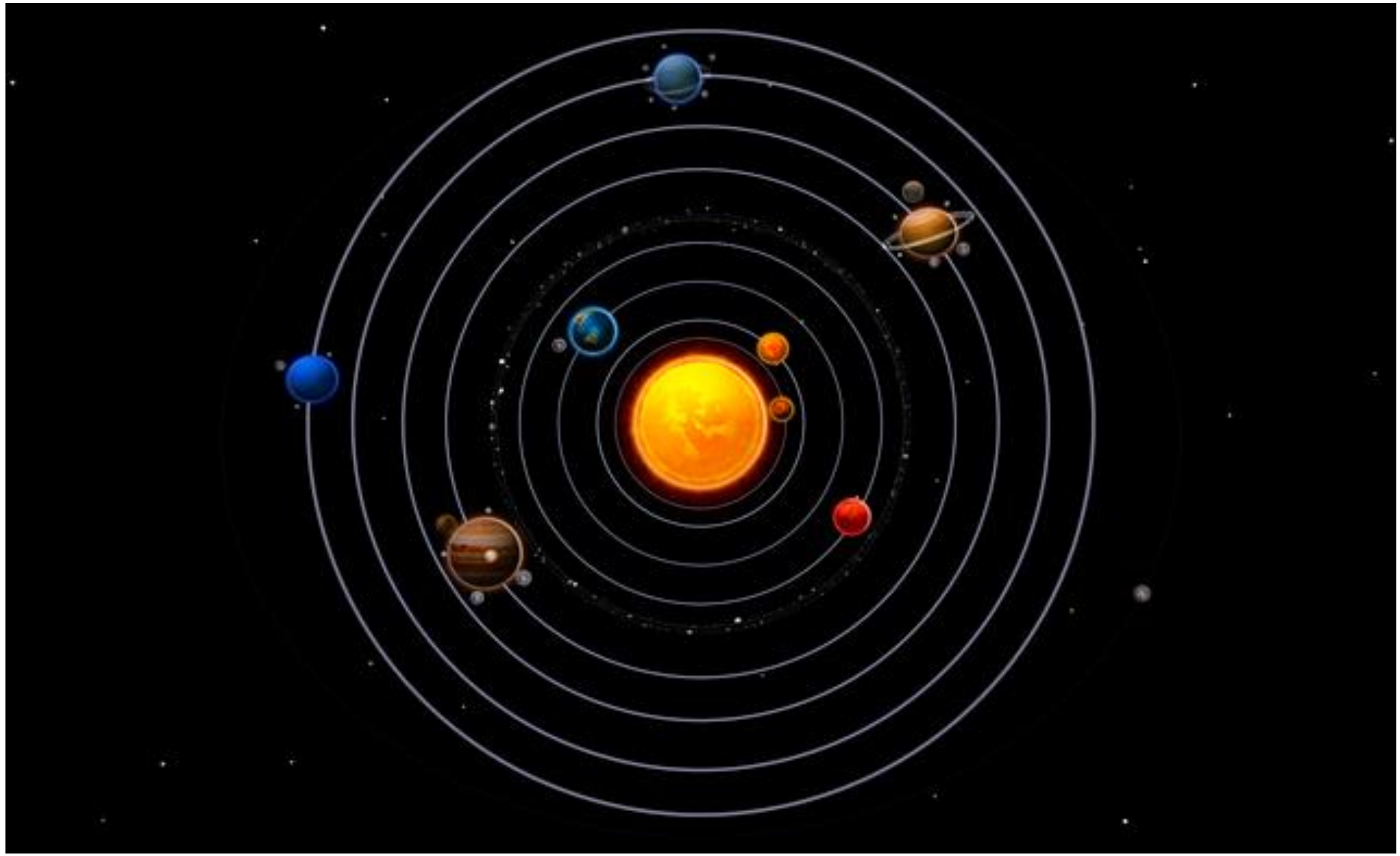
Auxiliar Pygame - MVC

Ejemplo simulador de planetas



Que se pide?

- Simular una aplicación 2D que permita dibujar distintos planetas en pantalla
- Debe soportar primitivas e imágenes (sprites)
- Mediante el input del usuario añadir planetas de manera aleatoria

Primer paso: Idear bien en papel qué hacer

- Identificar modelo, ¿Qué objetos se presentan en la escena?
  - Planetas
- ¿Qué hacen los planetas?, ¿Qué variables necesitaría para modelar lo que se pide?
- Una vez que tengo los planetas, ¿Cómo dibujarlos?

Modelo

- Variables del planeta:
  - Velocidad angular
  - Radio del planeta
  - Radio de giro
  - Color del planeta
  - Textura
- ¿Se necesitan mas? Por lo general responder esto se puede sólo cuando se programa, la idea es que se pueda hacer todo con el **MINIMO**

Modelo

- Variables del planeta Posición:
  - Siempre se trabaja con posiciones absolutas o relativas, en este caso la absoluta es con respecto a su propio sistema de coordenadas.
  - Dado que el mundo tiene una coordenada (x,y) se definirá cada planeta con respecto a una coordenada “origen” la cual corresponde al centro del Sol.

  - Como inicialmente no se tiene la coordenada del Sol esta se deberá ingresar una vez se haya instanciado un objeto.

