

## Evidencia 2. Avances y presentación del reto

**Equipo:** NA

**Grupo:** 301

**Integrantes:**

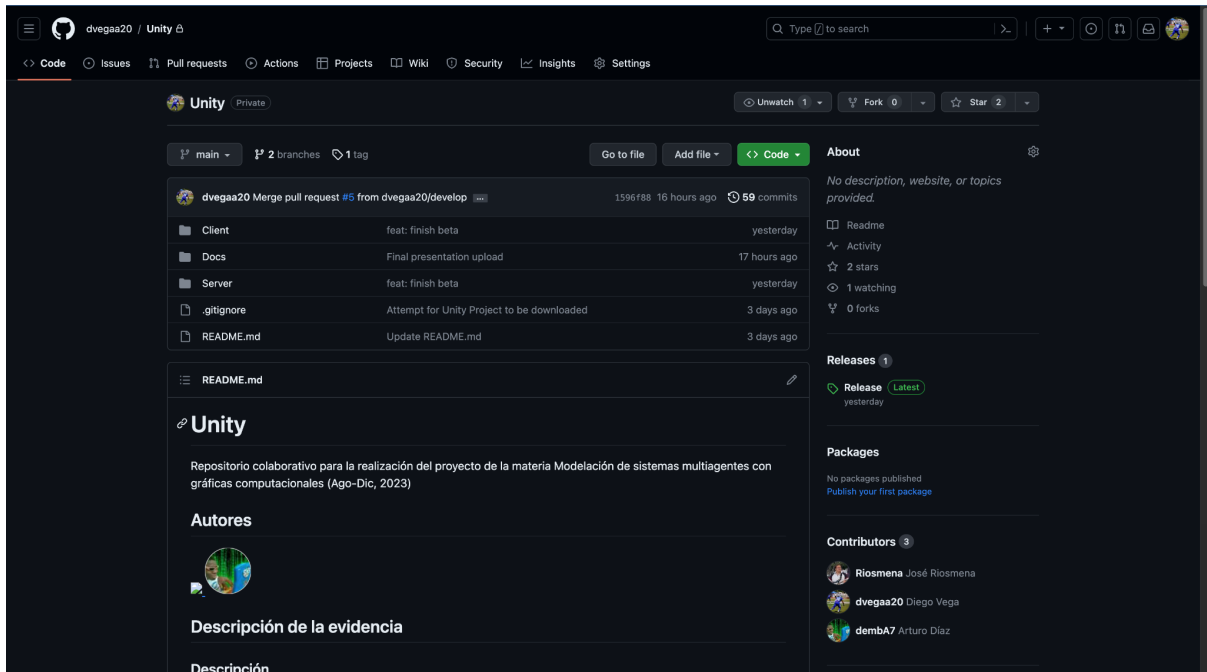
- José Emiliano Riosmena Castañón
- Arturo Cristían Díaz López
- Diego Vega Camacho

**Profesores:**

- Alejandro Fernández
- Denisse Lizbeth Maldonado Flores
- Pedro Oscar Pérez Murueta

