Lab07. Particle System

2 Environment for Particle System

- Files :
  - CGEngine.py, Background.py, Camera.py, Lighting.py, Scene.py, Time.py Particle.py
- CGEngine.py
  - Class Graphics : drawGrid(), drawBall()
  - Class Loading : __init__(), setBackground(), reshape(), grid():grid mode 설정, rotaeGrid(), timerStart(), timerStop(), timerReset(), getDt()-timer의 getDt() 호출, getEt(), frame(), afterFrame(), start(), drawBall(), cameraAt(), addObject(), addSphere(), addCube(), addTransparentObject(), addTransparentSphere(), addTransparentCube()
- Background.py
  - Class Background : __init__(), loadImage(), draw()
- Camera.py
  - Class Camera : __init__(), applyCamera(), setLens(), setPos(), setAsp()
- Lighting.py
  - Class Lighting : __init__(), LightSet(), LightPosition(), SetLightPos()
  - Class VisualObj:__init__(), translate(), rotate(), scale(), color(), setNameAndType(), addChild(), find(), drawObject()
  - Class Scene : __init__(), addObject(), addTransparentObject(), find(), draw()
- Timer.py
  - Class Timer : __init__(), isTimerOn(), start(), stop(), reset(), getDt(), getEt()

Environment for Particle System(cont.)

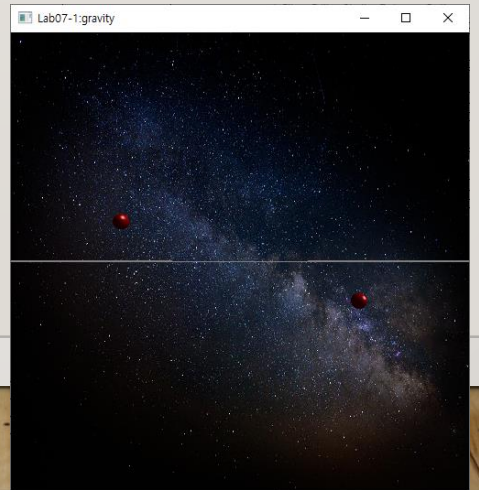
- Particle.py
  - Class Particle : __init__(), draw(), cdraw(), set(), setColPlane(), setRadius(), setMass(), setGravity(), addForce(), resetForce(), simulate(), computeForce(), colHandlePair(), colHandle()

```
def __init__(self):
    self.loc = np.array([0.,0.,0.])
    self.vel = np.array([0.,0.,0.])
    self.radius = 1.0
    self.mass = 1.0
    self.force = np.array([0., 0., 0.])
    self.gravity = np.array([0., 0., 0.])
    self.colPlane = None #np.array([0., 1., 0., 0.])

    return
```

4 Simulation Without using Particle

```
from OpenGL.GLUT import *
from OpenGL.GL import *
from OpenGL.GLU import *

import random
import numpy as np

import CGEngine

# without any particle system
class mySim(CGEngine.Loading):
    def __init__(self, w, h, title):
        super(mySim,self).__init__(w, h, title)

        self.loc = []
        self.vel = []
        self.mass = []
        self.force = []
        self.loc.append(np.array([-15.,  5., 0.])) #loc of ball1
        self.loc.append(np.array([ 15., -5., 0.])) #loc of ball2
        self.vel.append(np.array([  5.,  0., 0.])) #vel of ball1
        self.vel.append(np.array([ -5.,  0., 0.])) #vel of ball2
        self.mass.append(1.0) # mass of first ball
        self.mass.append(20.0) # mass of second ball
        self.force.append(np.array([0., 0., 0.]))#Force of ball 1
        self.force.append(np.array([0., 0., 0.]))#Force of ball 2

        self.cameraAt([0,0,50], [0,0,0])


        self.setBackground(b"bg_cosmos.jpg")

    def initObjects(self):
        self.loc[0] = np.array([-15., 5., 0.])
        self.loc[1] = np.array([15., -5., 0.])
        self.vel[0] = np.array([5., 0., 0.])
        self.vel[1] = np.array([-5., 0., 0.])
```

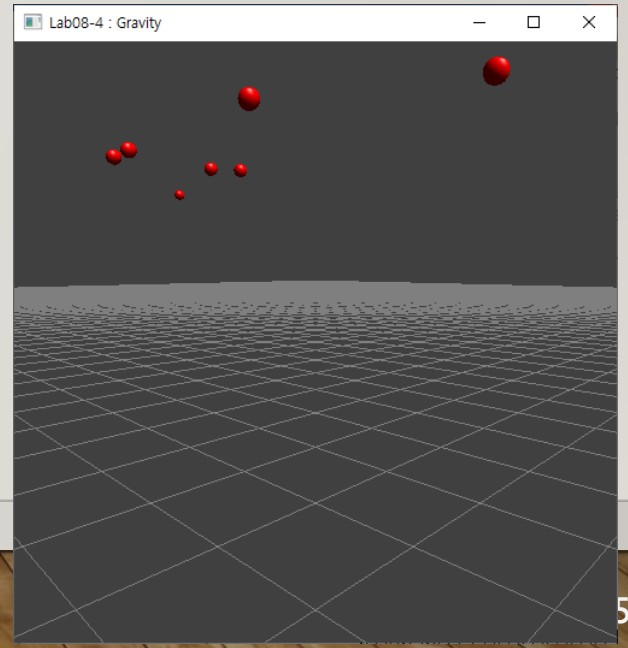
```
def frame(self):

        dt = self.getDt()

        super(mySim,self).frame()
        # your code here

        # compute force
        for i in range(0, 2) :
            self.force[i] = np.array([0.,0.,0.])

        G = 350.0
        dir = self.loc[1] - self.loc[0]
        d = np.linalg.norm(dir)
        dir = dir / d;
        mag = G * self.mass[0] * self.mass[1] / (d*d)
        self.force[0] = mag * dir;
        self.force[1] = -self.force[0]

        # simulate with the force
        for i in range(0, 2) :
            self.vel[i] += self.force[i]*dt/self.mass[i]
            self.loc[i] += self.vel[i]*dt

        for balls in self.loc:
            self.drawBall(balls)

        super(mySim,self).afterFrame()
```



