

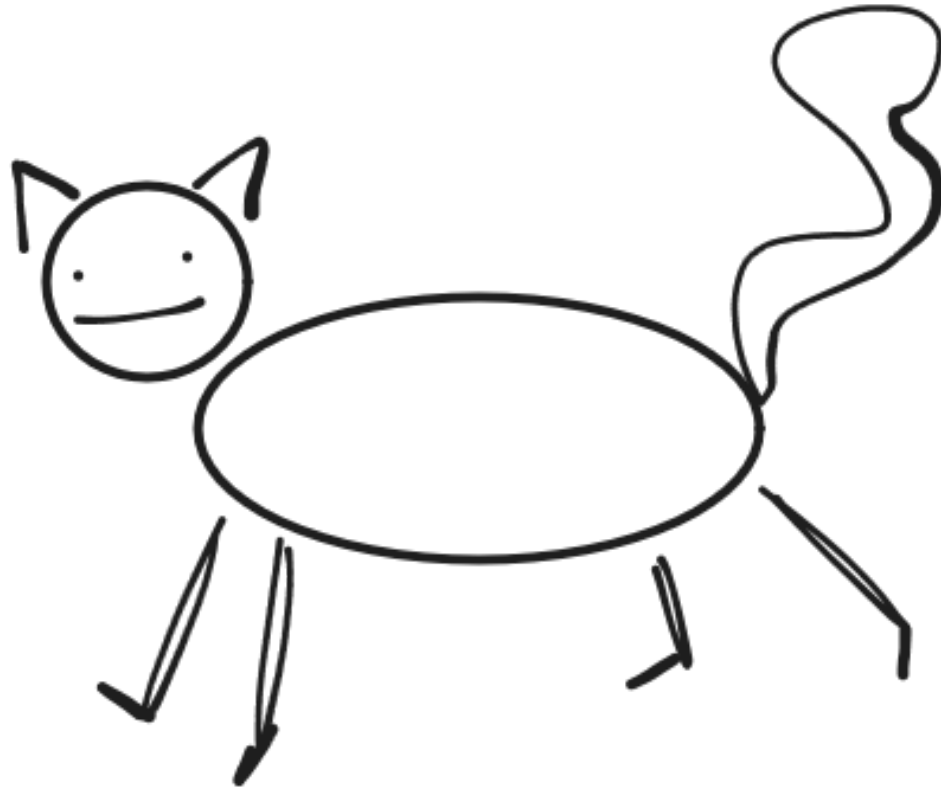
CC3501: Computación Gráfica

Auxiliar 5: Gráfo de Escena e Implementación

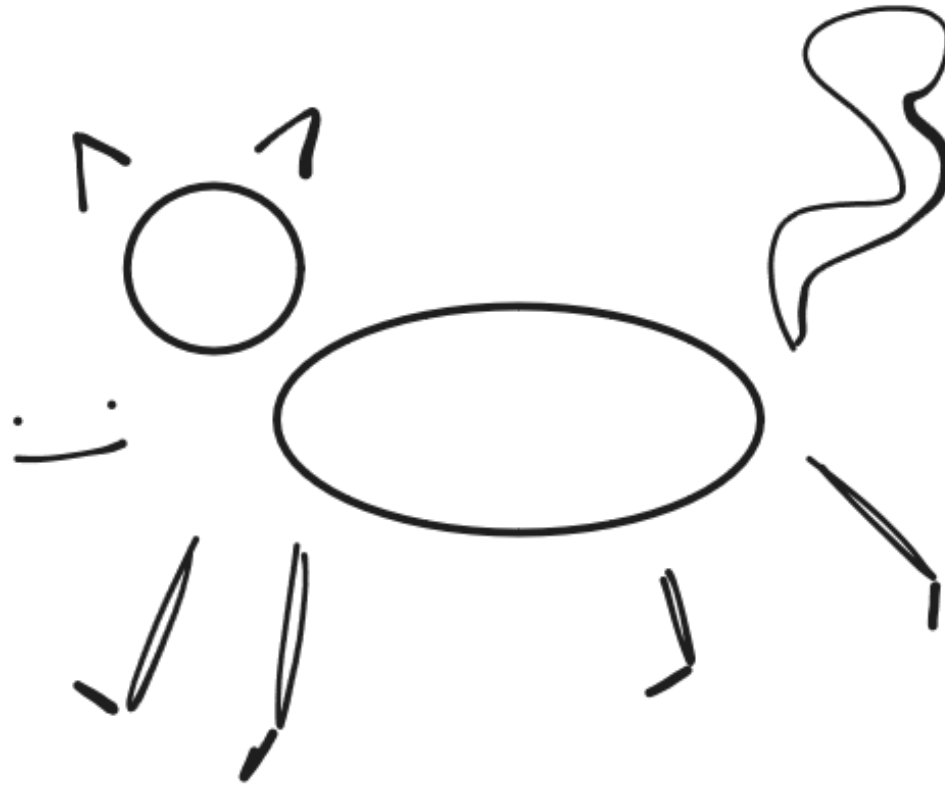
Auxiliares: Julieta Coloma y Vicente Gonzales

