예 : 컴퓨팅적 사고력을 활용하여 물리 모델링하기

기준

- CC 수학 F-TF: 삼각 함수를 사용하여 진자 현상 모델링하기.
- CC 수학 S-ID: 2개의 범주와 양적 변수의 데이터를 요약하기, 표현하기 및 해석하기.
- CC 수학 G-GPD: 방정식과 기하학적 특성 표현하기.

개요

구체적인 개별 실험들에서 자연 현상을 분석하여 패턴을 찾고, 방정식을 만들어 규칙을 도출함으로써 과학자들은 몇 세기에 걸쳐 컴퓨팅적 사고력의 원리를 활용하여 왔다. 이러한 사례들을 종합하여 학생들은 이전의 과학자의 세계를 더 잘 이해 할 수 있다.

주의: 이 사례는 체계화된 교수학습과정안을 의도한 것이 아니라 컴퓨팅적 사고력을 사용하여 현재의 교과 과정을 보완하기 위한 방법을 제공하는 것이다.

전제 조건

- 학생들은 삼각 함수와 루트를 사용할 수 있다.

재료

- 모든 사례는 컴퓨터를 사용하지 않고 수행할 수 있다. - 화이트보드나 종이 / 연필의 사용은 제한 없다.
- 선택적 기술 사항:
  - Python 2.7과 VPython 5 (Windows, Mac 또는 Linux)를 설치합니다.
  - 스프레드시트 소프트웨어 - Google 스프레드시트, 오픈 오피스, 엑셀
  - GeoGebra
  - 이 실험을 반복하여 자신의 데이터를 수집하려면 :
    - 진자 (질량과 길이를 변화시킬 수 있기 때문에 문자열에 와셔가 잘)
    - 경사면(책 위의 자, 장난감 자동차 경주 트랙)과 미끄러지는 공.
  - 데이터를 수집하는 방법 :
    - 스톱워치, 줄자 / 눈금자
    - Probeware / 센서 (트래커 비디오 분석, 파스코, 버니어, 물리학 기즈모)

Examples

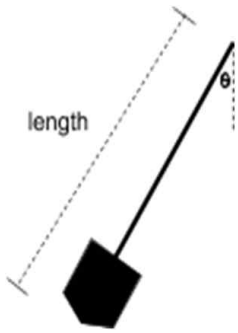
진자
경사면과 자유 낙하
포물선 운동

지도 시 유의 사항:
각각의 개념은 기존의 물리학 교실에서 이미 배운 것이다. 학생과 가상 모델을 사용하면 적어도 두 가지 장점이 있다. 첫 번째는 시뮬레이션 실행을 통해 모든 사례를 분석하고 추적 할 수 있다. 실험실 실험은 종종 측정에 의한 실험 오차를 많이 포함하고 있지만, 가상 시뮬레이션은 정확성과 정밀도를 가질 수 있다. 둘째, 직접적인 장점으로 실험실에서 실습 학생들에게 실험이 가능하지 않은, 소설과 같은 가설이나 계산을 넘어서 검증을 통한 공식을 확인할 수 있다 (예를 들어, 중력이 없거나 반대가 되었을 때 무슨 일이 일어날까).
학생들이 그들의 가설을 검증하고 수정하기 위한 코드를 제공하고 코드에 관해 질문할 수 있도록 고려해야 한다. 물리 현상에 대한 알고리즘을 생성하기 위해서는 유사 보호(의사코드)를 작성하는 학생들에게 요구 방정식을 넘어 생각하는 것을 격려하는 것이 또 다른 좋은 방법이라고 할 수 있다.

예 1 : 진자

역사는 갈릴레오에게 진자의 움직임을 정의하는 규칙을 발견하는데 기여하도록 하였다. 진자는 중심점에서 자유롭게 흔들리며 움직이는 물체이다. 이야기에 따르면, 갈릴레오는 흔들리는 샹들리에를 보고 영감을 받아 실험을 하였다. 최고조에서의 진자운동을 완료하는 시간은 샹들리에의 시작 각도와 상관없이 지속적으로 움직이는 것처럼 보였다. 흥미로운 사실은 그가 스톱워치를 가지고 있지 않기 때문에 일정한 시간을 측정하기 위해 자신의 맥박에 의존하였다는 것이다.

