

**Laporan *Project* Akhir Perencanaan dan Pembuatan  
Simulasi Perbandingan Dua Pegas Mata Kuliah  
Pemrograman Visual**

*UNTUK MEMENUHI TUGAS AKHIR MATA KULIAH PEMROGRAMAN VISUAL*



Disusun oleh:

Ahmad Izza Zain Firdaus      19051214063

Amara Indah Putri              19051214076

**PRODI SISTEM INFORMASI  
JURUSAN TEKNIK INFORMATIKA  
FAKULTAS TEKNIK  
UNIVERSITAS NEGERI SURABAYA**

**2021**

## DAFTAR ISI

DAFTAR ISI.....	i
1. Deskripsi dan penggunaan program.....	1
1.1 Deskripsi Program .....	1
1.2 Manfaat Program .....	1
2. Studi Pustaka.....	2
2.1 Pegas.....	2
2.2 Rumus pada Pegas .....	2
2.3 Vpython .....	3
3. Perencanaan Kebutuhan .....	4
3.1 Kebutuhan Hardware dan Software.....	4
3.2 Perencanaan Desain.....	4
3.3 Object Vpython .....	4
4. Implementasi .....	6
4.1 Analisis Kebutuhan .....	6
4.1.1 Analisis Kebutuhan Pengguna .....	6
4.1.2 Analisis Kebutuhan Hardware .....	6
4.1.3 Analisis Kebutuhan Software .....	6
4.2 Langkah Pembuatan Simulasi Pegas .....	6
4.2.1 Pembuatan Kode .....	6
4.2.2 Program yang dihasilkan.....	9
5. Kesimpulan .....	12
Daftar Pustaka.....	13
Lampiran .....	14

# **1. Deskripsi dan penggunaan program**

## **1.1 Deskripsi Program**

Simulasi pegas adalah program yang dibuat untuk menampilkan kondisi pegas sesuai dengan masukan yang diinginkan. Program ini merupakan sebuah animasi yang akan menampilkan perbandingan dua buah pegas dengan parameter yang dapat diubah yaitu massa pegas serta parameter tetap yaitu koefisien pegas dan perubahan panjang pegas. Dua buah pegas ini merupakan berbeda jenis yaitu, sebuah pegas ganda dan sebuah pegas tunggal. Program ini agar dapat dijalankan dengan menggunakan *library vpython* dan *tkinter*.

## **1.2 Manfaat Program**

Pemanfaatan program dapat dimanfaatkan sebagai sarana pembelajaran dan menampilkan simulasi kinerja dari sebuah pegas berdasarkan massa dan koefisien pegas yang berbeda-beda.

## 2. Studi Pustaka

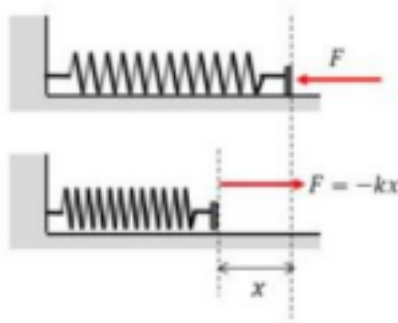
### 2.1 Pegas

Pegas adalah sebuah komponen yang memiliki sifat elastis, dimana memiliki kecenderungan untuk Kembali kedalam bentuk sempurna setelah dilakukan sebuah gaya seperti tekanan, tarikan atau dorongan. Sebuah pegas jika ditarik akan mengalami pertambahan Panjang, dan apabila dilepaskan maka cenderung Kembali ke bentuk semula

### 2.2 Rumus pada Pegas

#### Hukum Hooke

Hukum hooke menyatakan:



$$F = k \cdot x$$

Keterangan:

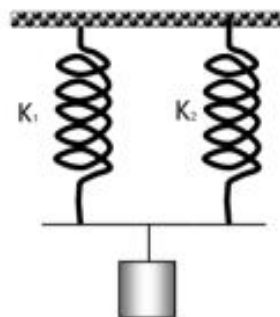
$F$  : Gaya yang bekerja (N)

$k$  : konstanta pegas

$x$  : Perubahan panjang pegas

#### Susunan Pegas Paralel

Jika beberapa pegas disusun parallel, maka panjang pegas akan tetap sama dengan panjang pegas semula, namun luas penampangnya menjadi lebih besar. Dengan rumus, sebagai berikut :



$$K_p = 2K$$

Keterangan :

$K_p$  : persamaan pegas susunan

$k$  : konstanta pegas (N/m)

Sedangkan persamaan  $n$  untuk pegas yang disusun parallel yaitu :

$$K_p = n \cdot K$$

Dimana  $n$  adalah jumlah pegas.

