationen för ”Snett kast” (utan luftmotstånd) alltid att landa nära den teoretiska längden.

I simulationen för fjäderrörelse så är perioden för fjädern väldigt nära den förväntade tidsperioden.

6. Diskussion och slutsats

Jag tycker att den slutgiltiga produkten blev bra, även fast att båda simulationerna har sina brister. En brist är på grund av de felmarginaler som uppstår i simulationen för snett kast. En annan brist i programmet uppstår eftersom alla variabler i fjäderrörelsens uträkning egentligen bara beror på fjäderns utsträckning. Det betyder det att denna resonans kommer att pågå i evighet. Detta är ganska olikt fjäderrörelse i verkliga livet, men är bra nog för att ge en ungefärlig bild över hur det fungerar.

Slutligen kom jag fram till att ett minimalistiskt program som fungerar som en koppling mellan fysikteori och visuellt framförbara fysikfenomen är uppnåbart. Jag anser att felkällorna är helt undvikbara till exempel genom val av annan spelmotor, eller även vid bättre val av design. Jag anser även att dessa brister är försumbara eftersom en perfekt simulation inte är programmets huvudsakliga syfte.

7. Källförteckning

https://docs.godotengine.org/en/stable/

https://sv.wikipedia.org/wiki/Luftmotstånd

Formelblad

8. Länkar

Källkoden till programmet finns tillgänglig på sidan https://github.com/HoppenR/KA2 och är till för att importeras till Godot. Och finns även tillgänglig för att läsa källkoden i mappen ”scripts”.

Programmet för Windows finns tillgänglig på sidan https://github.com/HoppenR/KA2/releases/tag/v1.1.0 döpt KAv1.1.zip. Zip-filen behöver extraheras till en ny mapp innan programmet körs.