Programación en Lenguaje Java

Práctica 2.3. Simulación del movimiento orbital de un satélite y su planeta

Michael González Harbour

Mario Aldea Rivas

Departamento de Matemáticas,
Estadística y Computación

Este tema se publica bajo Licencia:
Creative Commons BY-NC-SA 4.0

Práctica 2-3: Simulación del movimiento orbital de un satélite y su planeta

El 14 de julio de 2015 la nave New Horizons llegará al planeta enano Plutón, después de un viaje de 9 años

Desde la distancia, la nave ha podido fotografiar la extraña órbita de Caronte alrededor de Plutón (Imágenes de la NASA)

© Michael González Harbour y Mario Aldea, 8/oct/15

Descripción

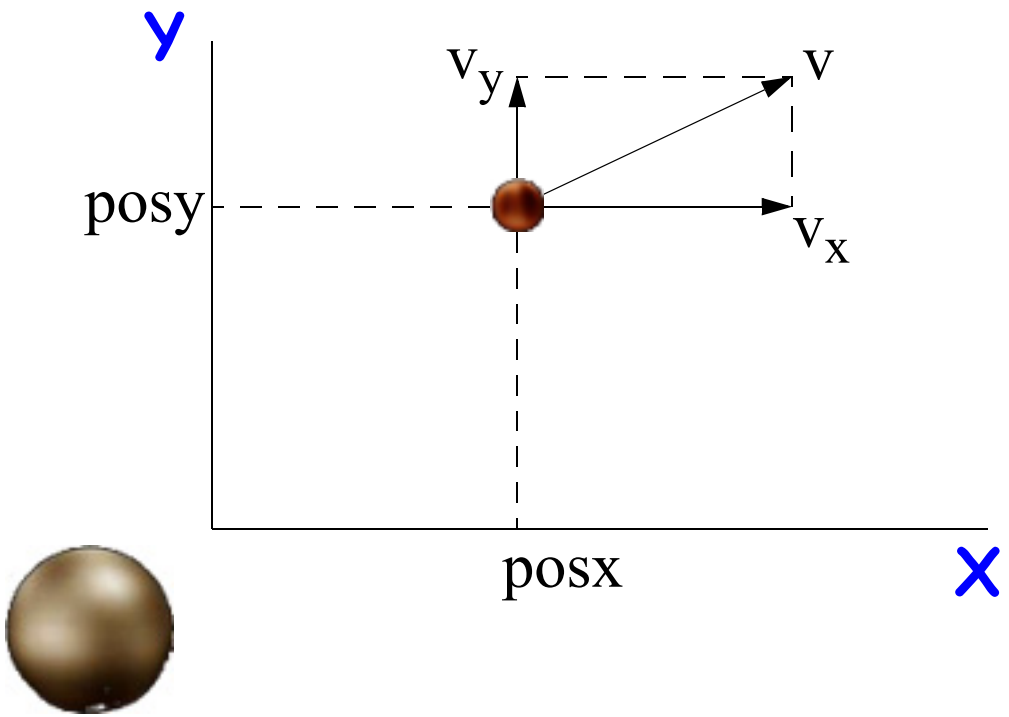
Objetivos: Crear una clase con atributos y métodos

Simularemos el movimiento orbital de la luna Caronte y su planeta Plutón

Representaremos cada uno de estos astros con un objeto

Atributos:

- posición (`posx`,`posy`)
- velocidad (`vx`,`vy`)
- masa (`masa`)



© Michael González Harbour y Mario Aldea, 8/oct/15

Ecuaciones del movimiento

Ecuaciones del movimiento, para calcular nueva posición y velocidad transcurrido un intervalo t

$$posx^{nuevo} = posx + v_x \cdot t + \frac{a_x \cdot t^2}{2}$$

$$posy^{nuevo} = posy + v_y \cdot t + \frac{a_y \cdot t^2}{2}$$

$$v_x^{nuevo} = v_x + a_x \cdot t$$

$$v_y^{nuevo} = v_y + a_y \cdot t$$

Son las ecuaciones del movimiento uniformemente acelerado

Aceleración de la gravedad para un astro

La aceleración vectorial $\vec{a}$ se calcula con la fórmula de Newton:

$$\vec{a} = \frac{G \cdot M_{op}}{x^2 + y^2} \cdot \vec{u}$$

siendo:

- G=6.674*10^{-11} N*m^2/Kg^2 la constante de gravitación universal
- M_{op} es la masa del astro opuesto
- $\vec{u}$ es un vector unitario en la dirección al astro opuesto

$$\vec{u} = (u_x, u_y) = \frac{x}{\sqrt{x^2 + y^2}}, \frac{y}{\sqrt{x^2 + y^2}}$$

siendo $(x,y)=(posx_{op}\text{-}posx, posy_{op}\text{-}posy)$ el vector al astro opuesto (op)

© Michael González Harbour y Mario Aldea, 8/oct/15

Aceleración de la gravedad (cont.)

Por tanto, las componentes de la aceleración son:

$$x = posX_{op} - posX$$

$$y = posY_{op} - posY$$

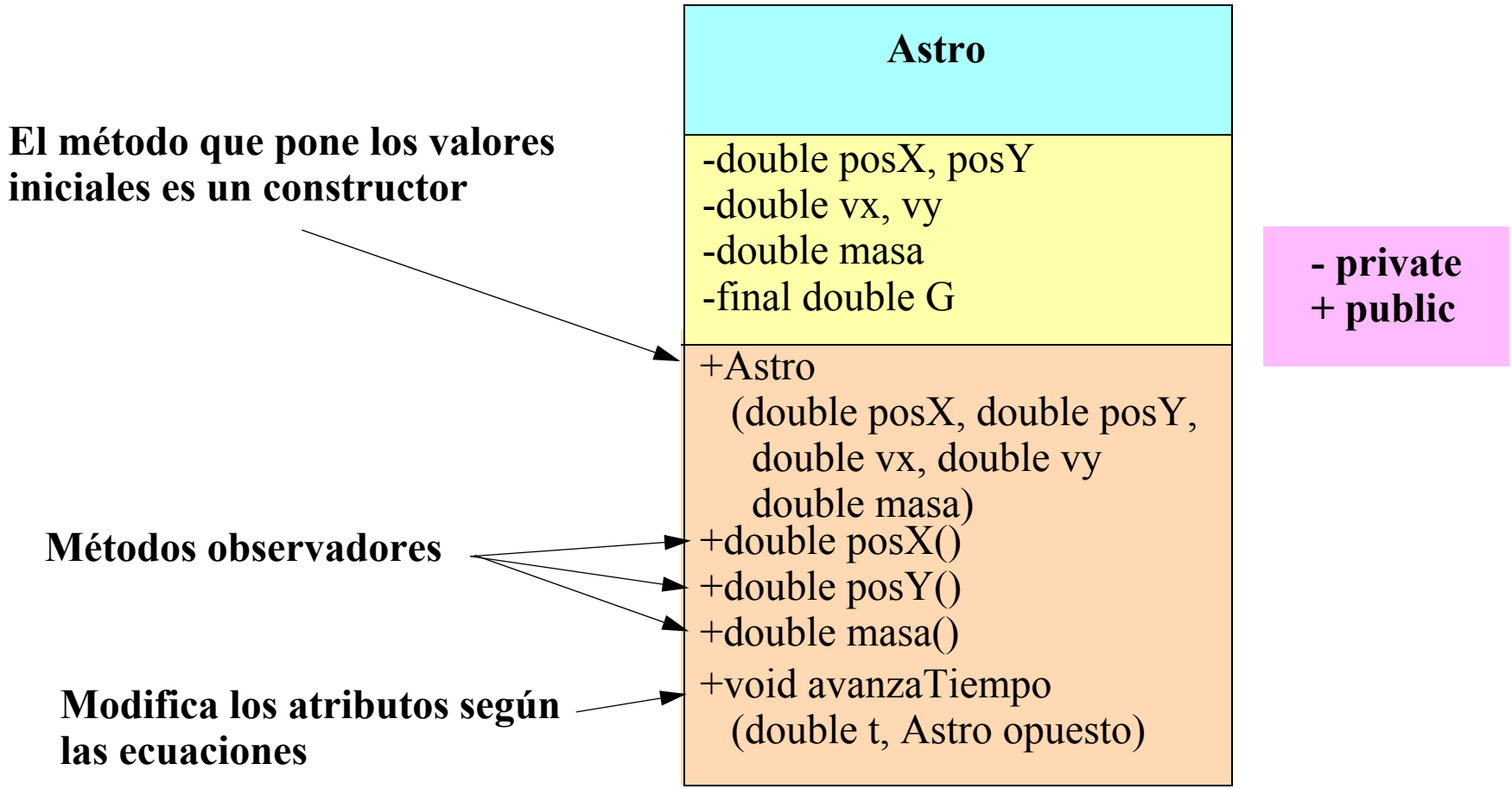
$$a_x = \frac{G \cdot M_{op}}{x^2 + y^2} \cdot \frac{x}{\sqrt{x^2 + y^2}}$$