### 2.3 Vpython

*VPython* adalah bahasa pemrograman *Python plus* modul grafis 3D yang disebut visual. *VPython* memungkinkan pengguna untuk membuat objek seperti bola dan kerucut di ruang 3D dan menampilkan objek ini di jendela. Ini membuatnya mudah untuk membuat visualisasi sederhana, memungkinkan pemrogram untuk lebih fokus pada aspek komputasi program mereka. Kesederhanaan *VPython* telah menjadikannya alat untuk ilustrasi fisika sederhana, terutama di lingkungan pendidikan.

### 3. Perencanaan Kebutuhan

#### 3.1 Kebutuhan *Hardware* dan *Software*

Alat penelitian yang digunakan dalam penyusunan project akhir dibagi dua, yaitu *hardware* (perangkat keras) dan *software* (perangkat lunak). Alat yang digunakan sebagai berikut :

##### *Hardware*

Seperangkat komputer dengan spesifikasi

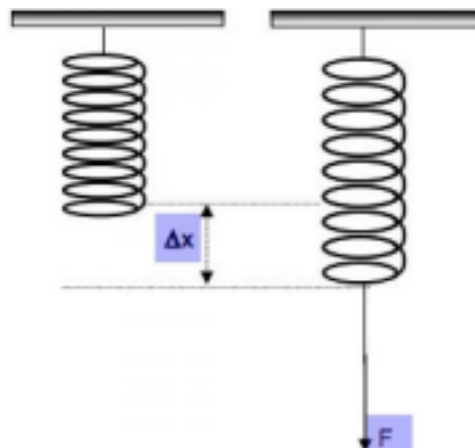
- *Processor* : AMD A4-9125
- *RAM* : 4096 MB
- *ROM* : 1000 GB HDD
- *VGA* : AMD Radeon R3 Graphics

Digunakan untuk pembuatan, penyusunan, penataan, dan uji coba *code* simulasi pegas.

##### *Software*

Software yang digunakan dalam pembuatan, penyusunan dan uji coba *code* pada simulasi pegas adalah *Spyder* yang merupakan editor kode untuk *VPython* dan *tkinter*.

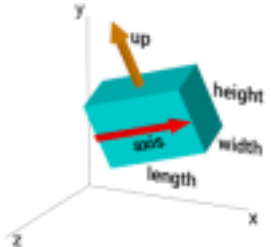

#### 3.2 Perencanaan Desain



Diambil dari rumushitung.com

#### 3.3 Object Vpython

<i>Object</i>	Penggunaan	<i>Image</i>
<i>Box()</i>	Penyangga atas	

<i>Box()</i>	Beban	
<i>Helix()</i>	Gambaran pegas	

## 4. Implementasi

### 4.1 Analisis Kebutuhan

#### 4.1.1 Analisis Kebutuhan Pengguna

Program ini dirancang dan diterapkan sebagai sarana pembelajaran dan menampilkan simulasi kinerja dari sebuah pegas berdasarkan massa dan koefisien pegas yang berbeda-beda. Dalam pemanfaatannya diharapkan dapat mempermudah proses memahami sebagai sarana pembelajaran dengan baik.

#### 4.1.2 Analisis Kebutuhan *Hardware*

Dalam pembuatan *project* akhir simulasi pegas ini dibuat dan dijalankan di *personal computer* (laptop) berbasis Windows. Spesifikasi hardware minimal untuk menjalankan *project* akhir simulasi pegas adalah sebagai berikut :

- *Processor* : AMD A4-9125
- *RAM* : 4096 MB
- *ROM* : 1000 GB HDD
- *VGA* : AMD Radeon R3 Graphics

#### 4.1.3 Analisis Kebutuhan Software

- a. *Microsoft Windows 10*
- b. *Spyder* atau editor kode *VPython* dan *tkinter* yang lain

### 4.2 Langkah Pembuatan Simulasi Pegas

#### 4.2.1 Pembuatan Kode

Pada pembuatan kode simulasi pegas ini penulis menggunakan editor kode Spyder. Dalam coding ini terdapat fungsi yang digunakan untuk membuat program simulasi pegas. Oleh karena itu, penulis akan menjelaskan kode-kode yang ada, sebagai berikut :