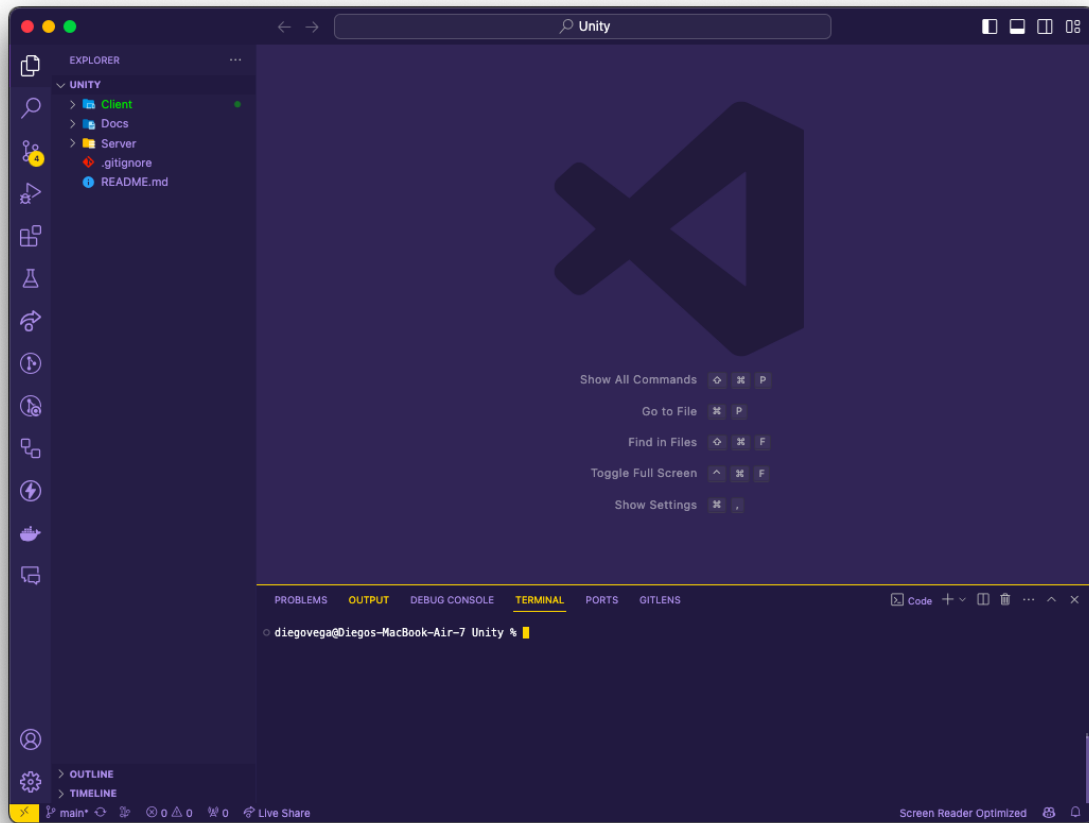
## Proceso de instalación, configuración y ejecución de la simulación.