$$a_y = \frac{G \cdot M_{op}}{x^2 + y^2} \cdot \frac{y}{\sqrt{x^2 + y^2}}$$

donde el subíndice op se refiere al astro opuesto

© Michael González Harbour y Mario Aldea, 8/oct/15

Diagrama de la clase



© Michael González Harbour y Mario Aldea, 8/oct/15

Detalles sobre la clase

- Atributos y otros datos
  - Usaremos unidades del sistema internacional
  - Documentar con comentarios las unidades usadas en los atributos, los argumentos de los métodos, y los valores retornados por ellos
- Constructor
  - copia en los atributos los respectivos parámetros del mismo nombre
- Observadores
  - retornan el atributo del mismo nombre
- `avanzaTiempo`: aplica las fórmulas de las páginas 3 y 5
  - primero calcula el vector (`x`,`y`) al astro opuesto, guardando sus componentes en variables locales
  - luego calcula las aceleraciones (`ax`, `ay`) y las guarda en variables locales
  - luego modifica las posiciones y finalmente las velocidades (página 3)

Realización

a) Escribir la clase `Astro`

b) Ejecutar el simulador de Plutón-Caronte:

- copiar en el proyecto las clases `Espacio` y `SimuladorPluton` que se ofrecen, así como los ficheros `pluton.png` y `caronte.png`
- ejecutar el método `main`, y observar la gráfica que se produce

Informe

Entregar:

- El código java de la clase `astro`
- Una captura de la imagen después de que se realice al menos una órbita completa

Parte avanzada

Realización: Modificar las clases `Espacio` y `SimuladorPluton` para mostrar imágenes de Plutón y Caronte, en vez de puntos:

- crear duplicados de las clases, con otros nombres, con objeto de conservar las clases originales)
- consultar la documentación de la clase `Dibujo` del paquete `fundamentos`, para ver cómo dibujar una imagen
- localizar en el código del duplicado de `Espacio` el lugar donde se dibujan los puntos, y en su lugar dibujar las imágenes
- modificar el código del `main` para usar el nuevo `Espacio`

Entregar:

- El código java de las clases modificadas
- Una captura de la gráfica obtenida

Observaciones

Nota sobre el funcionamiento del simulador: El movimiento del astro en realidad no es uniformemente acelerado

- la aceleración depende de la posición

Si el incremento de tiempo usado en `avanzaTiempo` es muy pequeño se puede considerar que la aceleración cambió muy poco en ese intervalo

- el error que se comete es pequeño

Se puede probar a cambiar el tiempo que se avanza a cada paso en el `main` del simulador, y ver el resultado incorrecto