- a. Kode Program Input Masa Pegas

```
print("=====SIMULASI PEGAS=====")
print(".....\n")
aa=input("Masa Pegas: ")
#Atur massa balok
mblock =float(aa)
single_mblock=float(aa)
#mblock=0.1
#single_mblock=0.1

#Atur konstanta pegas
ks = 15
ks = 2*ks
single_ks=15

#panjang awal pegas dan posisi awal pegas
L0 = 0.3
Lstarty = -2.0
```

Kode diatas merupakan kode yang jika di run akan menampilkan peng-input-an



massa pegas. Tipe data yang digunakan adalah float dimana bisa memasukkan angka pecahan pada tipe data ini.

b. Kode Program Daftar Planet (Percabangan)

```
#daftar gravitasi planet
planet=["mercurius", "venus", "bumi", "mars", "Jupiter", "Saturnus", "Uranus", "Neptunus", "Bulan"]
planet_grav=[3.7, 8.87, 9.8, 3.721, 24.79, 10.44, 8.87, 11.15, 1.62]

print("Nama Planet Gravitasi \n")
for i in range(9):
    print(i+1, ". %-15s %5.3f" % (planet[i], planet_grav[i]))
inp_planet=input("Pilih gravitasi yang digunakan: ")
inp_planet=int(inp_planet)
if inp_planet==1:
    g=planet_grav[0]
elif inp_planet==2:
    g=planet_grav[1]
elif inp_planet==3:
    g=planet_grav[2]
elif inp_planet==4:
    g=planet_grav[3]
elif inp_planet==5:
    g=planet_grav[4]
elif inp_planet==6:
    g=planet_grav[5]
elif inp_planet==7:
    g=planet_grav[6]
elif inp_planet==8:
    g=planet_grav[7]
elif inp_planet==9:
    g=planet_grav[8]
else:
    print("Pilih sesuai pilihan, Gravitasi yang digunakan Bumi")
    g=planet_grav[2]
#deklarasi kebutuhan meliputi: gravitasi, perubahan tiap waktu (deltat)
#g = planet_grav[3]
```

Kode diatas merupakan kode yang jika di *run* akan menampilkan daftar planet-planet beserta besar massa gravitasi yang nantinya akan dipilih. Percabangan dituliskan dengan kondisi “*if*” digunakan untuk mengeksekusi kode jika kondisi bernilai benar. Kondisi “*elif*” merupakan lanjutan dari kondisi “*if*”, dengan ini bisa membuat kode program yang akan menyeleksi beberapa kemungkinan yang bisa terjadi.

c. Kode Program Waktu

```
#bantuan looping
deltat = 0.01
t = 0
```

Kode program ini berisikan nilai waktu yang nantinya akan membantu *looping* atau perulangan pada simulasi pegas ketika di run.

d. Kode Program Visual

```
#penggambaran object
#Pegas ganda
ceiling = box(pos=vec(0,0,0), length = 5, height = 0.01, width = 0.2)
block = box(pos = vector(0,lstarty,0),size = vec(1,0.1,0.1), color = color.red)
spring = helix(pos= vec(-0.3,0,0), axis = block.pos, radius = .03, thickness = 0.008, coils = 30, color = color.yellow)
spring2 = helix(pos= vec(0.3,0,0), axis = block.pos, radius = .03, thickness = 0.008, coils = 30, color = color.yellow)
#Pegas Tunggal
single_block = box(pos = vector(2,lstarty,0),size = vec(0.5,0.1,0.1), color = color.red)
min=single_block.pos-vec(2,0,0)
single_spring = helix(pos= vec(2,0,0), axis = min, radius = .03, thickness = 0.008, coils = 30, color = color.green)

#Menampilkan informasi dan data
print("Nama Planet      Gravitasi \n")
for i in range(4):
    print(planet[i], planet_grav[i])
```

Kode program ini merupakan kode program yang nanti akan menampilkan visual

dari simulasi pegas berupa tongkat, *box* dan juga spiral (*peer*).

e. Kode Program Menampilkan Informasi dan Data

```
#Menampilkan informasi dan data
print("Nama Planet      Gravitasi \n")
for i in range(4):
    print(planet[i], planet_grav[i])

print("Massa Block: ", mblock)
print("Konstanta Pegas: ", ks)
```