Motivación

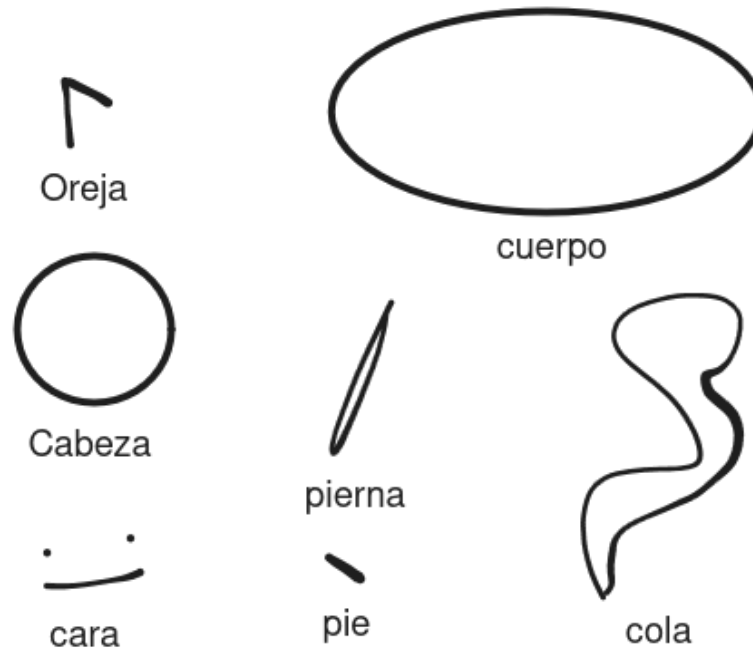
Digamos que queremos construir este gato



Pero queremos hacerlo parte por parte



Pie, pierna y oreja se repiten en distintas posiciones. Por lo que tenemos las siguientes partes únicas:



El gato es nuestro todo, nuestro conjunto completo. Pero podemos definirlo en base a subconjuntos de sus partes.

El gato es nuestro todo, nuestro conjunto completo. Pero podemos definirlo en base a subconjuntos de sus partes.

Por ejemplo, podemos definir **pata** como una **pierna** y un **pie** en cierta posición.

