

**ROBOEXHIBIT**



**UNIVERSITÀ  
DEGLI STUDI  
DI PALERMO**

## **UNIVERSITÀ DEGLI STUDI DI PALERMO**

**Dipartimento di Ingegneria**

**Corso di Laurea Magistrale in Ingegneria Informatica**

## **ROBOEXHIBIT**

**PROGETTO DI:**

**MOSCA LUCREZIA**

**PICONE ALESSANDRO**

**SPEDITO ANTONIO**

**DOCENTI:**

**PROF.SSA SEIDITA VALERIA**

**PROF. CHELLA ANTONIO**

**A.A. 2024/2025**

## Sommario

<b>Introduzione.....</b>	<b>4</b>
<b>Applicazione Reale .....</b>	<b>4</b>
<b>Ambiente di sviluppo .....</b>	<b>5</b>
<b>Modellazione del problema .....</b>	<b>6</b>
<b>Agenti.....</b>	<b>7</b>
<i>Bigliettaio .....</i>	8
<i>Guida del Museo .....</i>	8
<b>Ambiente .....</b>	<b>9</b>
<i>Struttura e Organizzazione.....</i>	10
<b>INTELLIGENZAARTIFICIALE 2 .....</b>	<b>14</b>
<b>Ontologia .....</b>	<b>15</b>
<i>Classi e Sottoclassi .....</i>	15
<i>Object Properties .....</i>	18
<i>Data Properties .....</i>	20
<i>Individuals .....</i>	22
<i>Diagramma delle classi.....</i>	29
<b>Processo di Recupero dei Dati della Base di Conoscenza e Generazione della Risposta.....</b>	<b>30</b>
<b>ROBOTICA .....</b>	<b>37</b>
<b>Struttura del progetto .....</b>	<b>37</b>
<b>Modellazione dell'ambiente.....</b>	<b>39</b>
<i>Realizzazione dell'Ambiente .....</i>	39
<i>Struttura del Museo .....</i>	40
<b>Modellazione degli agenti .....</b>	<b>41</b>
<i>Strategia Implementativa per gli Agenti.....</i>	42
<b>Hardware .....</b>	<b>52</b>
<b>Movimento degli agenti.....</b>	<b>54</b>
<i>NavMesh.....</i>	55
<i>Ricerca A* .....</i>	55
<i>Superficie Percorribile .....</i>	56
<i>Gestione degli Oggetti nella Scena .....</i>	56
<i>Filtro di Kalman .....</i>	57
<b>Pianificazione dell'itinerario .....</b>	<b>58</b>
<i>Ripianificazione dinamica dell'itinerario .....</i>	59
<b>Modellazione dell'utente.....</b>	<b>60</b>
<b>Modellazione interfaccia utente .....</b>	<b>61</b>
<i>Choose Ticket UI .....</i>	61
<i>Block Enter UI.....</i>	62
<i>Question UI .....</i>	62
<i>Interaction UI .....</i>	63
<b>Finestre di dialogo .....</b>	<b>63</b>
<b>Conclusioni.....</b>	<b>65</b>

## Introduzione

Il progetto rappresenta un'ambiente immersivo in cui agenti intelligenti e tecnologie avanzate siano integrati all'interno di un museo per creare un'esperienza culturale unica, immersiva e personalizzata. Questo sistema non si limita a fornire informazioni, ma trasforma ogni visita in un viaggio interattivo e dinamico, adattandosi continuamente alle esigenze e agli interessi dei visitatori. Gli agenti intelligenti presenti nel museo agiscono come assistenti virtuali e fisici, capaci di comprendere le preferenze culturali. Gli agenti monitorano in tempo reale le interazioni e il flusso dei visitatori, raccogliendo dati per ottimizzare i percorsi proposti. Ad esempio, un tour può essere modellato per evidenziare opere d'arte di un particolare periodo storico o per enfatizzare aspetti educativi nella stanza interattiva. Un esempio pratico potrebbe riguardare un visitatore con una forte predilezione per l'arte rinascimentale: il sistema organizza automaticamente un percorso tematico che include le opere principali. Al contrario, un gruppo scolastico potrebbe essere guidato verso attività dinamiche nella stanza interattiva dove i ragazzi possono partecipare a simulazioni o giochi educativi. La piattaforma di gestione che coordina questi agenti è centralizzata per garantire una visione d'insieme del museo, ma ogni agente dispone di un'autonomia locale sufficiente a rispondere rapidamente alle esigenze immediate dei visitatori. I dati raccolti confluiscono in un Knowledge Graph, una base di conoscenza che integra caratteristiche e preferenze dei visitatori con informazioni dettagliate sul museo, come la posizione e il contenuto delle opere, le modalità del tour e le tipologie di stanze. Questo strumento permette una pianificazione dinamica e ottimizzata, adattando l'esperienza al contesto in tempo reale. La principale sfida tecnica è sviluppare una base di conoscenza robusta, capace di integrare dati eterogenei e di alimentare un sistema decisionale che bilanci in modo efficace le preferenze personali con la valorizzazione del patrimonio artistico. La collaborazione tra visitatori e agenti intelligenti diventa il cuore pulsante del museo interattivo, offrendo un'esperienza arricchente, stimolante e profondamente personalizzata.