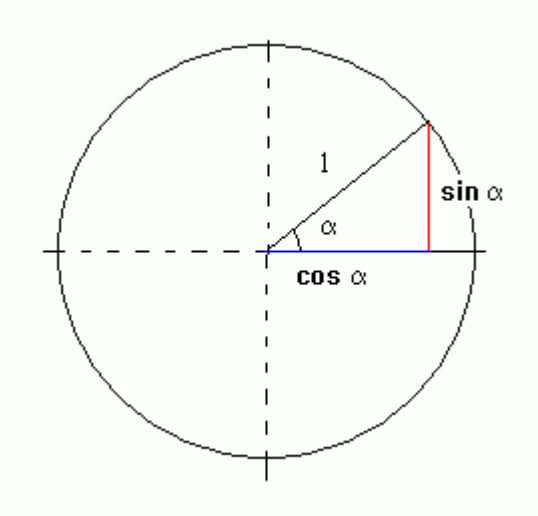
Modelo

- Variables del planeta Posición:

```
36          # Guarda las variables
37          self._velocidad_angular = w
38          self._radio_planeta = int(rplaneta)  # No se pueden dibujar radios tipo 3.4, no existen 3.4 pixeles
39          self._radio_giro = rgiro
40          self._theta = thetai
41          self._color = colorplaneta
42
43          # Calcula la posicion absoluta (con respecto a 0,0) y la guarda como propiedad del objeto
44          self._posicion_absoluta = [0, 0]  # (x,y)
45
46          # Posición del origen, sólo se puede cambiar una vez se crea el planeta
47          self._pos_origen = [0, 0]
48
```

Modelo

- ¿Cómo rotar los planetas?



(xr,yr) Posición **relativa** del planeta con respecto al origen global (xg, yg)

(xa,ya) Posición **absoluta** del planeta con respecto a su 0,0

$xa = R*\cos(\theta)$

$ya = R*\sin(\theta)$

$xr = xa + xg$

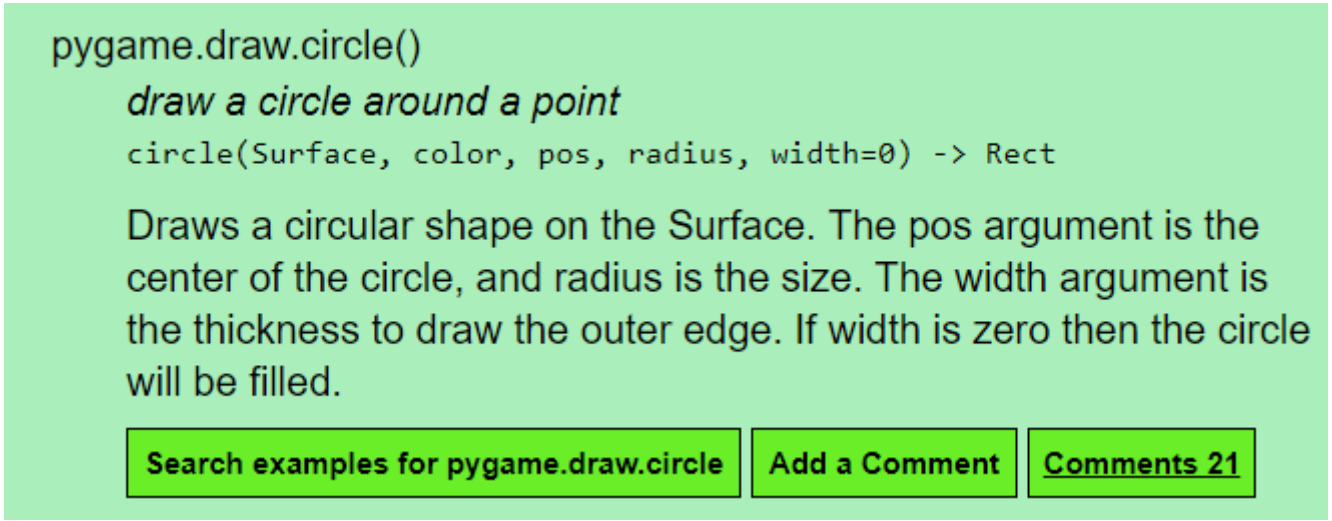
$yr = ya + yg$

Modelo

```
61
62      # noinspection PyTypeChecker
63      def actualizar_posicion(self, dt):
64          """
65          Actualiza la posición en un tiempo dt
66          :return:
67          """
68
69          # Actualiza el ángulo
70          self._theta += self._velocidad_angular * dt
71          self._theta %= 360
72
73          # Pasa a radianes
74          theta_rad = self._theta * math.pi / 180
75
76          # Calcula el par (x,y) de la posición
77          x = self._radio_giro * math.cos(theta_rad)
78          y = self._radio_giro * math.sin(theta_rad)
79
80          # Guarda la posición
81          self._posicion_absoluta[0] = x
82          self._posicion_absoluta[1] = y
83
```

Modelo

- ¿Cómo dibujar un planeta?



Modelo

- ¿Si el planeta tiene textura?

- Cargar imagen con **pygame.image.load**
- Volcar la imagen con **surface.blit**

Modelo

```
116
117     def dibujar_planeta(self, surface):
118         """
119         Dibuja el Planeta en un canvas
120         :param surface: Superficie de Pygame
121         :return:
122         """
123
124         # Calcula la posición relativa al centro
125         x_r = self._pos_origen[0] + self._posicion_absoluta[0]
126         y_r = self._pos_origen[1] + self._posicion_absoluta[1]
127
128         # Muy importante, pasar coordenadas a entero
129         x_r = int(x_r)
130         y_r = int(y_r)
131
132         # Si se definió una imagen se dibuja
133         if self._img is not None:
134             # Se resta el radio para centrar
135             surface.blit(self._img, [x_r - self._radio_planeta, y_r - self._radio_planeta])
136         else:
137             # circle(Surface, color, pos, radius, width=0) -> Rect
138             pygame.draw.circle(surface, self._color, [x_r, y_r], self._radio_planeta)
139
```