분해



갈릴레오는 진자의 움직임이 수반 될 것으로 예상했다:

- 삼각 함수 (주기 운동)
- 중력
- 진자 줄의 길이.
- 진자와 물체(있는 경우)의 질량

패턴 인식

갈릴레오는 진자 운동의 주기에 영향을 끼치는 진자의 각도, 길이, 질량을 결정하기 위한 실험을 수행하였다. 그리고 무게와 관계없이 진자 줄 길이가 짧을수록 진자운동이 빠르다는 사실을 발견하였다.

패턴 추상화

진자운동의 시간 변화는 진자 길이의 변화와 지수함수적 관계가 있다.

알고리즘 설계

현운동의 방정식 개발은 갈릴레오의 예측을 확인해주며 간단한 진자주기 시간을 정확하게 예측해 준다.

$$\text{시간 (초)} = \quad \pi \quad \frac{ength}{g}$$

g는 중력 가속도 (지구 = 9.8m/)

그러나 이 방정식은 작은 각도에서 계산할 수 있으며 단순화되어 있다. 새로운 데이터가 발견되면 알고리즘 및 설명은 항상 개선이 이루어진다.

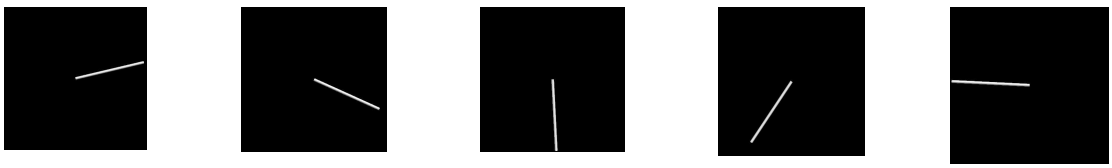
갈릴레오가 의사 코드를 사용하여 문제를 모델링한 사례는 다음과 같다.

1. *환경 (예를 들어, dt와 중력)을 준비하고 지정된 길이의 진자, 회전축과 각도를 만든다.*
2. *각도를 변경한다.*
3. *가속도, 속도 및 새로운 각도의 위치를 업데이트 한다.*
4. *2-3단계를 반복한다.*

코넬 대학의 제임스 Stehna의 진자 모델 코딩

```
from visual import *

#The next few lines are   pseudocode line 1
g = 9.8  #Acceleration   due to gravity
L = 1.0  #Physical   length of pendulum
theta = (2. * pi) / 3.       #Initial upper angle (from vertical)
velocity = 0.0  #Start   pendulum at rest (decimal precision is needed)
dt = 0.01    #Time steps for pendulum
pen = cylinder(pos=(0,   0, 0), axis = (L * sin(theta), -L * cos(theta), 0),
      radius = 0.02)

while (True):  #Pseudocode line 4
   rate(50)    #Slow down graphics to be visible to humans
   #The   following lines are pseudocode line 3
   acceleration = -(g / L) * sin(theta)  #Update acceleration for   new
angle
   velocity   += acceleration * dt  #Update velocity according to
acceleration
   theta   += velocity * dt  #Pseudocode line 2, update position
   #Pendulum   moves when pen.axis changes
   pen.axis = (L * sin(theta), -L * cos(theta), 0)
```

(다른 시간에) 출력의 예



예 2 : 경사면과 자유 낙하

어떻게 자유 낙하와 같이 이상적으로 미끄러지는 공의 가속도를 계산할까? 갈릴레오는 그가

다른 각도에서 경사면을 사용해 보았기 때문에 빠르게 움직이는 물체의 데이터를 수집하는 것이 어렵다는 것을 알고 있었다.