1. Abre el siguiente link [Unity](#)

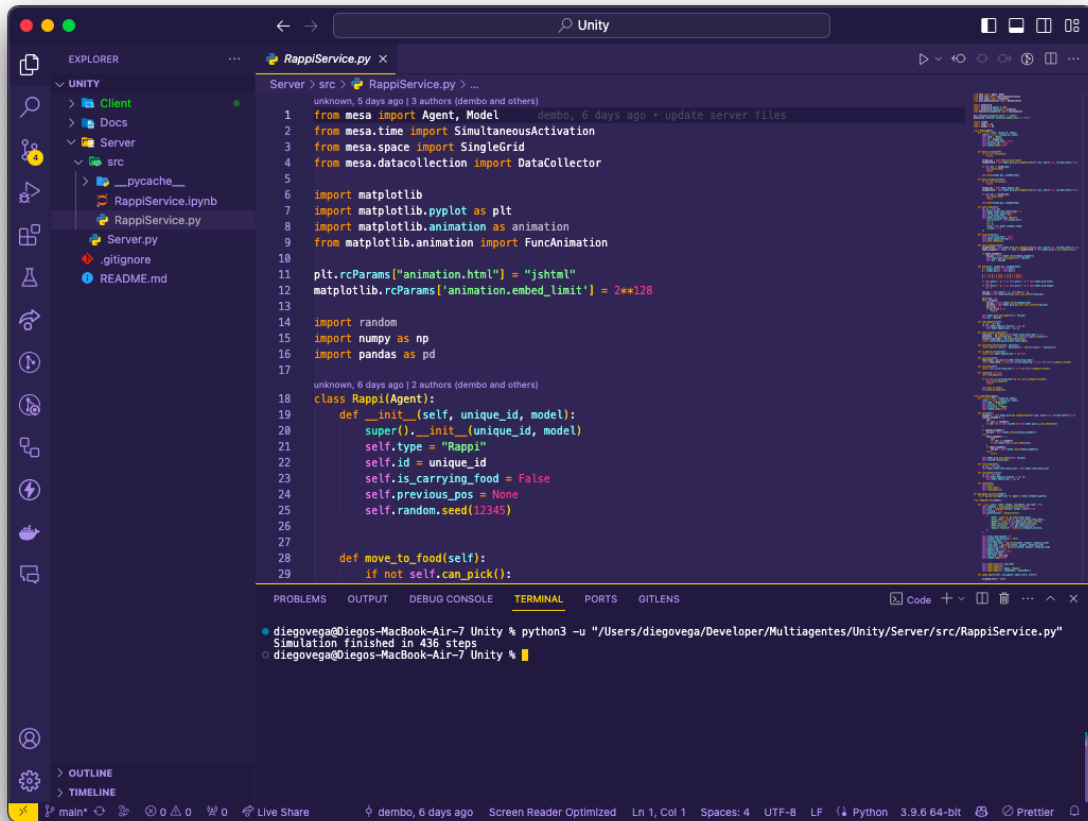


2. Clona el repositorio en tu computadora.
  - a. Abre la terminal de tu computadora.
  - b. Escribe “git clone <https://github.com/dvegaa20/Unity>” en la carpeta deseada.
  - c. Permite que se clonen los archivos.

3. Una vez clonado el repositorio, desde VS Code, dirígete al directorio donde se encuentra “RappiService.py” y “Server.py”.



4. Ejecutaremos nuestro archivo llamado “RappiService.py”, esto ejecutará la simulación de multiagentes que mandará los datos esenciales para la ejecución de nuestro proyecto en Unity.

