# BÁO CÁO DEMO: GAMA → Unity qua WebSocket

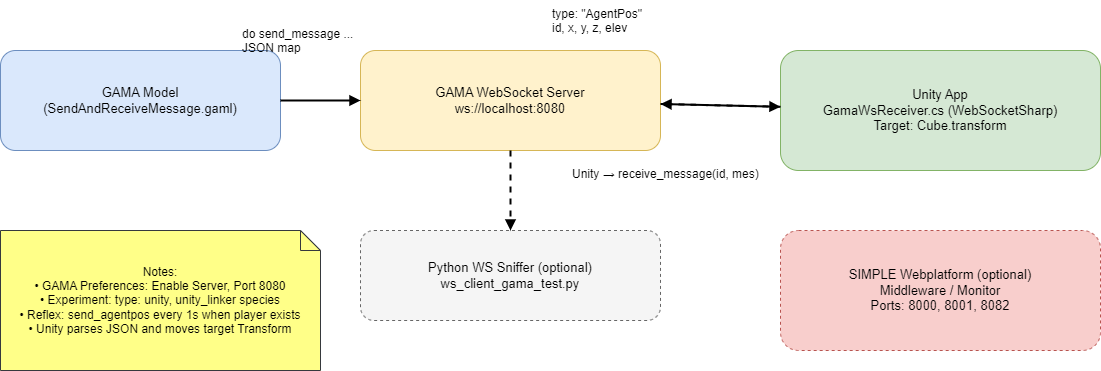
## 1) Mục tiêu

Hiểu rõ cơ chế GAMA gửi và Unity nhận dữ liệu (ưu tiên qua WebSocket/middleware), thực hiện thử nghiệm truyền gói toạ độ/độ cao và quan sát cập nhật trong Unity. Kết quả gồm sơ đồ Data Flow và gói demo chạy được.

## 2) Kiến trúc / Data Flow

Luồng dữ liệu tối giản được dùng trong demo:

GAMA (reflex gửi map) → GAMA WebSocket Server (cổng 8080) → Unity (GamaWsReceiver.cs) → Cập nhật Transform



## 3) Mã GAML trong model

Ghi vào file Send Receive Messages.gaml (hoặc model riêng).

model SendAndReceiveMessage  
  
species unity\_linker parent: abstract\_unity\_linker {  
 string player\_species <- string(unity\_player);  
 bool do\_send\_world <- false;  
  
 reflex ensure\_player when: (cycle = 0) and empty(unity\_player) {  
 do create\_player("py1");  
 }  
  
 reflex send\_agentpos when: every(1.0) and not empty(unity\_player) {  
 map msg <- ["type"::"AgentPos","id"::"a1","x"::rnd(100),"y"::rnd(100),"z"::rnd(10),"elev"::rnd(10)];  
 do send\_message players: (unity\_player as list) mes: msg;  
 }  
  
 action receive\_message (string id, string mes) {  
 write "Player " + id + " send the message: " + mes;  
 }  
}  
  
species unity\_player parent: abstract\_unity\_player;  
  
experiment vr\_xp type: unity {  
 float minimum\_cycle\_duration <- 0.05;  
 string unity\_linker\_species <- string(unity\_linker);  
}

## 4) Client Python kiểm tra WebSocket

Lưu thành ws\_client\_gama\_test.py và chạy để xem gói JSON từ GAMA:

# pip install websockets  
import asyncio, websockets  
URI = "ws://localhost:8080/"  
async def run():  
 async with websockets.connect(URI, max\_size=None) as ws:  
 print("Connected")  
 while True:  
 msg = await ws.recv()  
 for part in str(msg).split("|||"):  
 s = part.strip()  
 if s:  
 print("RECV:", s)  
asyncio.run(run())

## 5) Unity: GamaWsReceiver.cs

Thêm vào Unity tại Assets/Scripts/GamaWsReceiver.cs, add component lên một GameObject, set Target tới Cube.