## Applicazione Reale

Questo progetto mira a trasformare l'esperienza museale tradizionale, integrando tecnologie avanzate e agenti intelligenti per creare un ambiente interattivo, personalizzato e inclusivo. L'obiettivo è rendere la cultura più accessibile e coinvolgente, trasformando ogni visita in un'esperienza unica e adattiva. Gli agenti intelligenti monitorano in tempo reale il comportamento e le preferenze dei visitatori, ottimizzando i percorsi culturali e offrendo interazioni su misura. Un visitatore appassionato di arte rinascimentale, ad esempio, potrebbe essere guidato verso opere specifiche,

mentre un gruppo scolastico può partecipare a simulazioni o giochi educativi nella stanza interattiva. Questi musei possono avere un impatto positivo anche su persone con diverse problematiche, come disabilità motorie o cognitive. Tecnologie come la realtà aumentata e la realtà virtuale permettono di accedere ai contenuti senza barriere fisiche, mentre dispositivi interattivi personalizzabili possono adattarsi alle esigenze di persone con difficoltà sensoriali o linguistiche. Sono molti i musei innovativi presenti nei vari Paesi. Tra questi vi sono il *Museum of the Future* di Dubai, che combina intelligenza artificiale e realtà aumentata per un'esperienza multisensoriale, e il *National Museum of Emerging Science and Innovation (Miraikan)* di Tokyo, che utilizza robot interattivi per esplorare temi scientifici complessi. In Europa, il *Museo Nazionale della Scienza e della Tecnologia Leonardo da Vinci* di Milano presenta installazioni dinamiche e spazi dedicati alla realtà virtuale, rendendo la scienza e la tecnologia accessibili a visitatori di tutte le età e abilità. Questi approcci non solo valorizzano il patrimonio culturale, ma promuovono anche l'inclusione, garantendo che ogni visitatore possa vivere il museo in modo arricchente e stimolante, indipendentemente dalle proprie esigenze o capacità.

## Ambiente di sviluppo

Il progetto è stato sviluppato utilizzando il motore di gioco *Unity*, che offre un'integrazione avanzata di asset grafici, componenti logici e simulazioni fisiche.

- **Unity**
  - Unity costituisce il nucleo dello sviluppo del progetto grazie alla sua capacità di integrare diverse componenti e tecnologie:
    - **Gestione dei GameObject:** ogni elemento della scena è stato modellato come un GameObject, arricchito con componenti specifici per la gestione delle funzionalità logiche e fisiche.
    - **NavMesh:** utilizzato per generare superfici navigabili, consentendo un movimento fluido e realistico degli agenti intelligenti all'interno dell'ambiente.
    - **Interfaccia Grafica:** realizzata tramite il *Canvas System*, che gestisce gli elementi UI, come pulsanti e finestre di dialogo, migliorando l'interazione utente.
- **Visual Studio Code**

- Per la scrittura degli script in C# e Python è stato impiegato Visual Studio Code, configurato con estensioni dedicate al debugging e all'auto-completamento specifico per Unity.
- **Sintesi vocale di MacOS**
  - Gli strumenti di sintesi vocale nativi di MacOS sono stati utilizzati per abilitare l'interazione uditiva e verbale tra il robot e l'utente, migliorando l'accessibilità e l'esperienza immersiva.
- **Asset Store e Sketchfab**
  - Gli asset grafici utilizzati per la modellazione dell'ambiente del museo sono stati acquisiti principalmente dall'Asset Store di Unity e da Sketchfab, fornendo modelli 3D di alta qualità e riducendo i tempi di sviluppo.
- **Hugging Face Spaces**
  - Una macchina virtuale su Hugging Face Spaces è stata utilizzata per ospitare il modello di linguaggio *Llama 3.3 70B*, impiegato per generare query SPARQL e risposte in stile guida museale. Questa soluzione è stata adottata per garantire scalabilità ed efficienza nell'elaborazione delle richieste dell'utente.
- **Linguaggio Python e libreria RDFLib**
  - Il backend della macchina virtuale su *Hugging Face Spaces* è stato sviluppato in Python, utilizzando la libreria RDFLib per la gestione e l'interrogazione dell'ontologia del museo. Nonostante l'assenza di un database tradizionale, RDFLib consente di leggere e interrogare i dati strutturati in modo efficace.
- **Protégé**
  - Per la creazione e la gestione dell'ontologia del museo è stato utilizzato Protégé, uno strumento che ha facilitato la modellazione di classi, proprietà e relazioni tra le entità, migliorando l'organizzazione semantica delle informazioni.