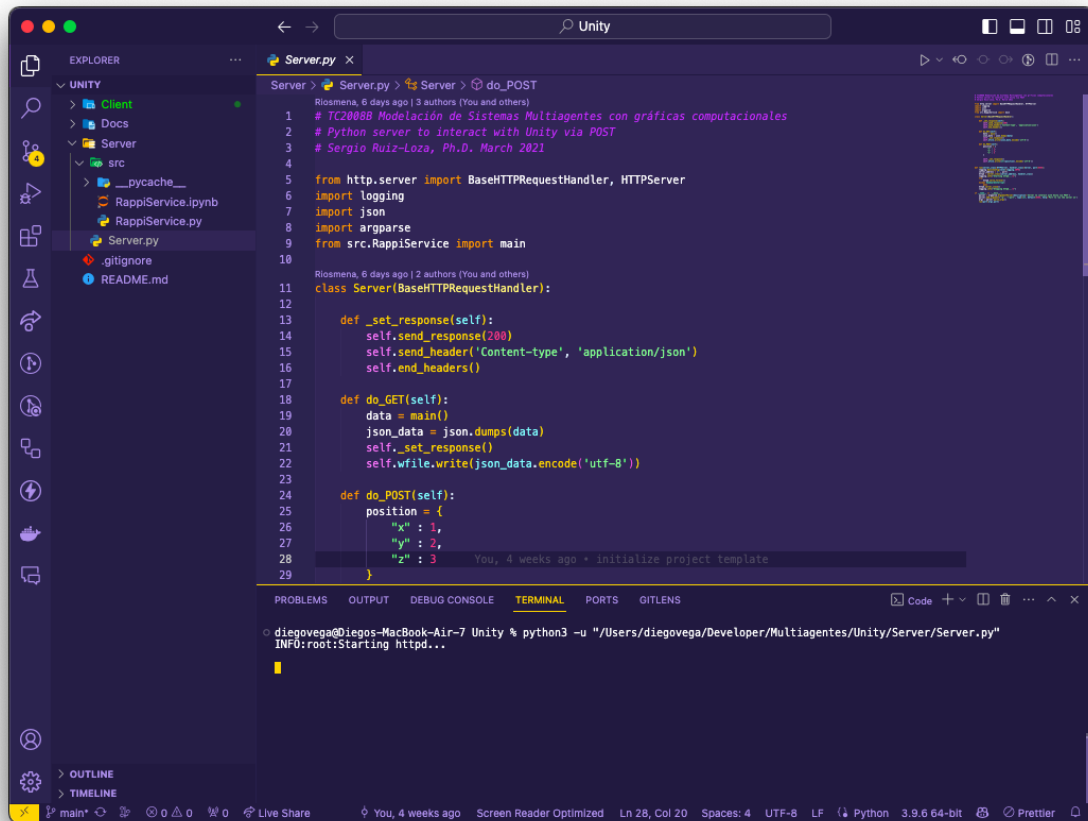


The screenshot shows a code editor with a dark theme. The Explorer panel on the left shows a project structure for 'UNITY' with folders 'Client', 'Docs', 'Server', and 'src'. The 'Server' folder is expanded, showing 'RappiService.py' and 'Server.py'. The main editor window displays the 'RappiService.py' file, which contains Python code for a multi-agent simulation using Mesa and Matplotlib. The code includes imports for Mesa, Matplotlib, and other libraries, and defines a 'Rappi' class with methods for initialization and movement. The terminal window at the bottom shows the command 'python3 -u "/Users/diegovega/Developer/Multiagentes/Unity/Server/src/RappiService.py"' being executed, and the output 'Simulation finished in 436 steps'.

```
Server > src > RappiService.py > ...
1 from mesa import Agent, Model
2 from mesa.time import SimultaneousActivation
3 from mesa.space import SingleGrid
4 from mesa.datacollection import DataCollector
5
6 import matplotlib
7 import matplotlib.pyplot as plt
8 import matplotlib.animation as animation