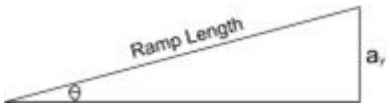
분해

갈릴레오가 한때 그랬던 것처럼 경사면 아래로 구르는 공을 모델링하기 위해, 우리는 각 요소들 간의 문제를 해결해야 한다. 경사면 아래로 물체의 가속도를 결정하는 데 관여하는 인자는 그 물체가 놓아지는 각도와 시간이다.

패턴 인식

갈릴레오는 각도를 조절 할 경사면에서 실험을 준비하였다. 그는 시간의 경과를 측정하기 위해 물 시계와 자신의 심박수를 사용하여 정확한 시간을 재는데 한계를 가지고 있었다. 그는 그의 실험에서 가속도가 주어진 각도와 일정하다는 사실을 발견했다. 그는 또한 각도가 90도 (수직)에 가까워질수록 가속도의 한계 (g)에 접근한다는 것을 발견했다.

패턴 추상화



Ay = g * 램프의 길이 * sin( )

1. θ가 90도에 가까워질수록 **g**는 한계에 도달한다.

2. θ가 0도에 도달하면 효과적으로 **g**는 0으로 떨어진다. 그 후, 아이작 뉴턴의 통찰로 발견된 관성의 법칙을 일컬어 운동의 제 1의 법칙이라고 부른다.

알고리즘 설계

각도에 의해 가속이 변화한다는 사실로부터 우리는 어떤 각도에서 경사면 아래로 구르는 공의 시뮬레이션을 개발 할 수 있는 방법을 알게 된다.

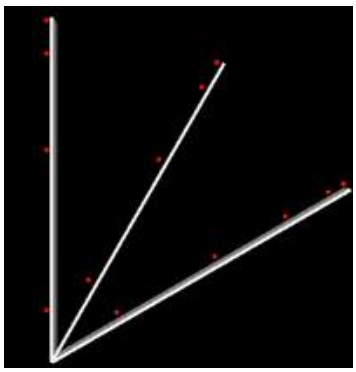
의사 코드의 예 :

1. *환경 (예 : DT와 중력). 준비하기*
2. *일정한 경사 / 각도로 경사를 만든다.*
3. *아래로 구르는 공을 만든다.*
4. *주어진 시간 동안의 속도와 위치를 업데이트한다.*
5. *경사로의 끝에 공이 구를 때까지 4 단계를 반복한다.*
6. *여러 경사로를 만들고 2-5 단계를 반복한다.*

```
from visual import *

#Pseudocode line 1, setup   the physical attributes
r = 0.2
velocity = 30
L = 10 #Ramp   Length
dt = 0.01    #Change in time
```

```
dv = dt * -velocity    #Change in velocity (negative direction)
scene.center = (L / 3,   L / 2, 0)  #The focus of the screen

#Pseudocode line 6
for theta in range(30, 100, 30):  #Create ramps at 30/60/90   degrees
   theta   = radians(theta)  #Converts angle to radians for trig   functions

   #Pseudocode   line 2, create a ramp
   ramp   = box(pos = ((L * 0.5 * cos(theta)), (L * 0.5 * sin(theta)), 0),
      axis   = (L * cos(theta), L * sin(theta), 0), width = 0.4, height =
0.1)

   #Psuedocode line 3, ball follows the slope of the   ramp
   ball   = sphere(pos = (ramp.length * cos(theta) - r * sin(theta),
      ramp.length   * sin(theta) + r * cos(theta), 0), radius = r,
      make trail   = True, trail type = "points", interval = 30)
   ball.trail_object.color   = color.red

   ball.velocity   = 0  #Ball's starting velocity (not moving)

   while (True):  #Pseudocode   line 5
      if ball.pos.y < 0:  #If the ball hits the   ground, make it invisible
         ball.visible   = 0
         break
      rate(50)    #Slow the refresh rate down to be seen

      #Pseudocode   line 4
      ball.velocity   += dv * sin(theta)  #Update the velocity
      ball.x   += ball.velocity *dt * cos(theta)  #Update the x position
      ball.y += ball.velocity * dt *   sin(theta)  #Update the y position
```

샘플 출력



예 3 : 포물선 운동

시간의 경과를 쉽게 계산할 수 있었던 제 1차 세계 대전 중에는 컴퓨터의 힘이 매우 분명해졌다. 한 사람이 컴퓨터에게 적절한 지시를 설정한다면 컴퓨터는 빠르고 정확한 계산을 할 수 있다.