using UnityEngine;  
using System;  
using System.Threading.Tasks;  
using System.Net.WebSockets;  
using System.Threading;  
using System.Text;  
using Newtonsoft.Json.Linq;  
  
public class GamaWsReceiver : MonoBehaviour  
{  
 public string uri = "ws://localhost:8080/";  
 public Transform target;  
 public bool swapZY = true;  
  
 ClientWebSocket \_ws;  
  
 async void Start() { await StartWs(); }  
  
 async Task StartWs()  
 {  
 \_ws = new ClientWebSocket();  
 await \_ws.ConnectAsync(new Uri(uri), CancellationToken.None);  
 Debug.Log("[GamaWsReceiver] Connected " + uri);  
 \_ = RecvLoop();  
 }  
  
 async Task RecvLoop()  
 {  
 var buf = new byte[1 << 16];  
 while (\_ws != null && \_ws.State == WebSocketState.Open)  
 {  
 var res = await \_ws.ReceiveAsync(buf, CancellationToken.None);  
 if (res.MessageType == WebSocketMessageType.Close) break;  
 var s = Encoding.UTF8.GetString(buf, 0, res.Count);  
  
 foreach (var raw in s.Split(new string[] {"|||"}, StringSplitOptions.RemoveEmptyEntries))  
 {  
 try  
 {  
 // Tầng ngoài từ GAMA  
 var obj = JObject.Parse(raw);  
 var content = obj["content"];  
  
 // Hai trường hợp: content là object { "message": "...json..." } hoặc chính là string json  
 string payloadJson = null;  
 if (content != null && content["message"] != null)  
 payloadJson = content["message"]!.ToString();  
 else if (content != null && content.Type == JTokenType.String)  
 payloadJson = content.ToString();  
  
 if (string.IsNullOrEmpty(payloadJson)) continue;  
  
 // Chuỗi JSON chứa "contents":[{"contents":{...}}]  
 var payload = JObject.Parse(payloadJson);  
 var contentsArr = payload["contents"] as JArray;  
 if (contentsArr == null || contentsArr.Count == 0) continue;  
  
 var firstContents = contentsArr[0]["contents"] as JObject;  
 if (firstContents == null) continue;  
  
 if ((string?)firstContents["type"] == "AgentPos" && target != null)  
 {  
 float x = firstContents["x"]?.Value<float>() ?? 0f;  
 float y = firstContents["y"]?.Value<float>() ?? 0f;  
 float z = firstContents["z"]?.Value<float>() ?? 0f;  
 if (swapZY) target.position = new Vector3(x, z, y);  
 else target.position = new Vector3(x, y, z);  
 }  
 }  
 catch (Exception e) { Debug.LogWarning(e.Message); }  
 }  
 }  
 }  
  
 void OnDestroy()  
 {  
 try { \_ws?.Abort(); \_ws?.Dispose(); } catch { }  
 }  
}

## 6) Quy trình chạy demo

• GAMA: Preferences → Network: bật “Enable GAMA Server”, Port = 8080.

• Mở model, chạy experiment vr\_xp. Quan sát Console thấy “ensure\_player: created py1”.

• Python (tuỳ chọn): chạy ws\_client\_gama\_test.py để thấy gói JSON “SimulationOutput…AgentPos…”.

• Unity: Play, WS client kết nối ws://localhost:8080/, Cube di chuyển theo dữ liệu nhận.

## 7) Ghi chú

- Nếu Unity không di chuyển: kiểm tra xem GAMA đang phát “SimulationOutput…AgentPos…”.

- Nếu console Unity chỉ có “Connected …” mà không có AgentPos: xem lại reflex send\_agentpos trong species unity\_linker.

- Firewall có thể chặn 8080.

- Nếu dùng SIMPLE Webplatform, cổng WebSocket có thể khác .env; demo này dùng trực tiếp GAMA server 8080.