9 from matplotlib.animation import FuncAnimation
10
11 plt.rcParams["animation.html"] = "jshtml"
12 matplotlib.rcParams['animation.embed_limit'] = 2**128
13
14 import random
15 import numpy as np
16 import pandas as pd
17
18 unknown, 6 days ago | 2 authors (dembo and others)
19 class Rappi(Agent):
20     def __init__(self, unique_id, model):
21         super().__init__(unique_id, model)
22         self.type = "Rappi"
23         self.id = unique_id
24         self.is_carrying_food = False
25         self.previous_pos = None
26         self.random.seed(12345)
27
28     def move_to_food(self):
29         if not self.can_pick():
```

diegovega@Diegos-MacBook-Air-7 Unity % python3 -u "/Users/diegovega/Developer/Multiagentes/Unity/Server/src/RappiService.py"  
Simulation finished in 436 steps  
diegovega@Diegos-MacBook-Air-7 Unity %

5. Escribe y ejecuta “py Server.py”, esto encenderá el servidor por el cual podremos mandar información hacia nuestro proyecto en Unity.

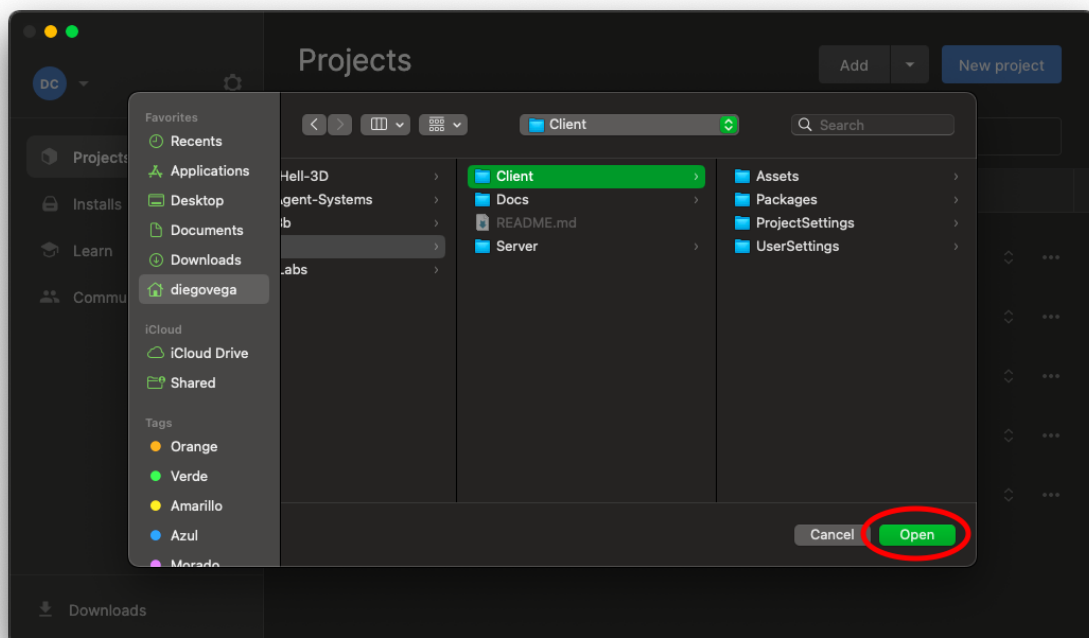
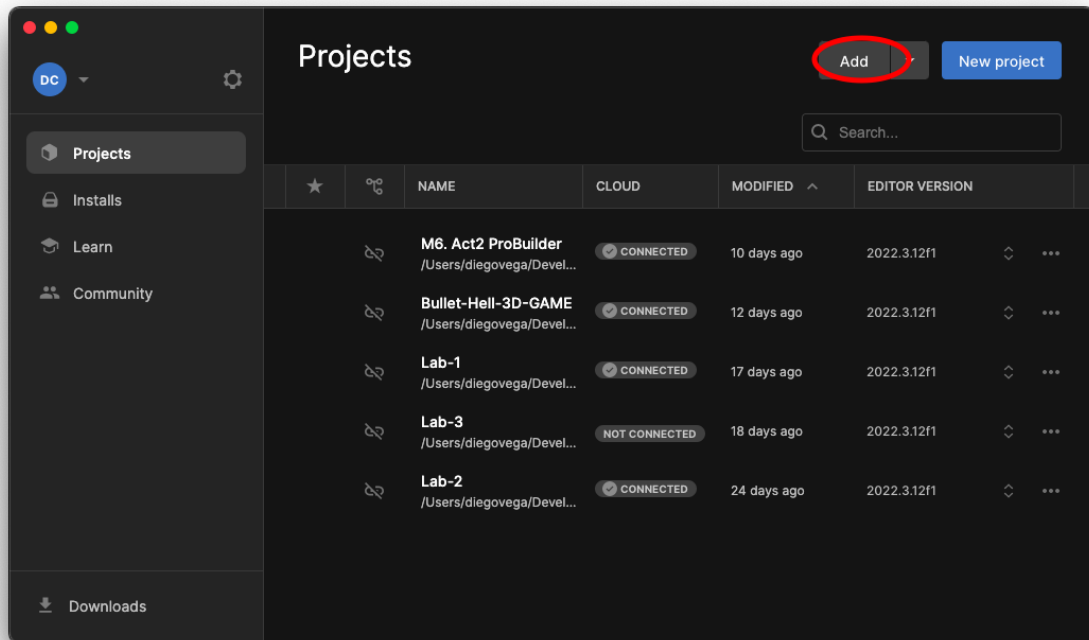