## Modellazione del problema

La modellazione del problema è stata una fase fondamentale per la progettazione e la realizzazione del progetto. Il contesto di riferimento è quello di un museo interattivo, in cui l'agente guida accompagna i visitatori attraverso un'esperienza coinvolgente e personalizzata, offrendo risposte a domande libere e percorsi alternativi su richiesta. Per rappresentare adeguatamente la complessità del sistema, il problema è stato suddiviso in sotto-problemi specifici, ciascuno affrontato con un approccio mirato.

- **Navigazione e movimento degli agenti**
  - Gli agenti devono essere in grado di:
    - Pianificare un percorso iniziale e, in base alle domande dei visitatori, ripianificare dinamicamente l'itinerario.
    - Evitare ostacoli fisici e adattarsi in tempo reale ai cambiamenti nella disposizione del museo.
    - Seguire percorsi ottimizzati per raggiungere le opere d'arte richieste.
- **Interazione uomo-macchina**
  - L'interazione tra gli agenti e i visitatori deve essere fluida e intuitiva, consentendo una comunicazione efficace tramite comandi testuali. A tal fine, sono stati implementati:
    - Un sistema di sintesi che sfrutta gli strumenti nativi di MacOS per rendere l'interazione più naturale e accessibile.
    - Un backend avanzato capace di elaborare domande libere e fornire risposte in linguaggio naturale, migliorando la qualità dell'esperienza utente.
- **Rappresentazione della conoscenza**
  - Per gestire in modo strutturato le informazioni relative al museo, è stata sviluppata un'ontologia utilizzando Protégé che consente di rappresentare con precisione diverse entità, tra cui:
    - Stanze tematiche e interattive, con percorsi specifici e contenuti dinamici.
    - Opere d'arte dettagliate con attributi come autore, periodo storico, materiali e altre informazioni pertinenti.

## Agenti

Gli agenti intelligenti costituiscono il cuore del museo interattivo e sono progettati per garantire un'esperienza culturale personalizzata e coinvolgente. Sono realizzati per comprendere le preferenze culturali e personali dei visitatori e offrire suggerimenti mirati come percorsi tematici, attività complementari o mostre speciali. La loro capacità di adattarsi in tempo reale rende ogni visita unica. Si tratta di agenti orientati al raggiungimento di goal e progettati per agire in modo cooperativo. Il loro comportamento è guidato dalla necessità di pianificare percorsi efficaci e utilizzare le capacità percettive per interagire dinamicamente con l'ambiente circostante. Sia il ruolo del bigliettaio che quello della guida possono essere ricoperti da robot o persone, garantendo flessibilità e un'esperienza fluida per tutti i visitatori.

## *Bigliettaio*

Il bigliettaio, sia in forma robotica che umana, gestisce l'accoglienza dei visitatori, le prenotazioni e la vendita dei biglietti e svolge funzioni fondamentali per l'organizzazione del museo. L'intervento umano è previsto per gestire situazioni non standard, come richieste particolari o problemi tecnici. In entrambi i casi, il ruolo è progettato per essere intuitivo, efficace e inclusivo.

- **Funzioni del bigliettaio**

- **Gestione dei biglietti:** Vendita di biglietti, gestione delle prenotazioni e registrazione di gruppi.

- **Collaborazione umano-robotica**

- **Bigliettaio Robot:** Ideale per la gestione automatizzata di biglietti e informazioni standard. Può operare autonomamente per la maggior parte delle funzioni. In caso di problemi tecnici o richieste complesse, il personale umano interviene per supportare i visitatori.
- **Bigliettaio Umano:** Garantisce un'interazione più empatica e flessibile, particolarmente utile per visitatori con esigenze particolari o gruppi scolastici. Interviene nel momento in cui si dovessero verificare problemi tecnici.

## *Guida del Museo*

La guida del museo, disponibile in versione robotica o umana, è il punto di riferimento per i visitatori durante il tour. Il robot guida è ideale per attività standard, percorsi predefiniti e quelli generati a seconda delle esigenze dell'utente. Le guide umane possono offrire un'esperienza più empatica e flessibile, particolarmente utile per gruppi con esigenze specifiche o nel caso in cui si dovessero verificare problemi tecnici.