분해

두 방향으로 움직이는 물체는 x와 y 방향의 운동 속도를 가지고 있다. 공기 저항이 없다고 가정하면, 물체의 가속도는 중력에 의한 것이다. 끝으로, x 방향 (속도, 가속도)의 물리적 특성은 y 방향의 특성에 영향을 미치지 않는다.

패턴 인식

물체에 작용하는 외부 힘이 없다면 물체는 직선으로 날아갈 것이다. 중력은 객체에 작용하는 y 방향으로 속도를 줄이고, 공기 저항은 양방향의 속도를 감소시킨다. 먼 거리에서 목표물을 공격하기 위해서는 목표물보다 더 높은 곳을 조준해야만 한다.

패턴 추상화

포물선 운동을 위한 규칙과 방정식은 x와 y 요소에서 물체의 성질을 나누어 도출 할 수 있다.

물체의 속도는 시간이 지나면 위치의 변화율과 관련된 의미 있는 속도 즉, 그것의 위치와 관련 있다. 마찬가지로, 가속도는 속도와 관련 있다.

$$\text{velocity} = \frac{\triangle position}{\triangle\ ime} \quad \text{and} \quad \text{acceleration} = \frac{\triangle velocity}{\triangle time}$$

따라서 시간이 흐르면, y 방향의 물체의 위치는:

$$\text{position} = \frac{acceleration * time}{2} + \text{velocity} * \text{time}$$

(이것은 적분에 의한 결과임을 알아두기)

이 이상적인 상황에서는 x 방향의 가속도가 일정량의 이동을 초래하지 않는다.

알고리즘 설계

시뮬레이션을 위한 알고리즘을 작성하기 위해, 방정식을 사용하는 것은 시간이 흐르면 속도와 위치가 얼마나 변화하는지, 그리고 목표물에 화살을 맞히던지 그렇지 않던지 간에 최초의 속도는 어떠한 영향을 끼치는지 관찰할 수 있다.

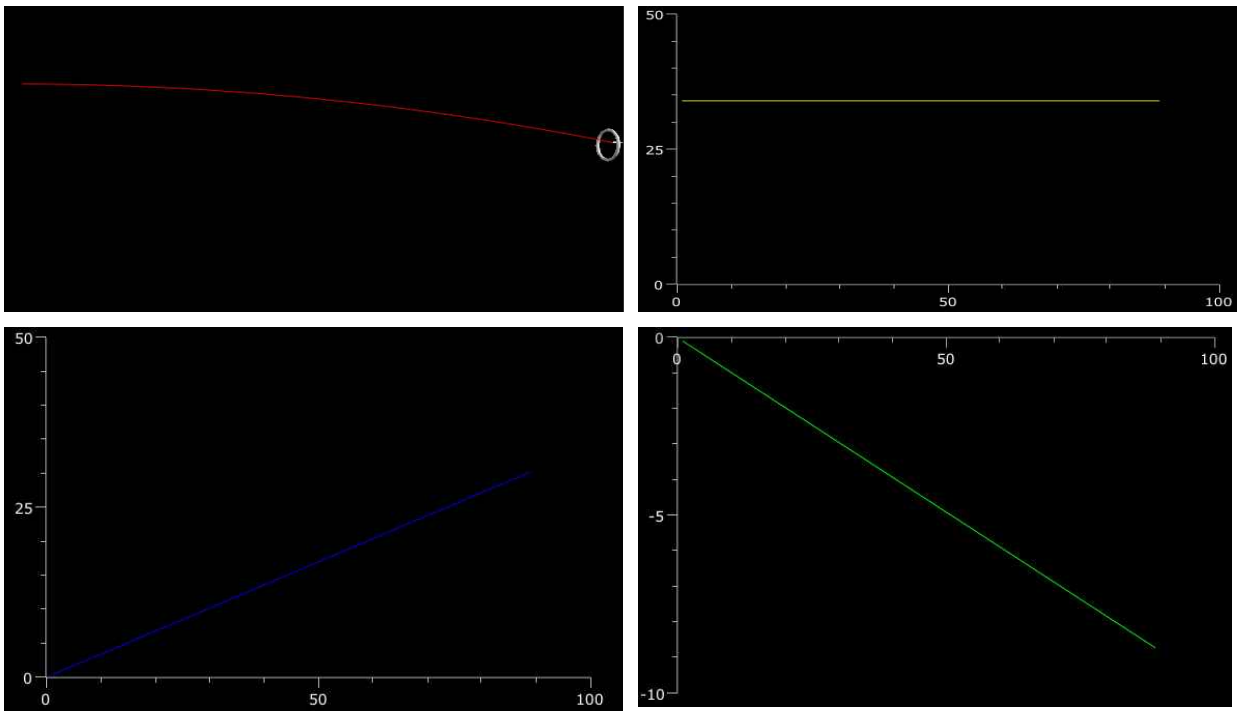
가능한 의사 코드 :