The screenshot shows a code editor with a dark theme. On the left, the Explorer panel shows a project structure for 'UNITY' with folders 'Client', 'Docs', and 'Server'. The 'Server' folder is expanded, showing files like 'src', 'RappiService.ipynb', 'RappiService.py', and 'Server.py'. The 'Server.py' file is selected and its content is displayed in the main editor. The code is a Python script that sets up a simple HTTP server using the 'http.server' module. It defines a 'Server' class that inherits from 'BaseHTTPRequestHandler' and implements methods for handling GET and POST requests. The POST method writes the received data to a file named 'data.json'. The script also includes imports for 'logging', 'json', and 'argparse'. At the bottom, the Terminal panel shows the command 'python3 -u "/Users/diegovega/Developer/Multiagentes/Unity/Server/Server.py"' being executed, and the output 'INFO:root:Starting httpd...'.

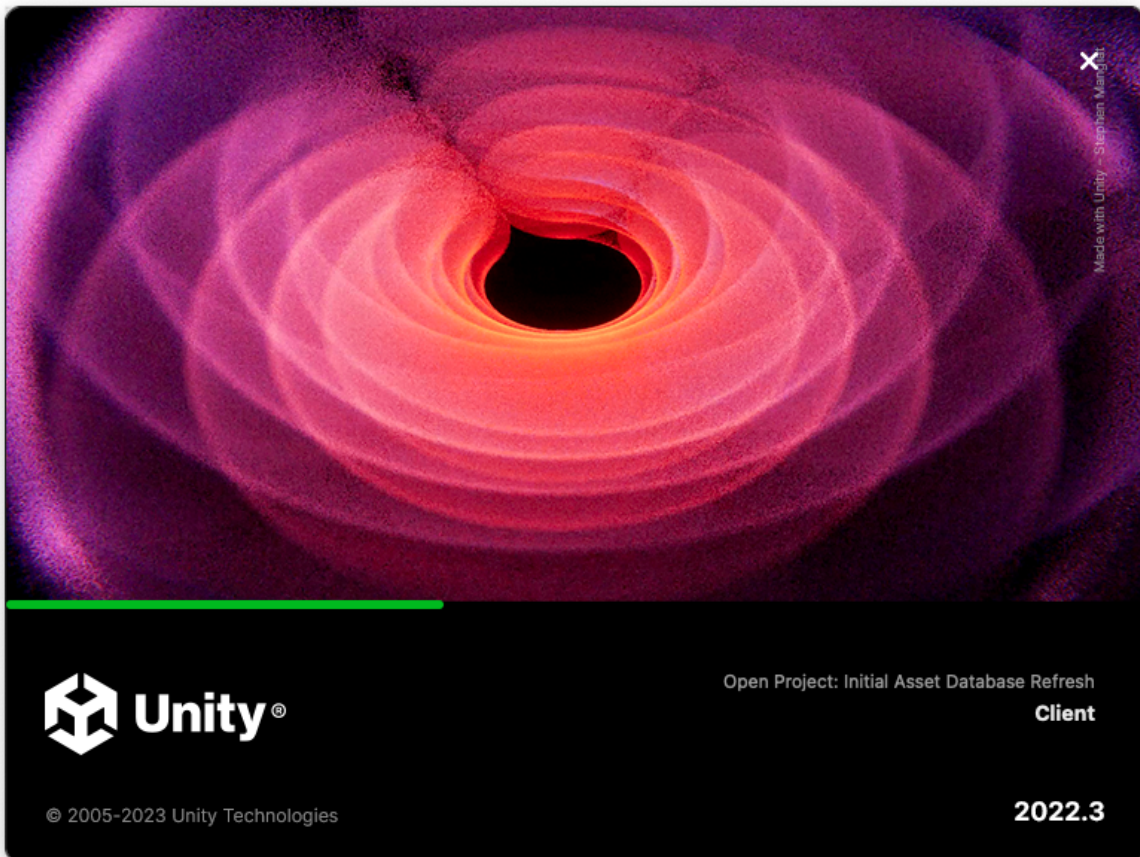
```
1 # TC20088 Modelación de Sistemas Multiagentes con gráficas computacionales
2 # Python server to interact with Unity via POST
3 # Sergio Ruiz-Loza, Ph.D. March 2021
4
5 from http.server import BaseHTTPRequestHandler, HTTPServer
6 import logging
7 import json
8 import argparse
9 from src.RappiService import main
10
11 class Server(BaseHTTPRequestHandler):
12
13     def _set_response(self):
14         self.send_response(200)
15         self.send_header('Content-type', 'application/json')
16         self.end_headers()
17
18     def do_GET(self):
19         data = main()
20         json_data = json.dumps(data)
21         self._set_response()
22         self.wfile.write(json_data.encode('utf-8'))
23
24     def do_POST(self):
25         position = {
26             "x" : 1,
27             "y" : 2,
28             "z" : 3
29         }
```

diegovega@Diegos-MacBook-Air-7 Unity % python3 -u "/Users/diegovega/Developer/Multiagentes/Unity/Server/Server.py"  
INFO:root:Starting httpd...