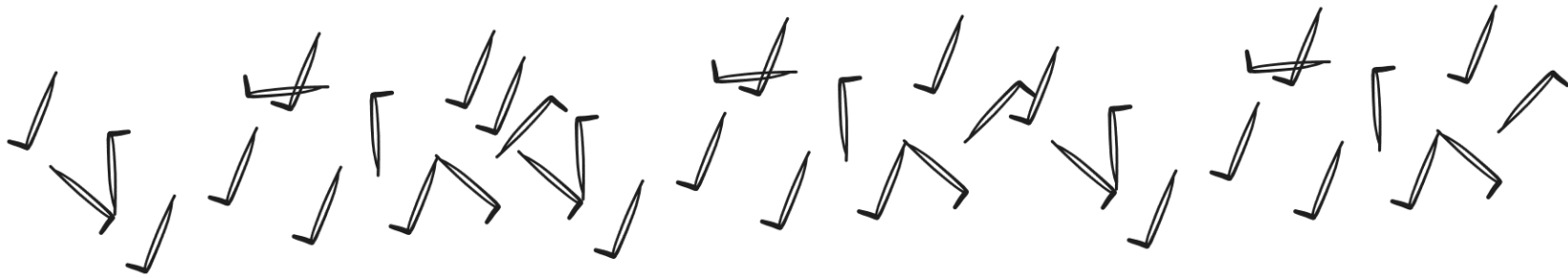


El gato es nuestro todo, nuestro conjunto completo. Pero podemos definirlo en base a subconjuntos de sus partes.

Por ejemplo, podemos definir **pata** como una **pierna** y un **pie** en cierta posición.



Y luego crear muchas patas.



Volviendo al gato, tendremos los siguientes subconjuntos:

- Pata = Pierna + pie
- Cuerpo = Torso + Cola + 4 Patas
- Orejas = Oreja + Oreja
- Cabeza = Cara + expresión + Orejas

Finalmente nuestro gato será Cabeza + Cuerpo

Yo definí estos conjuntos por un propósito educativo (se me dió la gana), podrían haber más formas de definir al gato...

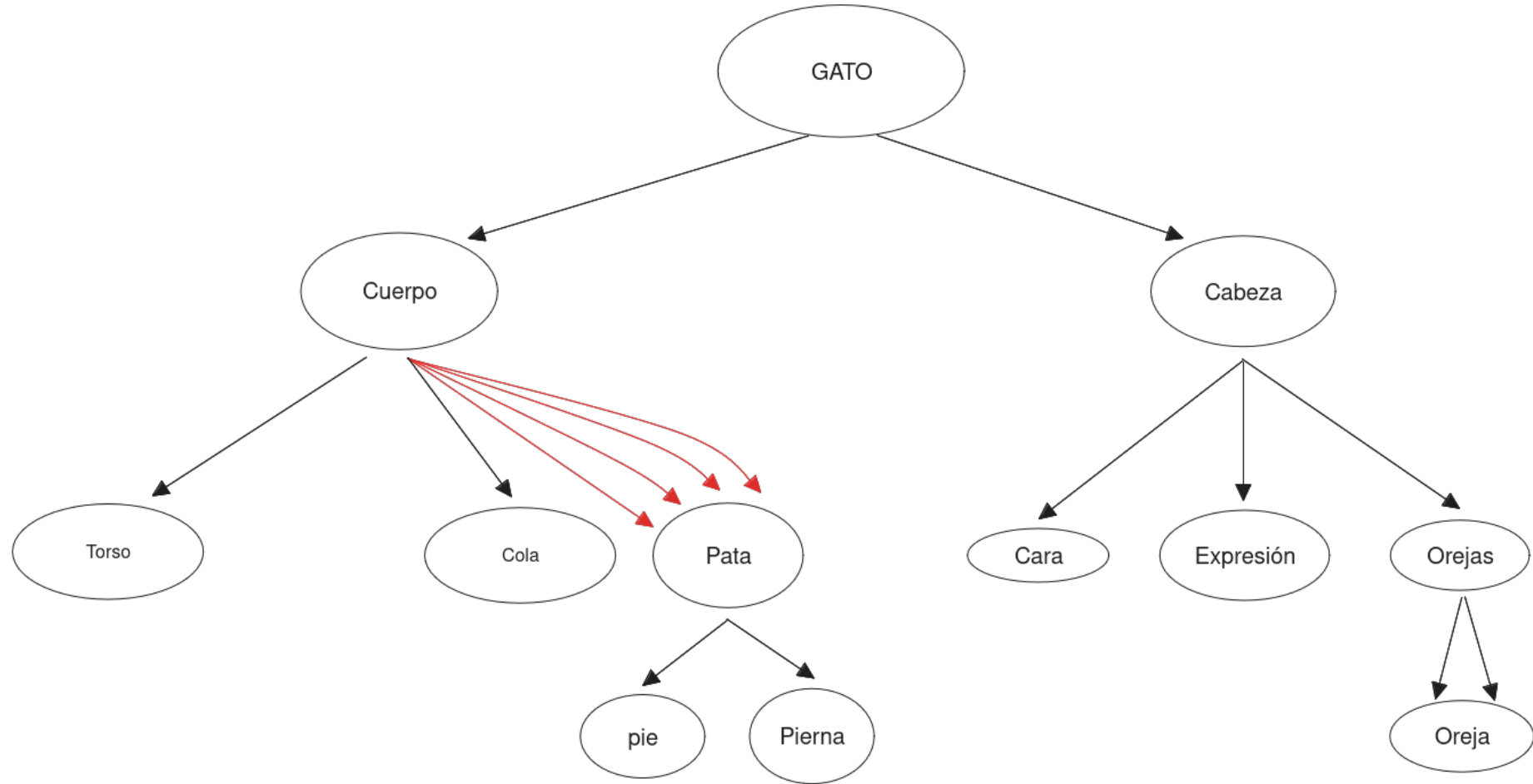
Volviendo al gato, tendremos los siguientes subconjuntos:

- Pata = Pierna + pie
- Cuerpo = Torso + Cola + 4 Patas
- Orejas = Oreja + Oreja
- Cabeza = Cara + expresión + Orejas

Finalmente nuestro gato será Cabeza + Cuerpo

Entonces, si decimos que **gato** es nuestra raíz... Y le siguen **Cabeza** y **Cuerpo**, y de cada uno salen mas objetos... es como un grafo...

El Gráfo



Gráfo de Escena

- La raíz es nuestra escena completa.
- Las hojas (nodos finales) representarán nuestros objetos bases.
- Los nodos internos (nodos que no son hojas ni raíz) serán los subconjuntos

No hay un consenso absoluto sobre cómo hacer un grafo de escena, podemos encontrar grafos donde los nodos internos son transformaciones... u otras cosas

El punto es reconocer que existe una **relación jerárquica** entre objetos.

Implementación

Se incluye en el módulo `scene_graph.py` la clase `SceneGraph()`

Y se usa así

```
graph = SceneGraph()
```

#o así

```
graph_camara = SceneGraph(camara)
```

Añadir nodos

Usamos la función `add_node()`

```
graph.add_node("nombre",  
               attach_to =  
               mesh =  
               pipeline =  
               color =  
               transformations =  
               scale =  
               position =  
               rotation =  
               )
```

Actualizar cosas en el gráfo

```
graph[objeto][parametro] = algo
```

Por ejemplo

```
graph["cabeza"]["scale"] = [0.5, 0.5, 0.5]
```

#o quizás:

```
pos = graph["car"]["position"]
```

```
graph["car"]["position"] = pos + [1, 0, 0]
```

#obtenemos la posición anterior y le sumamos 1 en X

Dibujar todo lo del grafo

```
@controller.event  
def on_draw():  
    graph.draw()
```

easy

El código

“caja negra”

P1: Vaca Mágica

a. Defina dos clases:

`GameModel(vertices, indices, pipeline)`

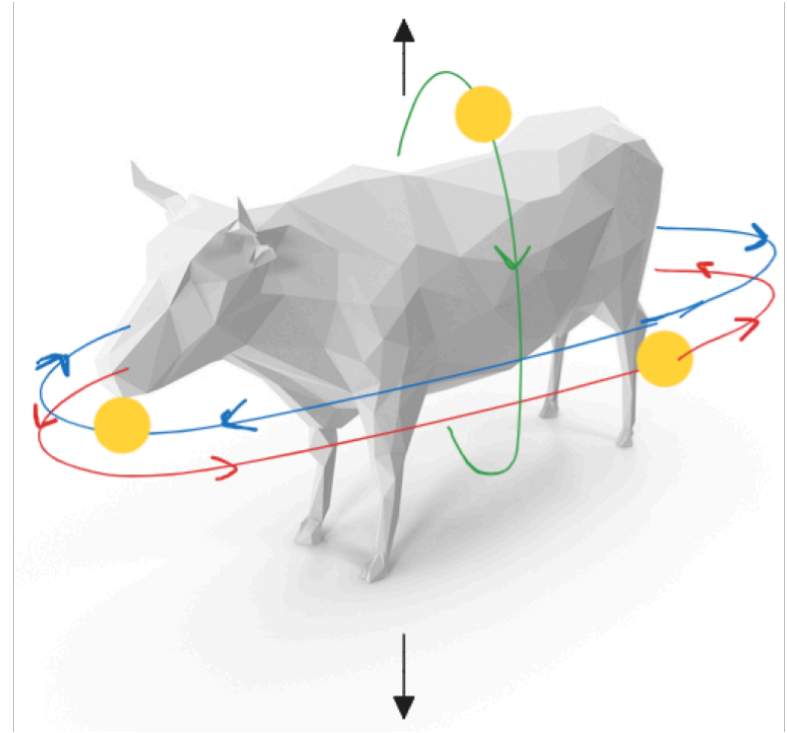
`Mesh(path, pipeline)`

Tome en cuenta que lo que entreguen estas clases se lo pasaremos al nodo en mesh.

b. Utilizando `Mesh()` defina un mesh con el archivo `cow.obj` y otro con `sphere.obj`

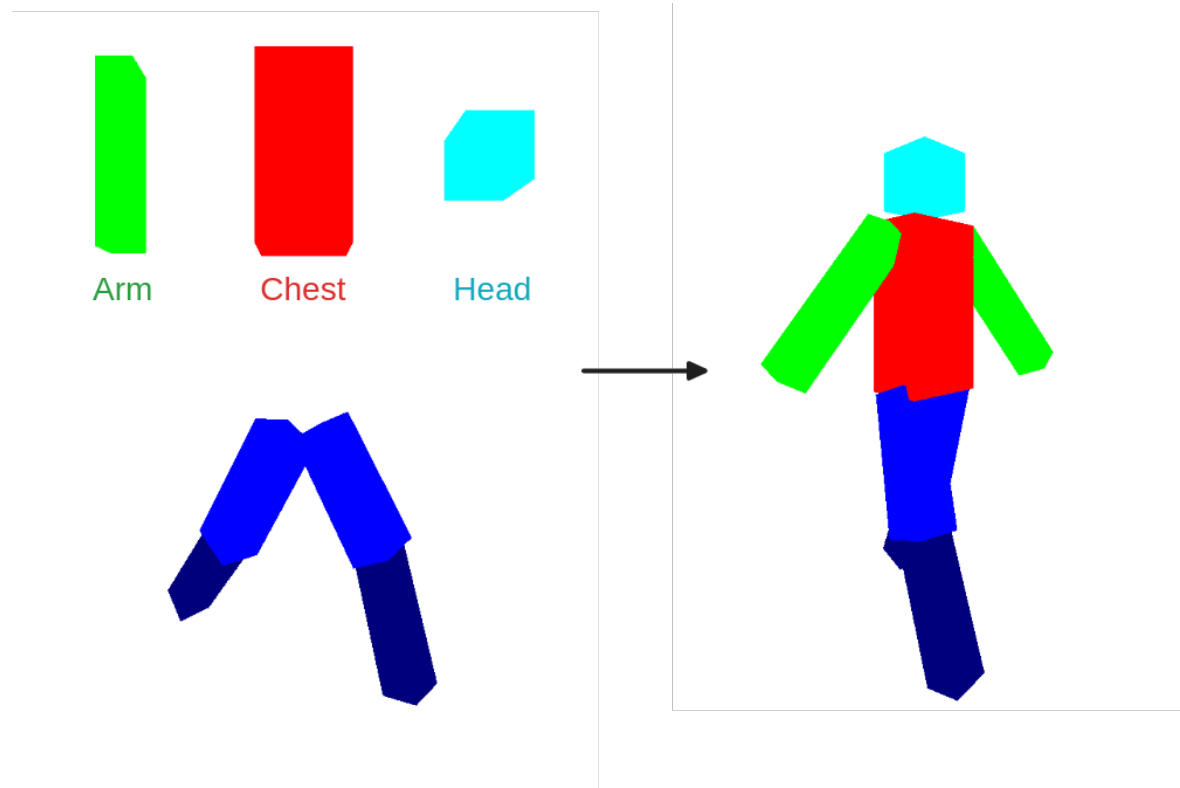
c. Para la siguiente escena, haga el grafo y programelo :)