Kode program ini untuk menampilkan informasi dan data yang menggunakan fungsi “range” untuk menghasilkan list planet beserta besar nilai gravitasinya.

f. Kode Program Menghitung Momentum *Block* dan Gaya Gravitasi

```
#menghitung momentum block
pblock = mblock*vector(0,0,0)
psingle_block=single_mblock*vector(0,0,0)
#menghitung gaya gravitasi
Fgrav = mblock*vector(0,-g,0)
Fgrav2= mblock*vector(0,-g,0)
deltapanjang=mag(block.pos)-L0
```

Kode program ini untuk menghitung besar nilai momentum *block* dan gaya gravitasi dengan rumus yang telah diketikkan dalam kode program tersebut.

g. Kode Program Perulangan

```
while t < 10:
    rate(20) #jumlah loop yang dilakukn dalam 1 detik
    L = block.pos
    L2 = mag(L)
    Lhat = norm(L) #norm mengubah menjadi vektor satuan, vektornya panjangnya bernilai 1 satuan
    Lhat2=norm(L2)
    s=mag(L)-L0 #mag adalah nilai mutlak, pertambahan panjang, mag 1 artinya besaran dari vector 1 dengan rumus akar dari x kuadrat
    s2=mag(L2)-L0
    Fspring=-ks*s*Lhat #or Lhat = L/mag(L)
    Fsingle_spring=-single_ks*s2*Lhat2
    #enter the spring stretch
    #enter the spring force
    Fnet = Fspring + Fgrav #net force
    Fsingle_net=Fsingle_spring+Fgrav2
    #if t == 0:
    #    print(Fnet)

    pblock = pblock + Fnet*deltat #update momentum
    block.pos = block.pos + (pblock/mblock)*deltat #update position
    spring.axis=block.pos
    spring2.axis=block.pos

    psingle_block=psingle_block+Fsingle_net*deltat
    single_block.pos=single_block.pos+(psingle_block/single_mblock)*deltat
    min=single_block.pos-vec(2,0,0)
    single_spring.axis=min

    t = t + deltat

gaya=-ks*mblock*deltapanjang
```

Kode Program ini untuk menampilkan perulangan pada simulasi pegas yang telah run. Perulangan “while” akan dieksekusi statement berkali-kali selama kondisi benar.

h. Kode Program Informasi dalam *String*

```
info_massa= "massa: " + str(mblock)
info_koef="koefisien pegas: " +str(ks)
info_pertpanjang="Pertambahan Panjang: "+str(deltapanjang)
info_gaya="gaya pada pegas: " +str(gaya)
information=[info_massa, info_koef, info_pertpanjang, info_gaya]
```

Kode program ini untuk mengubah informasi yang berupa *object* ke bentuk *string*.

i. Kode Program Tipe *File*

```
tipeFile = [('Text file', '*.txt'), ('All files', '*')]
out=asksaveasfilename(filetypes=tipeFile)
# buka file untuk ditulis
file_bio = open(out, "w")
# tulis teks ke file
file_bio.write('\n').join(information))
# tutup file
file_bio.close()
```

Kode program ini berfungsi untuk membaca *file* yang berisi teks dan menggunakan mode “w” yaitu akses untuk menulis *file*, jika *file* sudah tersedia maka *file* akan di-*replace* dan diganti dengan yang baru diketikkan.

#### 4.2.2 Program yang dihasilkan

Hasil program yang telah di-run adalah sebagai berikut :

a. Memasukkan Massa Pegas

```
=====SIMULASI PEGAS=====
.....

Massa Pegas: 18
```