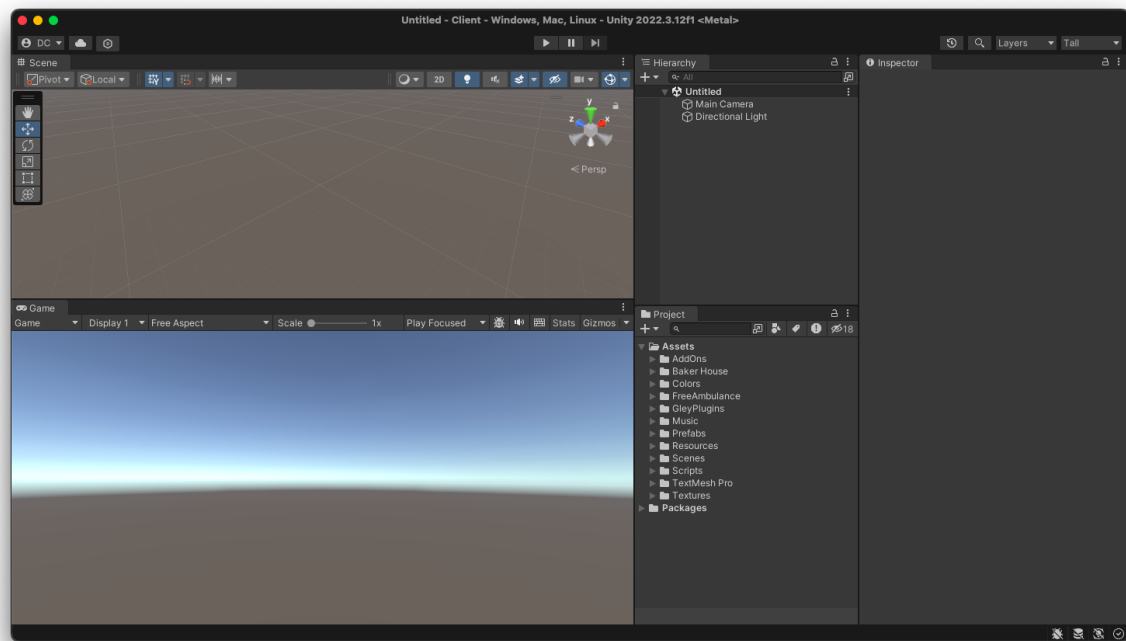
6. Ahora, abriremos Unity Desktop, aquí haremos click en “Add new” y abriremos la carpeta “Client” que generará nuestro proyecto en Unity



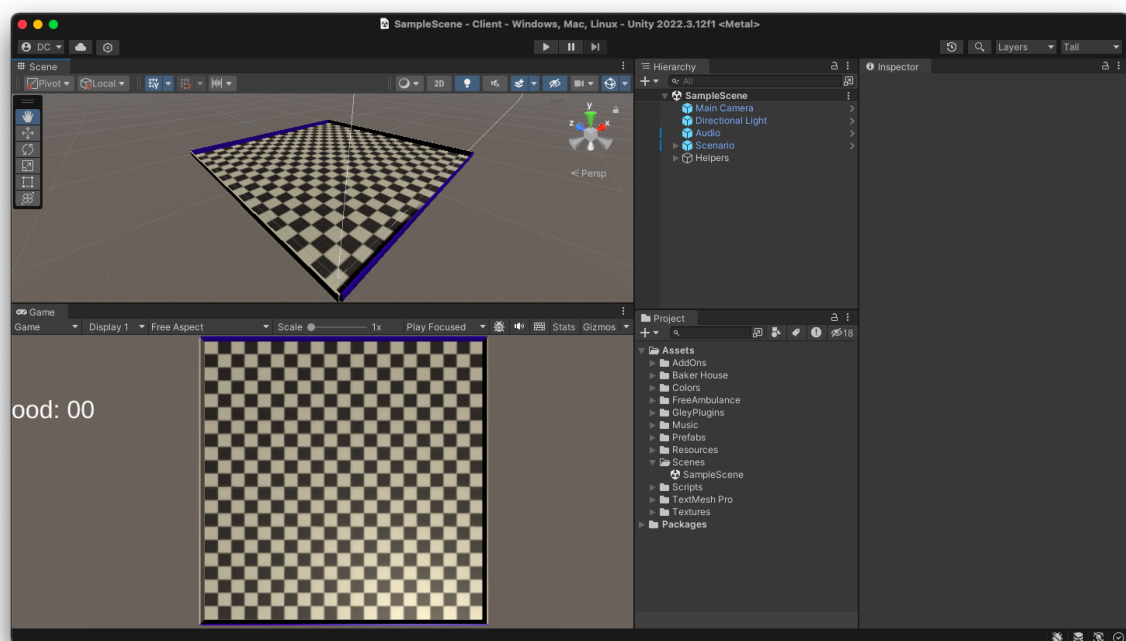
7. Esperamos un momento a que abra el proyecto



8. Una vez finalizado el proceso, se mostrará un proyecto vacío de Unity. Lo que simplemente tenemos que hacer es arrastrar la “SampleScene” que se encuentra en la carpeta de “Scenes” en la sección de “Project”



9. Ejecutamos el proyecto y se mostrará correctamente la visualización completa del mismo.





**Vídeo describiendo el proceso anterior**

<https://drive.google.com/file/d/1k6gzOgfPjAT9-di748XMv3SdvndrgeFI/view?usp=sharing>