Il robot guida del museo dotato di intelligenza artificiale accompagna i visitatori durante i tour, spiegando le opere quando gli vengono poste delle domande e conducendoli a visitare la stanza interattiva. Grazie a sensori avanzati e a una connessione continua con il sistema centrale, il robot garantisce un monitoraggio di eventuali problematiche logistiche.

- **Funzioni della guida**

- **Conduzione del tour:** il robot guida segue un percorso programmato che attraversa le stanze tematiche e la stanza interattiva, adattandosi alle esigenze dei visitatori.

Fornisce spiegazioni dettagliate sulle opere esposte e può rispondere a domande di carattere generale. Il robot è in grado di:

- Evitare stanze già visitate, tranne il corridoio centrale, se necessario.
- **Gestione degli ostacoli e dei flussi:** il robot è equipaggiato con sensori per rilevare ostacoli fisici. In questi casi, propone percorsi alternativi per garantire una visita confortevole.
- **Monitoraggio continuo e aggiornamenti:** il robot guida è connesso al sistema centrale del museo, che elabora in tempo reale dati su percorsi e attività in corso. Questa integrazione consente al robot di fornire informazioni sempre aggiornate e di adattarsi dinamicamente alle condizioni del museo.
- **Collaborazione umano-robotica**
  - **Guida del museo robot:** Il robot guida del museo è un dispositivo altamente avanzato dotato di intelligenza artificiale e sensori avanzati, progettato per migliorare l'esperienza museale e per accompagnare i visitatori durante i tour, spiegando le opere.
  - **Guida del museo persona:** In caso di problemi tecnici o richieste specifiche non gestibili autonomamente dal robot, il personale umano interviene per garantire un'esperienza senza interruzioni. Questa collaborazione tra tecnologia e presenza umana assicura che ogni visitatore si senta supportato. La guida umana offre un approccio più empatico e flessibile, ideale per gruppi con esigenze specifiche, come scolaresche o visitatori con disabilità.

## Ambiente

L'ambiente, modellato per rappresentare un museo, possiede diverse caratteristiche fondamentali che ne influenzano la complessità e le modalità operative:

- **Accessibile:** l'agente può ottenere informazioni complete, accurate e aggiornate sullo stato dell'ambiente. Questa caratteristica facilita la progettazione e lo sviluppo degli agenti, poiché riduce l'incertezza legata alla mancanza di dati o alla necessità di fare inferenze basate su informazioni incomplete.
- **Deterministico:** ogni azione intrapresa ha un unico effetto garantito. Non esistono elementi di causalità o fattori imprevedibili che possano alterare l'esito delle azioni, consentendo così di prevedere con precisione gli stati futuri derivanti dalle decisioni prese.

- **Sequenziale:** ogni decisione presa dall'agente ha un impatto sulle decisioni future, rendendo fondamentale un'attenta pianificazione delle azioni per ottimizzare le prestazioni nel lungo periodo.
- **Multi-agente cooperativo:** gli agenti operano in collaborazione per raggiungere obiettivi comuni. In questo contesto, la comunicazione tra gli agenti gioca un ruolo cruciale per garantire il coordinamento e la condivisione delle informazioni necessarie.
- **Semi-dinamico:** sebbene lo stato dell'ambiente non cambi autonomamente nel tempo, la valutazione delle azioni dell'agente può variare, richiedendo un monitoraggio costante e un adattamento delle strategie operative.

## Struttura e Organizzazione

L'ambiente, modellato per rappresentare un museo, è suddiviso in stanze tematiche che ospitano collezioni di opere organizzate in base a periodi storici, temi culturali o movimenti artistici. Un corridoio centrale collega queste stanze a uno spazio interattivo, un'area dinamica in cui i visitatori possono partecipare a esperienze pratiche ed educative. Il museo è costituito dall'ingresso e da quattro stanze diverse per area tematica e sono: "Stanza Grecia", "Stanza Italia", "Stanza Van Gogh" e "Stanza Interattiva". Ciascuna di queste ospita una selezione di opere d'arte e sculture, offrendo ai visitatori l'opportunità di esplorare e ammirare le collezioni in un contesto coinvolgente e immersivo.