Disini kita akan memasukkan massa pegas

b. Memilih Gravitasi Planet

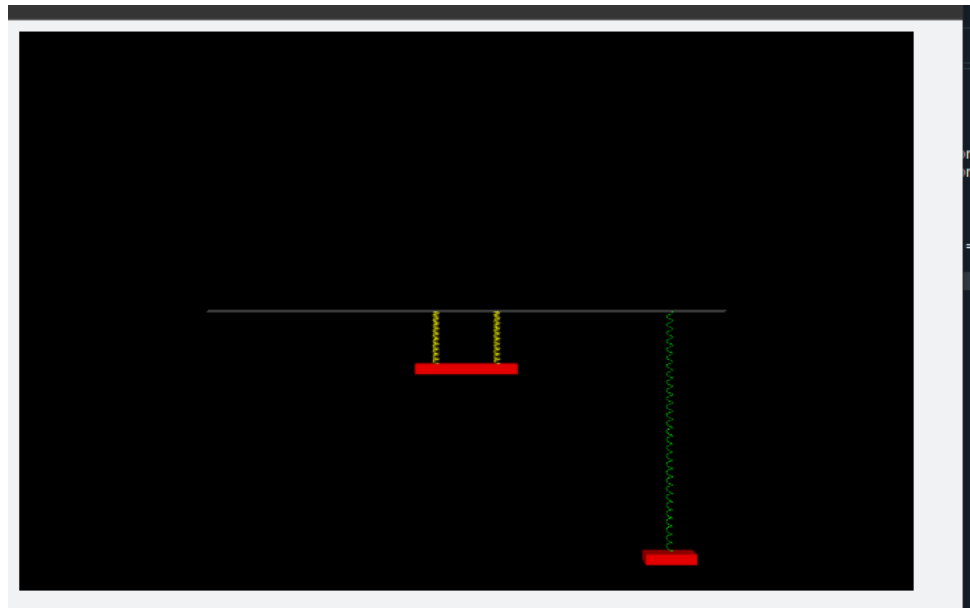
```
Nama Planet Gravitasi

1 . merkurius      3.700
2 . venus          8.870
3 . bumi           9.800
4 . mars           3.721
5 . Jupiter        24.790
6 . Saturnus       10.440
7 . Uranus         8.870
8 . Neptunus       11.150
9 . Bulan          1.620

Pilih gravitasi yang digunakan: 9
Nama Planet Gravitasi
```

Memilih salah satu gravitasi planet yang digunakan

c. Menampilkan Animasi Simulasi Pegas



Setelah massa pegas dan gravitasi planet yang dipilih dihitung oleh program. Lalu, program menampilkan animasi seperti gambar di atas.

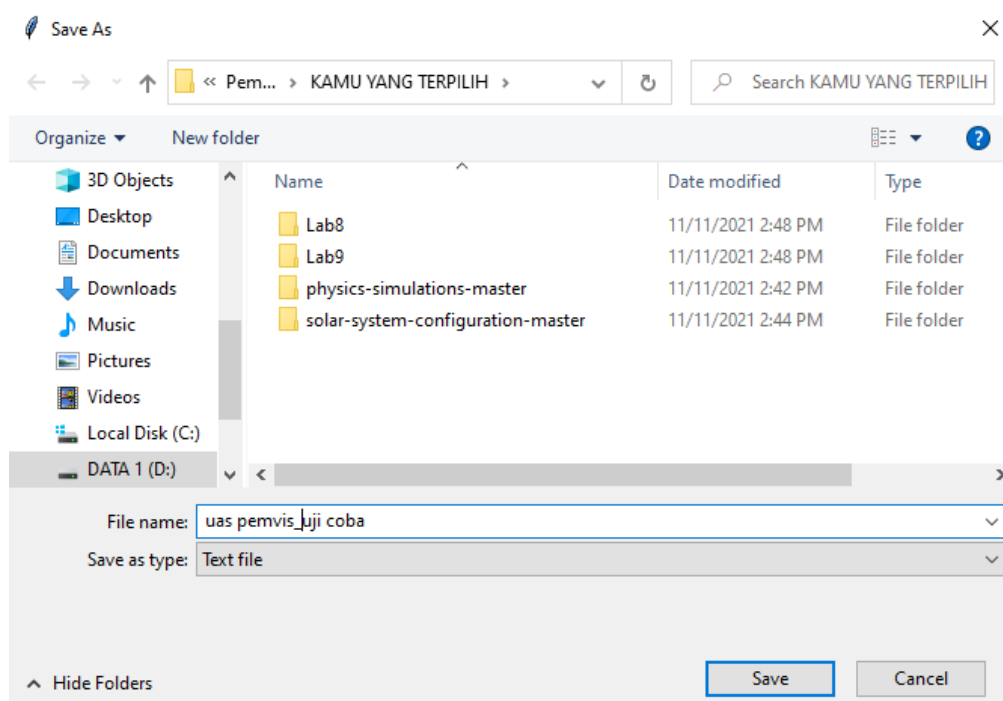
- d. Menampilkan Informasi dan data

```

Nama Planet      Gravitasi
mercurius 3.7
venus 8.87
bumi 9.8
mars 3.721
Massa Block: 18.0
Konstanta Pegas: 30
  
```

Program menampilkan informasi dan data yang telah di-input-kan.

