

KAUNAS UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATICS

P170B118 Physically Based Animation

Individual project 2

|  |
| --- |
| IFE-8, Martynas Kemežys |
| Date: 2021.11.08 |

Kaunas, 2021

Contents

[**1.** **Problem** 2](#_Toc87306848)

[**2.** **The solution** 3](#_Toc87306849)

[**3.** **Animations** 4](#_Toc87306850)

[**4.** **Displacements** 4](#_Toc87306851)

[**5.** **Programming code** 5](#_Toc87306852)

# **Problem**

Chosen problem - **7**

* From the right discs are falling
* The discs are solid bodies
* The discs roll down
* There contact is partially plastic
* Take friction into account for contact between discs and ground

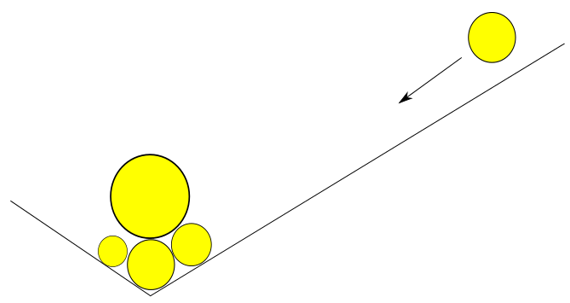


Figure 1. Problem illustration

1. **The solution**

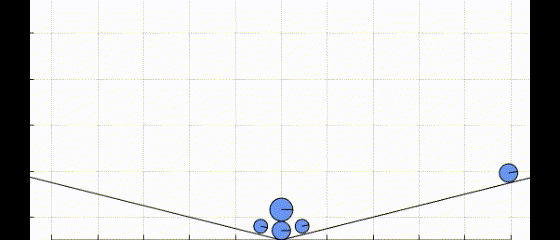


Figure 2. Solved problem illustration

Mechanical contact is defined using penalty method. It is assumed that particles interpenetrate into each other as far as stiffness and damping forces of contact allow. Stiffness and damping forces generated by penetration reduce motion of particles and push them away from each other

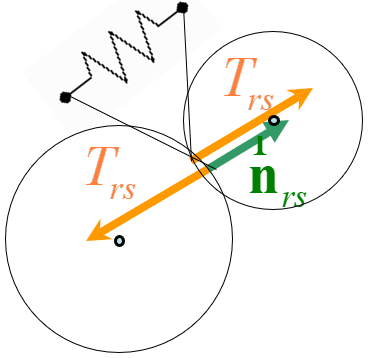


Figure 3. Penalty method illustration

# **Animations**

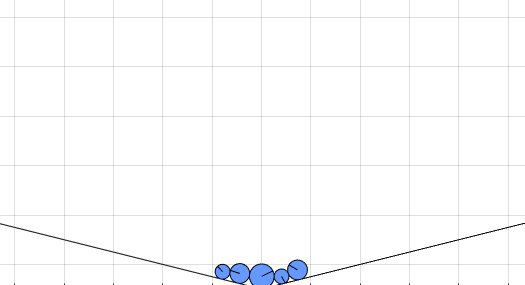
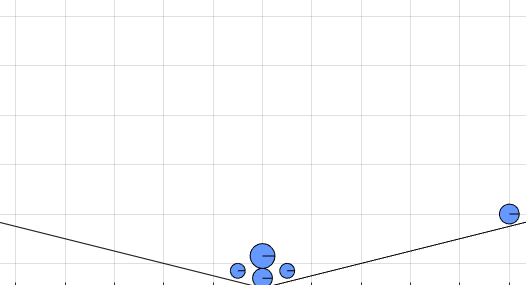


Figure 4. Before animation Figure 5. After animation

# **Displacements**

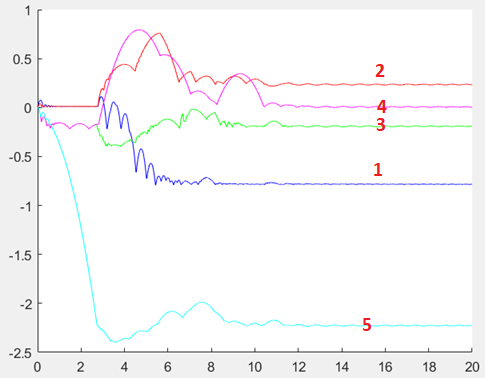


Figure 6. Discs displacements

As we can see from graph **Figure 6** that most displacements had fifth disc because it started roll down from right upper corner, second and fourth discs had also large displacement because they were in the end of impact.

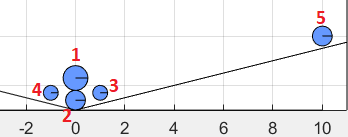


Figure 7. Numbered discs

# **Programming code**

function disc\_contact

close all; clc;

llsk=3;

mass=[5, 2, 1, 1,2]; % masses

iner = [0.05, 0.05, 0.05, 0.05, 0.05] \* 0.05; % inertia

rad=[0.5, 0.4, 0.3, 0.3, 0.4]; % radius

cor=[ 0 -1.7 0; 0 -2.6 0; 1 -2.3 0; -1 -2.3 0; 10 0 0]; % (x,y,angle)

nmz=length(mass); % total nodes

NN=nmz\*llsk; % degree of freedom

g=9.81; % gravity

F=zeros(1,nmz\*llsk);F(2:llsk:end)=-mass\*g; % fixed forces

% penalty coef

stifp\_n=[50000 50000 50000 50000 50000];

dampp\_n=[10 10 10 10 10]\*5;

dampp\_t=[1 1 1 1 1]\*5;

fric=[0.3 0.3 0.3 0.3 0.3];