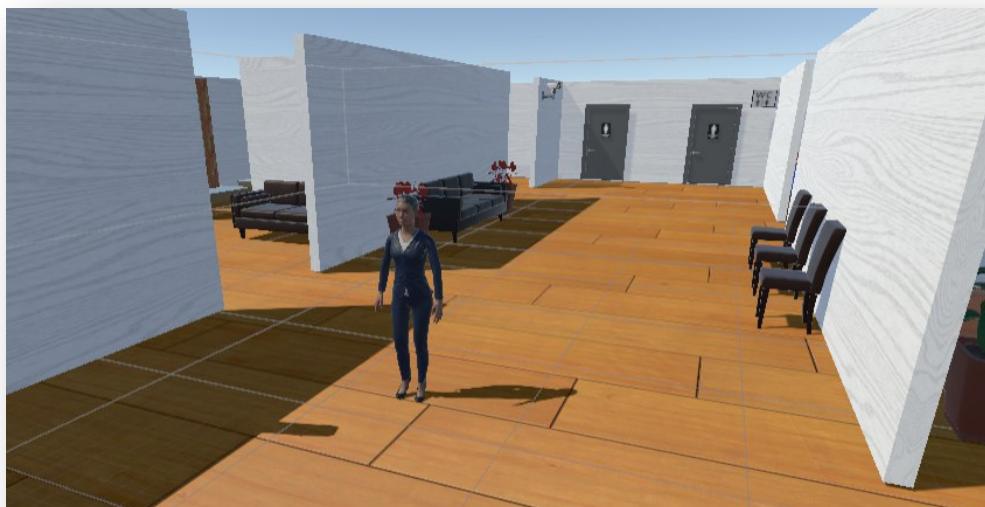
Nel museo sono presenti, inoltre, diversi tipi di ostacoli, suddivisi in due categorie:

- **Ostacoli statici:** includono le opere d'arte, i metal detector e l'arredamento del museo, elementi fissi che definiscono lo spazio e il percorso espositivo.
- **Ostacoli dinamici:** rappresentati dai visitatori stessi che si spostano all'interno delle sale, creando un ambiente in continuo movimento.

*Ingresso*



*Corridoio*



*Stanza Grecia*

La *Stanza Grecia* ospita opere come:

- **Discobolo**
- **Afrodite di Milo**
- **Vasi Greci**
- **Capitelli Ionici**



### *Stanza Italia*

La *Stanza Italia* ospita capolavori come:

- **Il David**
- **La Gioconda (Mona Lisa)**
- **Amore e Psiche**

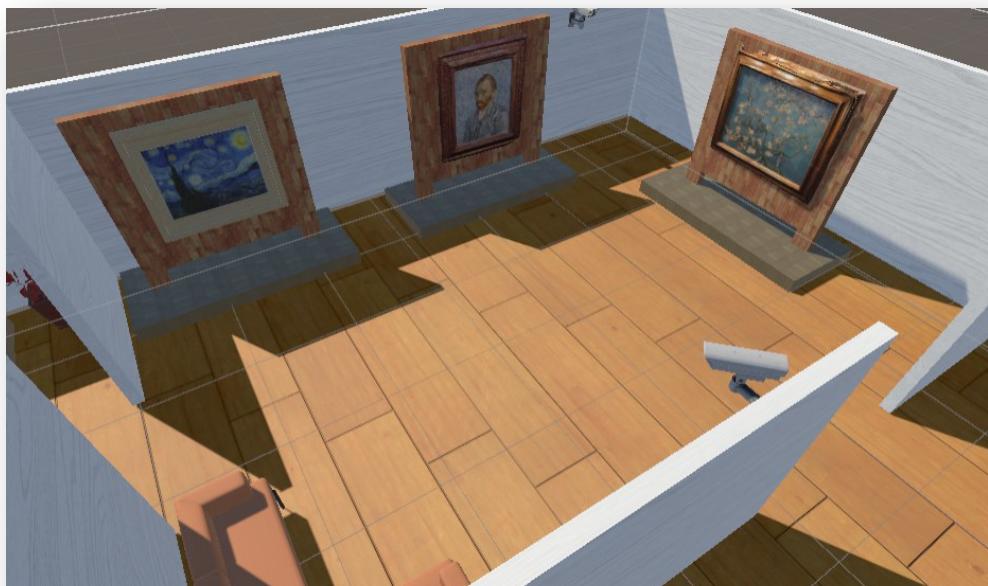


### *Stanza Van Gogh*

La *Stanza Van Gogh* ospita capolavori come:

- **La Notte Stellata**

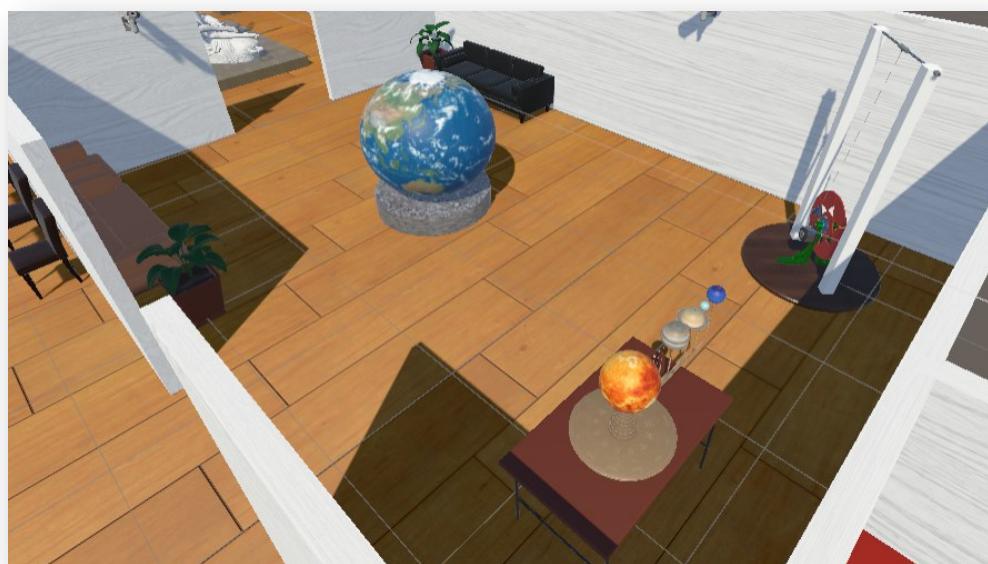
- **Autoritratto**
- **Ramo Di Mandorlo Fiorito**

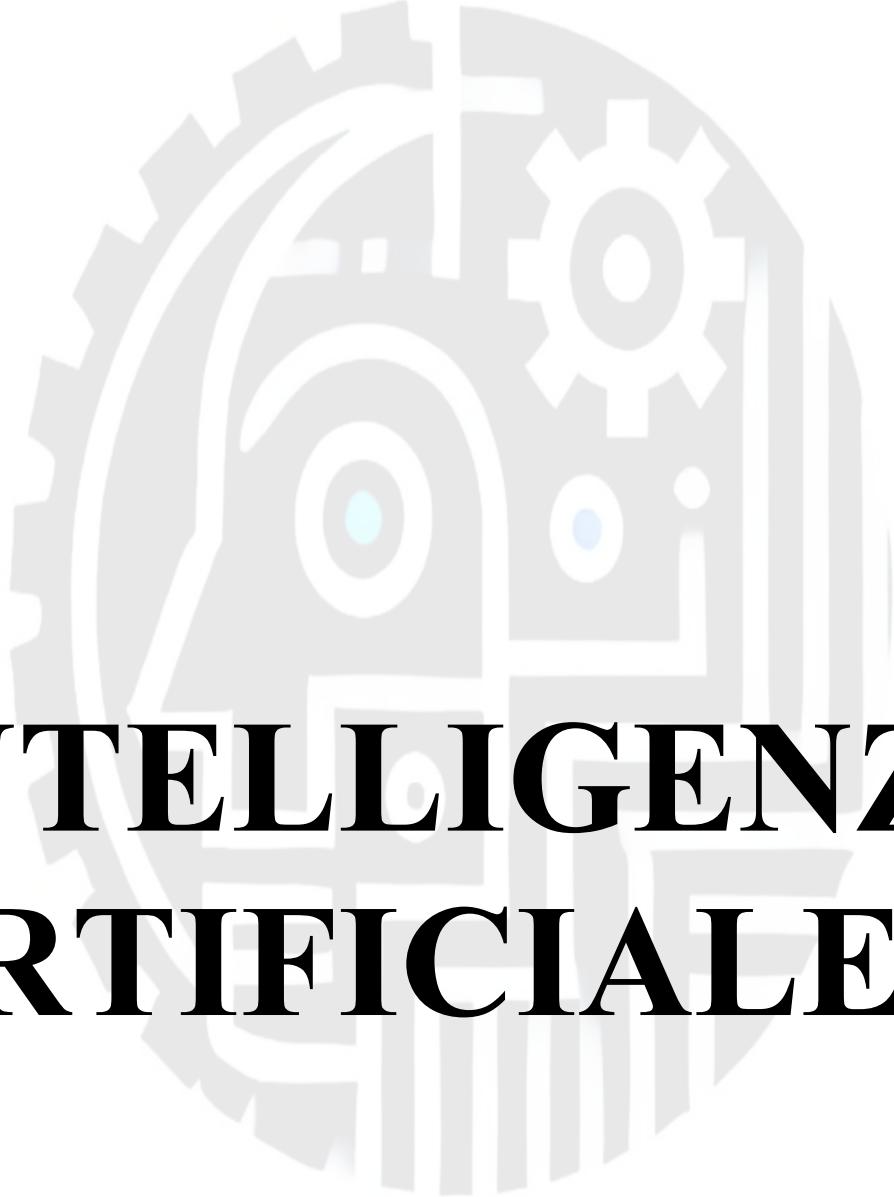


### *Stanza Interattiva*

La *Stanza Interattiva* ospita installazioni come:

- **Il Sistema Solare**
- **Pendolo**
- **Globo Terrestre**





# **INTELLIGENZA ARTIFICIALE 2**

## **ROBOEXHIBIT**

## Ontologia

Un'ontologia è una rappresentazione formale che descrive i concetti, le entità e le relazioni in un dominio specifico, insieme alle regole che ne governano il significato. L'obiettivo è quello di definire una struttura semantica che descrive cosa esiste in un dominio e come è organizzato. L'ontologia è progettata utilizzando il *W3C Web Ontology Language* (OWL), ovvero un linguaggio web semantico progettato per rappresentare una conoscenza ricca e complessa sulle cose, gruppi di cose e relazioni tra le cose. L'ontologia è stata realizzata mediante il software Protégé, ovvero un editor di ontologia e un sistema di gestione della conoscenza in grado di operare su tutte le piattaforme. L'ontologia rappresenta uno scenario museale con entità come stanze, manufatti, agenti e interazioni. Di seguito, vengono descritte le componenti principali: le classi, le proprietà e le relazioni.

- **Dominio:** Rappresentazione museale e del patrimonio culturale.
- **Scopo:** Modellare entità e relazioni all'interno di un museo, inclusi oggetti, visitatori, personale e mostre interattive.
- **Struttura:** L'ontologia utilizza classi, proprietà degli oggetti e proprietà dei dati per definire le relazioni semantiche e gli attributi delle entità.

## Classi e Sottoclassi

### 1. Museo

- Un'entità che rappresenta il museo come struttura fisica e l'istituzione nel suo insieme.
- **Object properties:**
  - contiene (relazione con le stanze).
  - organizza (relazione con gli eventi).
- **Data properties:**
  - **nomeMuseo:** il nome del museo.
  - **cittàMuseo:** la città in cui è situato il museo.
  - **statoMuseo:** lo Stato in cui è situato il museo.
  - **descrizioneMuseo:** una breve descrizione del museo.

### 2. Stanza

- Questa classe rappresenta uno spazio fisico all'interno del museo.
- **Sottoclassi:**
  - **StanzaTematica:** stanza focalizzata su un tema specifico.
  - **StanzaInterattiva:** stanza che contiene opere interattive.
  - **Corridoio:** spazio di passaggio.

- **Ingresso:** luogo che accoglie i visitatori entrati nel museo.
- **Object properties:**
  - espone (relazione con l'opera d'arte).
  - èContenutoIn (relazione con il museo).
- **Data properties:**
  - **nomeStanza:** il nome della stanza.
  - **descrizioneStanza:** una breve descrizione della stanza.
  - **materialeInterattivoStanza:** indica la presenza di opere interattive nella stanza (0 no, 1 sì).
  - **numeroStanza:** il numero della stanza all'interno del museo.
  - **tipoStanza:** il tipo di stanza (corridoio, stanza tematica o stanza interattiva).
  - **accessibilitàStanza:** indica la possibilità di accedere nella stanza (0 no, 1 sì).

### 3. Opera d'Arte

- Questa classe rappresenta un'opera esposta nel museo.
- **Sottoclassi:**
  - **Statua:** opere tridimensionali.
  - **Quadro:** dipinti.
  - **Artefatto:** oggetti storici.
  - **InstallazioneInterattiva:** opere con cui i visitatori possono interagire.
- **Object properties:**
  - èEspostoIn (relazione con le stanze).
- **Data Properties:**
  - **nomeOpera:** il nome dell'opera d'arte.
  - **autoreOpera:** l'autore dell'opera d'arte.
  - **dataDiCreazioneOpera:** la data di creazione dell'opera d'arte.
  - **descrizioneOpera:** una breve descrizione dell'opera d'arte.
  - **dimensioneOpera:** la dimensione dell'opera d'arte.
  - **interattivitàOpera:** indica se l'opera d'arte è interattiva (0 no, 1 sì).
  - **materialeOpera:** il materiale con cui è stata realizzata l'opera d'arte.
  - **periodoStoricoOpera:** il periodo storico dell'opera d'arte.
  - **tecnicaOpera:** la tecnica con cui è stata realizzata l'opera d'arte.
  - **posizioneOpera:** la posizione in cui è collocata l'opera.

### 4. Agente

- Individui o entità che svolgono ruoli operativi nel museo.