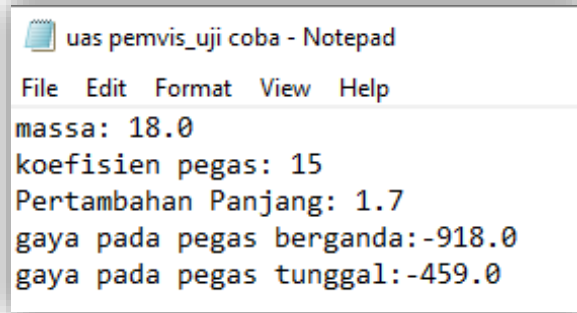
- e. Menyimpan perhitungan dalam tipe *file* “*txt*”



Setelah program animasi perbandingan dua pegas berhenti. Lalu, diarahkan

untuk menyimpan perhitungan dalam tipe *file* “*txt*”.

f. Hasil perhitungan



Dari dijalankannya uji coba pada program animasi perbandingan dua pegas dengan meng-input-kan massa dengan besar 18.0 yang kemudian dihitung bersama parameter tetap yaitu koefisien pegas dengan besar 15 dan pertambahan panjang 1.7 menghasilkan besar gaya pada pegas ganda -918.0 dan besar gaya pada pegas tunggal -459.0.

## 5. Kesimpulan

Berdasarkan pembahasan yang telah diuraikan dapat diambil kesimpulan bahwa *project* akhir simulasi pegas ini dapat dimanfaatkan sebagai sarana untuk mempermudah dalam memahami suatu pegas dengan nilai massa dan nilai koefisien yang berbeda dengan menampilkan simulasi pegas.

## Daftar Pustaka

Mahareni, Dina May, dkk. LAPORAN TUGAS AKHIR PEMROGRAMAN VISUAL PEMBUATAN GAME PEMBELAJARAN BERBASIS FLASH. Universitas Negeri Surabaya. 2015.

Utomo, Budi. LAPORAN PRAKTIKUM PEMROGRAMAN VISUAL *Microsoft Visual Basic 6.0*. Sekolah Tinggi Teknologi Duta Bangsa. 2012.

Muhardian, Ahmad. Fungsi range() di pemrograman python. 2021. Diakses pada petanikode.com pukul 9.05 wib, tanggal 17 Desember 2021.

# Lampiran

## 1. Source Code

```
from vpython import *
from tkinter.filedialog import askopenfilename, asksaveasfilename
print("=====SIMULASI PEGAS=====")
print(".....\n")
aa=input("Massa Pegas: ")
#Atur massa balok
mblock =float(aa)
single_mblock=float(aa)
#mblock=0.1
#single_mblock=0.1

#Atur Konstanta pegas
ks = 15
ks = 2*ks
single_ks=15

#panjang awal pegas dan posisi awal pegas
L0 = 0.3
Lstarty = -2.0

#daftar gravitasi planet
planet=["mercurius", "venus", "bumi", "mars","Jupiter", "Saturnus", "Uranus", "Neptunus", "Bulan"]
planet_grav=[3.7, 8.87, 9.8, 3.721, 24.79, 10.44, 8.87, 11.15, 1.62]

print("Nama Planet Gravitasi \n")
for i in range (9):
    print(i+1, ". %-15s %5.3f" % (planet[i], planet_grav[i]))
inp_planet=input("Pilih gravitasi yang digunakan: ")
inp_planet=int(inp_planet)
if inp_planet==1:
    g=planet_grav[0]
elif inp_planet==2:
    g=planet_grav[1]
elif inp_planet==3:
    g=planet_grav[2]
elif inp_planet==4:
    g=planet_grav[3]
elif inp_planet==5:
    g=planet_grav[4]
elif inp_planet==6:
    g=planet_grav[5]
elif inp_planet==7:
```



```

    g=planet_grav[6]
elif inp_planet==8:
    g=planet_grav[7]
elif inp_planet==9:
    g=planet_grav[8]
else:
    print("Pilih sesuai pilihan, Gravitasi yang digunakan Bumi")
    g=planet_grav[2]
#deklarasi kebutuhan meliputi: gravitasi, perubahan tiap waktu (deltat)
#g = planet_grav[3]

#bantuan looping
deltat = 0.01
t = 0

#penggambaran object
#Pegas ganda
ceiling = box(pos=vec(0,0,0), length = 5, height = 0.01, width = 0.2)
block = box(pos = vector(0,Lstarty,0),size = vec(1,0.1,0.1), color = color.red)
spring = helix(pos= vec(-0.3,0,0), axis = block.pos, radius = .03, thickness = 0.008, coils = 30, color =
color.yellow)
spring2 = helix(pos= vec(0.3,0,0), axis = block.pos, radius = .03, thickness = 0.008, coils = 30, color
= color.yellow)
#Pegas Tunggal
single_block = box(pos = vector(2,Lstarty,0),size = vec(0.5,0.1,0.1), color = color.red)
min=single_block.pos-vec(2,0,0)
single_spring = helix(pos= vec(2,0,0), axis = min, radius = .03, thickness = 0.008, coils = 30, color =
color.green)

#Menampilkan informasi dan data
print("Nama Planet      Gravitasi \n")
for i in range (4):
    print(planet[i], planet_grav[i])

print("Massa Block: ", mblock)
print("Konstanta Pegas: ", ks)

#menghitung momentum block
pblock = mblock*vector(0,0,0)
psingle_block=single_mblock*vector(0,0,0)
#menghitung gaya gravitasi
Fgrav = mblock*vector(0,-g,0)

