**QUY TRÌNH TỪ VRExperiment.java → GAMAGeometryLoader.cs (Terrain)**

# Tổng quan ngắn

Mục tiêu: chỉ rõ các BƯỚC và ĐIỂM GỌI/GỬI/NHẬN từ VRExperiment.java (init/step) qua AbstractUnityLinker.java, đi trên WebSocket, đến Unity và chốt ở GAMAGeometryLoader.cs → ManageTerrain. Không giải thích nội bộ thuật toán—chỉ nêu hàm, file, và trật tự.

# Bước 1 — VRExperiment.java: init()/step() gọi prim\*

**File: src/gaml/extension/unity/species/VRExperiment.java**

// init(IScope scope)  
unityLinker.primSentInitData(scope);  
unityLinker.primSentWorld(scope);  
  
// step/tick  
unityLinker.primMessageGeoms(scope);  
unityLinker.primUpdateTerrain(scope); // QUAN TRỌNG CHO TERRAIN  
unityLinker.primSendParameters(scope); // nếu dùng

Ý nghĩa: VRExperiment là nơi điều phối thời điểm gọi các prim\* sang lớp Linker.

# Bước 2 — AbstractUnityLinker.java: prim\* đóng gói JSON

**File: src/gaml/extension/unity/species/AbstractUnityLinker.java**

Các hàm liên quan: primSentInitData(IScope), primSentWorld(IScope), primMessageGeoms(IScope), primUpdateTerrain(IScope), ...

// Trong từng prim\*, ví dụ terrain  
JSONObject message = buildTerrainPayload(scope); // JSON Builder (chứa type='terrain', size, scale, heights...)  
String mesStr = SerialisationOperators.toJson(scope, message, false);

Ý nghĩa: JSON Builder sống ngay trong/được gọi bởi các prim\*, tạo chuỗi JSON hợp lệ.

# Bước 3 — AbstractUnityLinker.java: HÀM GỬI CHUNG

**Gọi chốt gửi (mọi gói):**

PlatformAgent pa = GAMA.getPlatformAgent();  
pa.sendMessage(scope, ConstantExpressionDescription.createNoCache(mesStr));

Ý nghĩa: mọi chuỗi JSON (mesStr) đều đẩy qua PlatformAgent.sendMessage(...). Đây là điểm gửi chung duy nhất ra kênh socket.

# Bước 4 — WebSocket Server (GAMA) → đẩy frame TEXT

**Thành phần: PlatformAgent/WebSocket server phía GAMA**

// Tầng dưới (khái niệm)  
session.getAsyncRemote().sendText(mesStr); // WebSocket TEXT frame

Ý nghĩa: JSON được chuyển thành WebSocket text frame trên ws://<ip>:<port>.

# Bước 5 — Unity: WebSocketConnector.cs nhận sự kiện

**File: Assets/Scripts/Gama Provider/Connection/WebSocketConnector.cs**

// Start(): mở kết nối  
socket = new WebSocket("ws://" + host + ":" + port + "/");  
await socket.Connect();  
  
// Đăng ký nhận  
socket.OnMessage += (byte[] msg) => {  
 string s = Encoding.UTF8.GetString(msg);  
 ManageMessage(s);  
};  
  
// Bơm sự kiện về main thread (Desktop)  
void Update() {  
#if !UNITY\_WEBGL || UNITY\_EDITOR  
 socket.DispatchMessageQueue();  
#endif  
}

Ý nghĩa: OnMessage là ĐIỂM NHẬN CHUNG ở Unity; Update() đảm bảo callback được kích hoạt đều.