- La vaca mágica se ubica en el $(0, 0)$, se encuentra levitando entre $y = -1$ e $y = 1$.
- Hay 3 orbs girando al rededor de la vaca mágica

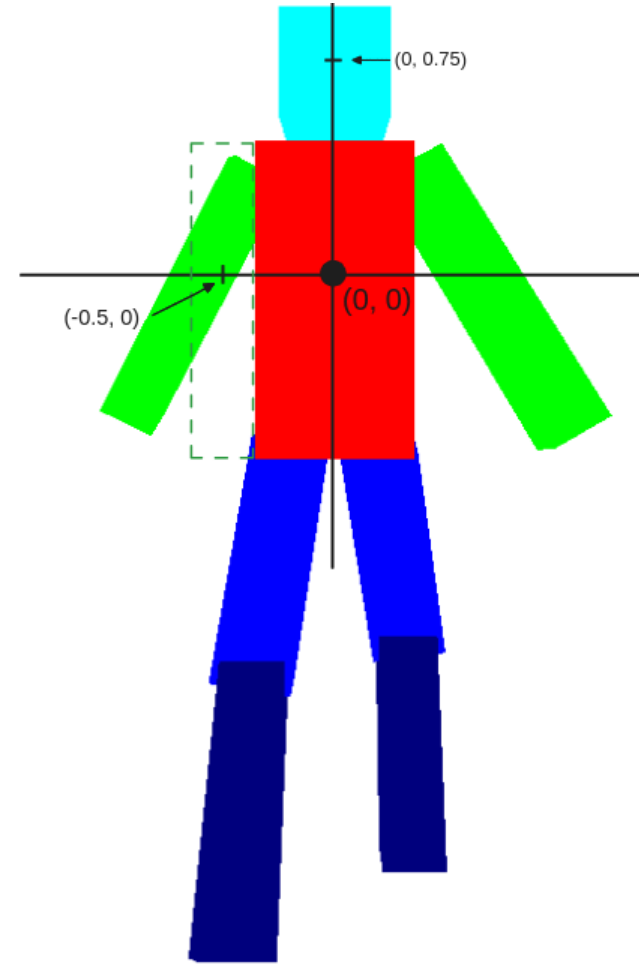


P2: Caminanding

En este problema te entregan un programa abandonado que nadie terminó y hay que terminarlo



- Diseña el grafo de escena para el monito completo.
- Analizando el código ya existente crea los nodos que faltan.
- Finalmente añade el movimiento de los brazos.



d. (Propuesto) Coloca todo el código de la personita en una clase llamada `Person(mesh, position)`.

Además de construir el grafo de escena, esta clase debe poseer las siguientes funciones:

```
def draw():  
    #Dibuja todo el grafo
```

```
def walk():  
    #Ejecuta el código del movimiento
```

Intenta crear montones de personitas con esta clase