% planes

xmin=-11;xmax=11, ymin=-3; ymax=9; % window

NRM=[0 1; 0 -1;1 0 ; -1 0; -0.5 2; 0.5 2]; % norlal vectors

PNT=[0 ymin; 0 ymax; xmin 0; xmax 0; 6 -1.5; -6 -1.5]; % points in lines

U=zeros(NN,1); DU=zeros(NN,1); % initial displacements and velocities

% preparing for rendering

figure(1); axis equal;axis ([xmin xmax ymin ymax]);grid on;hold on;

rendering(U,cor,rad, NRM,PNT);

pause

% integration data

TT=20; dt=0.001;

nsteps=TT/dt;

t=0;

Urez=zeros(NN,1); % array for results

for i=1:nsteps

% updating acceleration velocities and displacements

DDU=acceleration(U,DU,t,mass,stifp\_n,dampp\_n,dampp\_t, fric,F,cor,rad, iner,NRM,PNT);

DU=DU+dt\*DDU';

U=U+dt\*DU;

Urez(:,i+1)=U;

% rendering

if(mod(i,10) ==0),

cla; hold on

rendering(U,cor,rad, NRM,PNT);

hold off

pause(0.01);

end

t=t+dt;

end

figure(2);hold on;

plot([0:dt:TT],Urez(2,:),'-b');

plot([0:dt:TT],Urez(5,:),'-r');

plot([0:dt:TT],Urez(8,:),'-g');

plot([0:dt:TT],Urez(11,:),'-m');

return

end

function DDU=acceleration(U,DU,t,mass,stifp\_n,...

dampp\_n,dampp\_t,fric,F,cor,rad, iner, NRM,PNT);

llsk=3;

nmz=length(mass);NN=nmz\*llsk;

T=F;

for i=1:nmz % colisions with planes

r=[(i-1)\*llsk+1:i\*llsk]; du=DU(r); c=cor(i,:)'+U(r);

nconstr=size(NRM,1);

for j=1:nconstr

n=-NRM(j,:);n=n/norm(n); tau=[n(2), -n(1)]; % normal ant tangential vectors

A=PNT(j,:); % point in plane

dlt=dot(c(1:2)'-A,n)+rad(i);

if dlt > 0,

rN= dlt\*stifp\_n(i)+dot(du(1:2),n)\*dampp\_n(i); if rN<0, rN=0; end

rT=(dot(du(1:2),tau)-du(3)\*rad(i))\*dampp\_t(i); if abs(rT)>fric(i)\*abs(rN), rT=sign(rT)\*fric(i)\*rN; end

T(r)=T(r)+[-rN\*n-rT\*tau,rT\*rad(i)];

end

end

for j=i+1:nmz % check collision between bodies

s=[(j-1)\*llsk+1:j\*llsk]; duj=DU(s); cj=cor(j,:)'+U(s);

n=(cj(1:2)-c(1:2))/norm(cj(1:2)-c(1:2)); tau=[n(2);-n(1)]; % normal ant tangential vectors

dlt=dot(c(1:2)-cj(1:2),n)+rad(i)+rad(j);

if dlt > 0 % if contact

stifpn=min(stifp\_n(i),stifp\_n(j));

damppn=min(dampp\_n(i),dampp\_n(j)); damppt=min(dampp\_t(i),dampp\_t(j));

frc=min(fric(i),fric(j));

rN= dlt\*stifpn+dot(du(1:2),n)\*damppn; if rN<0, rN=0; end

rT=(dot(du(1:2)-duj(1:2),tau)-(du(3)\*rad(i)-duj(3)\*rad(j)))\*damppt; if abs(rT)>frc\*abs(rN), rT=sign(rT)\*frc\*rN; end

T(r)=T(r)+[-rN\*n-rT\*tau; rT\*rad(i)]';

T(s)=T(s)+[ rN\*n+rT\*tau;-rT\*rad(j)]';

end

end

end

DDU(1:llsk:NN)=T(1:llsk:end)./mass; % forces divides from masses

DDU(2:llsk:NN)=T(2:llsk:end)./mass; % forces divides from masses

DDU(3:llsk:NN)=T(3:llsk:end)./iner; % forces divides from inertia

return

end

function rendering(U,cor,rad, NRM,PNT)

nmz=length(rad);llsk=3;

for i=1:nmz

% ploting particles

r=[(i-1)\*llsk+1:i\*llsk];

u=U(r)+cor(i,:)';

spind=rad(i);

rectangle('Position',[u(1)-spind,u(2)-spind,2\*spind,2\*spind],'Curvature',[1,1],'FaceColor',[0.4 0.6 1]);

% ploting line of particle

plot([u(1),u(1)+spind\*cos(u(3))], [u(2),u(2)+spind\*sin(u(3))],'k-');

end

% ploting lines of window

for i=1:size(NRM,1)

plot([PNT(i,1)-10\*NRM(i,2),PNT(i,1)+10\*NRM(i,2)],[PNT(i,2)+10\*NRM(i,1),PNT(i,2)-10\*NRM(i,1)],'k-')

end

return

end