5 Simulation using Particle

```
from OpenGL.GLUT import *
from OpenGL.GL import *
from OpenGL.GLU import *

import random
import numpy as np
import math

import Particle

import CGEngine


def myRand(start, end) :
    interval = end - start
    return start + interval * random.random()

class mySim(CGEngine.Loading) :

    def __init__(self, w, h, title):
        super(mySim,self).__init__(w,h,title)

        self.particle = []
        for i in range(10) :
            self.particle.append(Particle.Particle())


        self.initObjects()


    def initObjects(self):
        for i in range(10) :
            self.particle[i].set(np.array([myRand(-10,10),5.0,myRand(-
10,10)]))
            self.particle[i].setRadius(0.2)
            self.particle[i].setGravity(np.array([0., -9.8, 0.]))

        return
```

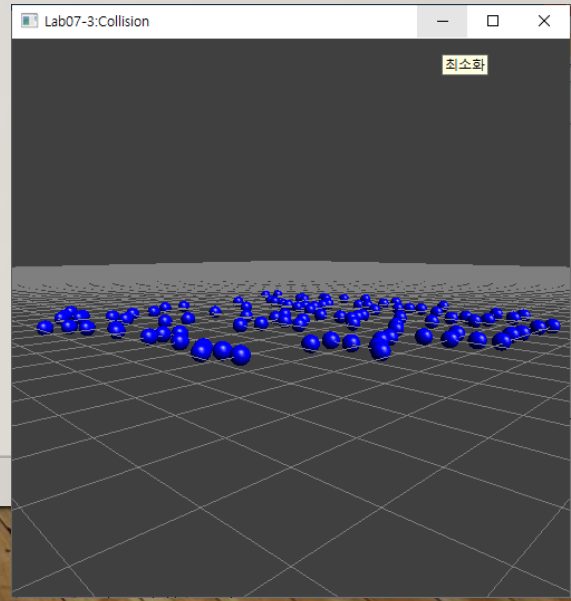
```
    def frame(self):
        dt = self.getDt()


        super(mySim,self).frame()

        for p in self.particle :
            p.simulate(dt)

        for p in self.particle :
            #p.draw()
            p.cdraw([1.0,0.0,0.0,1.0])


        super(mySim,self).afterFrame()

ani = mySim(500,500, b"Lab08-4 : Gravity")
ani.grid(True)


def key(k, x,y) :
    if k == b' ':
        if ani.timer.timerRunning:
            ani.timerStop()
        else:
            ani.timerStart()
    if k == b'r':
        ani.initObjects()


def draw():
    ani.frame()

ani.start(draw, key)
```



Simulation using Particle and collision

```
from OpenGL.GLUT import *
from OpenGL.GL import *
from OpenGL.GLU import *

import random
import numpy as np
import math

import Particle
import CGEngine


def myRand(start, end) :
    interval = end - start
    return start + interval * random.random()

class mySim(CGEngine.Loading) :

    def __init__(self, w, h, title):
        super(mySim,self).__init__(w,h,title)

        self.particle = []
        for i in range(100) :
            self.particle.append(Particle.Particle())


        self.initObjects()


    def initObjects(self):
        root2 = math.sqrt(2.0)
        for i in range(100) :
            l = np.array([myRand(-6, 6), 0.0, myRand(-6, 6)])
            self.particle[i].set(l,-l*0.1)

            self.particle[i].setRadius(0.2)
            self.particle[i].setGravity(np.array([0., 0.0, 0.]))

        return
```

```
    def frame(self):
            dt = self.getDt()

            super(mySim,self).frame()

            for p in self.particle :
                p.simulate(dt)
                p.colHandle()

            for i in range(100) :
                for j in range(i+1, 100) :

self.particle[i].colHandlePair(self.particle[j])

            for p in self.particle :
                p.cdraw([0.0,0.0,1.0,1.0])
            super(mySim,self).afterFrame()

ani = mySim(500,500, b"Lab07-3:Collision")
ani.grid(True)


def key(k, x, y) :
    if k == b' ':
            if ani.timer.timerRunning:
                ani.timerStop()
            else:
                ani.timerStart()
    if k == b'r':
            ani.initObjects()


def draw():
    ani.frame()

ani.start(draw, key)
```



7 실습문제

1. Lab07-2에서 공이 바닥에 닿으면 다시 튀어오르도록 수정하자.
2. 아주 작은 particl을 이용하여 분수대에서 물이 쏟아오르는 것을 시뮬레이션 해보자. 동시에 튀어오르는 partcle은 10개씩 생성되어 나온다. 바닥에 떨어진 particle은 일정한 시간이 지나면 소멸된다.
3. 2번이 구현되면 사실성을 높이기 위해 그 양을 조정해보자.

- 최종제출할 프로그램은 lab07-2 프로그램을 개선한 것과 분수를 시뮬레이션한 프로그램이다. 즉 lab07-2-m.py와 lab07-found.py 프로그램을 압축하여 제출하면 된다.

실습문제 제출

이번 실습문제는

- 모든 프로그램은 압출하여 하나의 파일로
- PLMS Lab xx 해당 번호폴더
- 마감은 12/9

제출합니다.