

```
1: from os import closerange
2: from ursina import *
3: import json
4: from ursina.prefabs.first_person_controller import FirstPersonController
5:
6: # Classes
7: def init_world(x, z):
8:     sky = Sky(texture="assets/top.png")
9:
10:    floor = Button(parent= scene,
11:                   color = color.white33,
12:                   position = (0,0,0),
13:                   origin_y = 0,
14:                   model = "cube",
15:                   texture = "white_block",
16:                   scale = (x, 1, z)
17:                   )
18:
19:    player = FirstPersonController()
20:
21:
22: def resize(texture: Texture, new_x):
23:     aspect_ratio = texture.width / texture.height
24:     new_y = new_x / aspect_ratio
25:
26:     return (new_x, new_y)
27:
28: class Frame(Button):
29:     def __init__(self, img, position, scale_x, description):
30:         texture = load_texture(img)
31:         scale = resize(texture, scale_x)
32:         self.description = description
33:         super().__init__(model = "cube",
34:                          parent = scene,
35:                          texture = texture,
36:                          position = position,
37:                          color = color.white,
38:                          scale = scale,
39:                          origin_y = -0.5)
40:
41:     def input(self, key):
42:
43:         if self.hovered:
44:             if key == "left mouse down":
45:                 #pass to a viewer-like vision
46:                 print(self.description)
47:                 pass
48:
```

```
49:
50: def load_frames(json_file):
51:     entities = []
52:     with open(json_file, "r") as f:
53:         frames = json.load(f)
54:
55:     north, south = frames["North"], frames["South"]
56:     west, east = frames["West"], frames["East"]
57:
58:     for frame in north:
59:         #positive z, x remain
60:         #negative x rotation
61:         fr = Frame(img="assets/" + frame["file"],
62:                    position = (frame["pos"]["x"], frame["pos"]["y"], frame["pos"]["z"]),
63:                    scale_x = frame["scale"],
64:                    description = frame["description"])
65:         fr.rotation_y += frame["Yrotate"]
66:         fr.rotation_x -= frame["Xrotate"]
67:         entities.append(fr)
68:
69:         antifr = Frame(img="assets/" + frame["file"],
70:                        position = (frame["pos"]["x"], - frame["pos"]["y"] - 1, frame["pos"]["z"]),
71:                        scale_x = frame["scale"],
72:                        description = frame["description"])
73:         antifr.rotation_y += frame["Yrotate"]
74:         antifr.rotation_x -= frame["Xrotate"] - 185
75:         entities.append(fr)
76:
77:     for frame in south:
78:         #negative z, remain x
79:         #positive x rotation
80:         fr = Frame(img="assets/" + frame["file"],
81:                    position = (frame["pos"]["x"], frame["pos"]["y"], - frame["pos"]["z"]),
82:                    scale_x = frame["scale"],
83:                    description = frame["description"])
84:         fr.rotation_y += frame["Yrotate"]
85:         fr.rotation_x += frame["Xrotate"]
86:         entities.append(fr)
87:
88:     for frame in west:
89:         #remain z, negative x
90:         #positive x rotation + 90°
91:         fr = Frame(img="assets/" + frame["file"],
92:                    position = (-frame["pos"]["x"], frame["pos"]["y"], frame["pos"]["z"]),
93:                    scale_x = frame["scale"],
94:                    description = frame["description"])
95:         fr.rotation_y += frame["Yrotate"] + 90
96:         fr.rotation_x += frame["Xrotate"]
```

```
97:         entities.append(fr)
98:
99:     for frame in east:
100:         #remain z, negative x
101:         #positive x rotation - 90°
102:         fr = Frame(img="assets/" + frame["file"],
103:                    position = (frame["pos"]["x"], frame["pos"]["y"], -frame["pos"]["z"]),
104:                    scale_x = frame["scale"],
105:                    description = frame["description"])
106:         fr.rotation_y += frame["Yrotate"] + 90
107:         fr.rotation_x -= frame["Xrotate"]
108:         entities.append(fr)
109:
110:
111:
112:     return entities
113:
114: class Spectator(FirstPersonController):
115:     def __init__(self):
116:         super().__init__(position = (0, 1, 0))
117:         self.flying = False
118:
119:     def update(self):
120:         if self.flying:
121:             self.gravity = 0
122:             self.y += (held_keys["v"] - held_keys["b"]) * time.dt * 10
123:         else:
124:             self.gravity = 1
125:
126:         return super().update()
127:
128:     def input(self, key):
129:         if key == "v":
130:             self.start_flying()
131:         elif key == "c":
132:             self.stop_flying()
133:
134:         return super().input(key)
135:
136:     def start_flying(self):
137:         self.flying = True
138:
139:     def stop_flying(self):
140:         self.flying = False
141:
142: app = Ursina()
143:
144: init_world(100, 100)
```

```
145: frames = load_frames("/Users/rserrano/grepos/ProyectoFinal/src/obras.json")
146: player = Spectator()
147:
148: app.run()
```