# Bước 6 — Unity: ConnectionManager.cs điều phối theo type

**File: Assets/Scripts/Gama Provider/Connection/ConnectionManager.cs**

protected override void ManageMessage(string raw) {  
 var t = JsonUtility.FromJson<EnvelopeType>(raw)?.type;  
 switch (t) {  
 case "json\_state": HandleState(raw); break;  
 case "init": sim.ManageInit(raw); break;  
 case "json\_simulation": sim.ManageSimulation(raw); break;  
 case "terrain": geo.ManageTerrain(raw); break; // QUAN TRỌNG CHO TERRAIN  
 case "json\_water": sim.ManageWaterData(raw); break;  
 default: Debug.LogWarning("Unknown type: " + t); break;  
 }  
}

Ý nghĩa: ConnectionManager là nơi tách nhánh theo 'type'. Đối với terrain, route → GAMAGeometryLoader.ManageTerrain.

# Bước 7 — ĐÍCH TERRAIN: GAMAGeometryLoader.cs → ManageTerrain

**File: Assets/Scripts/Utils/GAMAGeometryLoader.cs**

public void ManageTerrain(string raw) {  
 // Parse payload { type:'terrain', size:[W,H], scale:[SX,SY,SZ], heights:[...] }  
 // Áp dụng vào TerrainData: SetHeightsDelayLOD(...); terrain.Flush();  
}

Ý nghĩa: đây là điểm kết thúc chuỗi đối với gói 'terrain'—dữ liệu DEM/heightmap được áp dụng vào Terrain trong Scene.

# Ánh xạ nhanh prim\* → type → điểm nhận

|  |  |  |  |
| --- | --- | --- | --- |
| Gọi từ VRExperiment | Prim bên AbstractUnityLinker | type JSON | Unity nhận ở đâu |
| init() | primSentInitData(scope) | init | ConnectionManager → sim.ManageInit |
| init() | primSentWorld(scope) | json\_simulation (static) | ConnectionManager → sim.ManageSimulation → ManageStaticData |
| step() | primMessageGeoms(scope) | json\_simulation (dynamic) | ConnectionManager → sim.ManageSimulation → ManageDynamicData |
| step() | primUpdateTerrain(scope) | terrain | ConnectionManager → geo.ManageTerrain |
| step() | primSendParameters(scope) | json\_state/params | ConnectionManager → HandleState/áp dụng tham số |

# Điểm mấu chốt để show

**• HÀM GỬI CHUNG: AbstractUnityLinker.java → PlatformAgent.sendMessage(scope, ConstantExpressionDescription.createNoCache(json))**

**• ĐIỂM NHẬN CHUNG: WebSocketConnector.cs → OnMessage(byte[]) → ManageMessage(string)**

**• TERRAIN ĐÍCH: ConnectionManager.cs → nhánh 'terrain' → GAMAGeometryLoader.cs → ManageTerrain(raw)  
  
IScope scope (GAMA)**

Định nghĩa:

IScope là giao diện ngữ cảnh thực thi của GAMA, đại diện cho toàn bộ bối cảnh (simulation, agent, thời gian, biến) tại thời điểm một hành động/operator chạy.

Vai trò chính:

* Cấp quyền truy cập Simulation hiện tại (scope.getSimulation()).
* Cấp quyền truy cập Agent đang thực thi (scope.getAgent()).
* Truy cập biến môi trường, tham số, thời gian tick, scheduler.
* Đảm bảo operator/primitive hoạt động đúng trong runtime và thread-safety của GAMA.

Tại sao xuất hiện trong AbstractUnityLinker.prim\*(IScope scope):

* Để đọc dữ liệu mô phỏng cần đóng gói JSON (agents, geometries, terrain, time).
* Để log, truy cập tham số cấu hình gửi (precision/world/scale).
* Để truyền đúng ngữ cảnh khi gọi PlatformAgent.sendMessage(...).

Liên hệ với hàm gửi chung:

AbstractUnityLinker.\* → SerialisationOperators.toJson(scope, message, false) → PlatformAgent.sendMessage(scope, ConstantExpressionDescription.createNoCache(json)) — scope được dùng xuyên suốt để định nghĩa và gửi đúng bối cảnh mô phỏng.