```

```
Fgrav2= mblock*vector(0,-g,0)
deltapanjang=mag(block.pos)-L0
```

```
while t < 10:
```

```
    rate(20) #jumlah loop yang dilakuakn dalam 1 detik
```

```
    L = block.pos
```

```
    L2 = min
```

```
    Lhat = norm(L) #norm mengubah menjadi vektor satuan, vektornya panjangnya bernilai 1 satuan
```

```
    Lhat2=norm(L2)
```

```
    s=mag(L)-L0 #mag adalah nilai mutlak, pertambahan panjang, mag l artinya besaran dari vector l
    dengan rumus akar dari x kuadrat, y kuadrat, z kuadrat
```

```
    s2=mag(L2)-L0
```

```
    Fspring=-ks*s*Lhat #or Lhat = L/mag(L)
```

```
    Fsingle_spring=-single_ks*s2*Lhat2
```

```
        #enter the spring stretch
```

```
        #enter the spring force
```

```
    Fnet = Fspring + Fgrav      #net force
```

```
    Fsingle_net=Fsingle_spring+Fgrav2
```

```
    #if t == 0:
```

```
    #    print(Fnet)
```

```
    pblock = pblock + Fnet*deltat #update momentum
```

```
    block.pos = block.pos + (pblock/mblock)*deltat #update position
```

```
    spring.axis=block.pos
```

```
    spring2.axis=block.pos
```

```
    psingle_block=psingle_block+Fsingle_net*deltat
```

```
    single_block.pos=single_block.pos+(psingle_block/single_mblock)*deltat
```

```
    min=single_block.pos-vec(2,0,0)
```

```
    single_spring.axis=min
```

```
    t = t + deltat
```

```
gaya=-ks*mblock*deltapanjang
```

```
info_massa= "massa: " + str(mblock)
```

```
info_koef="koefisien pegas: " +str(ks)
```

```
info_pertpanjang="Pertambahan Panjang: "+str(deltapanjang)
```

```
info_gaya="gaya pada pegas: " +str(gaya)
```

```
information=[info_massa, info_koef, info_pertpanjang, info_gaya]
```

```
tipeFile = [("Text file", '*.txt'), ('All files', '*.*')]
```

```
out=asksaveasfilename(filetypes=tipeFile)
```

```
# buka file untuk ditulis
```

```
file_bio = open(out, "w")
```

```
# tulis teks ke file
file_bio.write('\n').join(information))
# tutup file
file_bio.close()
```

```
#if bisa dimanfaatkan untuk mengukur apakah gaya termasuk besar atau tidak
#luaran terformat dan string mengatur posisi anu itu
#tabel coba dimanfaatkan untuk mengukur ketinggian dari pegas itu sendiri

#kurangan: luaran terformat, if, if3, tabel
```