1. *환경 (초속 DT) 준비 및 물체의 초기 위치 파악하기.*
2. *위치를 설정한 화살표와 목표물 만들기.*
3. *시간이 지남에 화살표의 위치와 속도 업데이트하기.*
4. *화살이 과녁을 맞출수 있게 조절될 때까지 반복하기.*

```
from visual import *
from visual.graph import *
scene1 = display(title   = "Crossbow Lab", width = 600, height = 400,
   center=(15,   0, 0))
scene1.autoscale = True

#Data Windows (optional)
graph1 = gdisplay(title   = "Velocity in y direction", xmax = 100, xmin = 0,
```

```
   ymax   = 0, ymin = -10, x = 600, y = 400, width = 600)
vely = gcurve(color =   color.green)

graph2 = gdisplay(title   = "Velocity in x direction", xmax = 100, xmin = 0,

   ymax   = 50, ymin = 0, x = 600, y = 0, width = 600)
velx = gcurve(color =   color.yellow)

graph3 = gdisplay(title   = "Distance in x", xmax = 100, xmin = 0,
   ymax   = 50, ymin = 0, x = 0, y = 400, width = 600)
distx =   gcurve(color=color.blue)

#Pseudocode line 1,   physical properties
arrow.velocity =   vector(34, 0, 0)
arrow.trail =   curve(color=color.red)
dt= 0.01
t = 0
a = -9.8

#Pseudocode line 2,   Physical objects
arrow = arrow(pos = (0, 5, 0), axis=(5, 0, 0), shaftwidth = 10, length = 1,
   color   = color.red)
target = ring(pos =   (30, 1, 0), axis = (1, 0, 0), radius = 1)

#Pseudocode line 4, loop   until arrow has gone through or missed
while arrow.pos.x < target.pos.x:
   rate(50)
   arrow.pos   += arrow.velocity * dt #Pseudocode line 3
   arrow.velocity.y   = arrow.velocity.y + a * dt
   arrow.trail.append(pos   = arrow.pos)

   t +=   1
   vely.plot(pos   = (t, arrow.velocity.y))
   velx.plot(pos   = (t, arrow.velocity.x))
   distx.plot(pos = (t, arrow.pos.x))
```

샘플 출력



기타 리소스

- 진자 방정식의 도출 : http : //hyperphysics.phy-astr.gsu.edu/hbase/pend.html
- 인터랙티브 Gailieo 실험 : http : //www.pbs.org/wgbh/nova/physics/galileo-experiments.html
- 하늘 비디오 갈릴레오 배틀 (PBS) : http : //video.pbs.org/video/2036276385/
- 다른 예는 "Python과 물리"또는 검색 "VPython 물리 "
- http://code.google.com/p/python-physutil/:VPython 물리를 위한 많은 기능이 Physutil 모듈을 통해 찾을 수 있다.

더 많은 수업과 사례는 Google의 검색 컴퓨팅 사고력의 웹 사이트에서 찾을 수 있다.