- **Sottoclassi:**
  - **GuidadelMuseo:** accompagna i visitatori durante il tour.
    1. **GuidaPersona:** persona fisica che accompagna i visitatori durante il tour.
    2. **GuidaRobot:** robot che accompagna i visitatori durante il tour.
    3. **NessunaGuida:** non c'è una guida che accompagna i visitatori durante il tour.
  - **Bigliettaio:** vende i biglietti.
    1. **BigliettaioPersona:** persona fisica che vende i biglietti.
    2. **BigliettaioGuida:** robot che vende i biglietti.
- **Object properties:**
  - conduce (relazione tra la guida del museo e il tour guidato).
  - vende (relazione tra il bigliettaio e il biglietto).
- **Data Properties:**
  - **idAgente:** il numero che identifica l'agente.
  - **nomeAgente:** il nome dell'agente.
  - **tipoAgente:** il tipo di agente (guida del museo o bigliettaio).

## 5. Visitatore

- Questa classe rappresenta i visitatori del museo.
- **Sottoclassi:**
  - **VisitatoreIndividuale:** visitatore che visita il museo individualmente.
  - **GruppoDiVisitatori:** gruppo di visitatori che visita il museo.
- **Object properties:**
  - interagisceCon (relazione con l'installazione interattiva).
  - partecipaA (relazione con l'evento).
- **Data Properties:**
  - **nomeVisitatore:** il nome del visitatore.
  - **cognomeVisitatore:** il cognome del visitatore.
  - **dataDiNascitaVisitatore:** la data di nascita del visitatore.
  - **dataVisita:** la data e l'orario in cui è stata effettuata la visita.

## 6. Biglietto

- Entità che rappresenta un documento d'accesso al museo.
- **Sottoclassi:**

- **BigliettoConGuida:** documento d'accesso al museo che prevede la visita senza guida.
- **BigliettoSenzaGuida:** documento d'accesso al museo che prevede la visita con guida.
- **Object properties:**
  - èVendutoDa (relazione con il bigliettaio).
- **Data properties:**
  - **idBiglietto:** il numero che identifica il biglietto.
  - **prezzoBiglietto:** il prezzo del biglietto.
  - **tipoBiglietto:** il tipo di biglietto (biglietto con guida o biglietto senza guida).

## 7. Evento

- Entità che rappresenta un'attività organizzata nel museo.
- **Sottoclassi:**
  - **TourGuidato:** attività organizzata nel museo supportata da una guida.
  - **TourNonGuidato:** attività organizzata nel museo che non è supportata da una guida.
- **Object properties:**
  - èOrganizzatoDa (relazione con il museo).
  - èPartecipatoDa (relazione con il visitatore).
- **Data properties:**
  - **idEvento:** il numero che identifica l'evento.
  - **nomeEvento:** il nome dell'evento.
  - **dataEvento:** la data in cui è stato effettuato l'evento.
  - **descrizioneEvento:** una breve descrizione dell'evento.
  - **tipoEvento:** il tipo di evento (tour guidato o tour non guidato).

## *Object Properties*

Le Object Properties sono un tipo fondamentale di proprietà nelle ontologie OWL (Web Ontology Language). Sono utilizzate per definire relazioni tra due entità, collegando un individuo a un altro. Ogni Object Property è caratterizzata da due elementi principali:

- **Dominio:** Specifica la classe di appartenenza delle entità che possono essere soggetti della relazione.

- **Range:** Definisce la classe di appartenenza delle entità che possono essere oggetti della relazione.

Proprietà	Dominio	Range	Tipo	Significato
<b>conduce</b>	GuidaDelMuseo	TourGuidato	Funzionale	La guida conduce un tour.
<b>èCondottoDa</b>	TourGuidato	GuidaDelMuseo	Funzionale	Il tour è condotto da una guida.
<b>contiene</b>	Museo	Stanza	Transitiva	Il museo contiene una stanza.
<b>èContenutoIn</b>	Stanza	Museo	Funzionale	La stanza è contenuta in un museo.
<b>espone</b>	Stanza	OperaDArte	Non funzionale	La stanza espone un'opera d'arte.
<b>èEpostoIn</b>	OperaDArte	Stanza	Funzionale	L'opera d'arte è esposta in una stanza.
<b>interagisceCon</b>	Visitatore	InstallazioneInterattiva	Funzionale	Un visitatore interagisce con un'installazione interattiva.
<b>èInteragitaDa</b>	InstallazioneInterattiva	Visitatore	Non funzionale	L'installazione interattiva è usata dal visitatore.
<b>organizza</b>	Museo	Evento	Non funzionale	Il museo organizza un evento.

<b>èOrganizzatoDa</b>	Evento	Museo	Funzionale	L'evento è organizzato da un museo.
<b>partecipaA</b>	Visitatore	Evento	Non funzionale	Il visitatore partecipa a un evento.
<b>èPartecipatoDa</b>	Evento	Visitatore	Non funzionale	Un evento ha come partecipante il visitatore.
<b>vende</b>	Bigliettaio	Biglietto	Funzionale	Il bigliettaio vende un biglietto.
<b>èVendutoDa</b>	Biglietto	Bigliettaio	Funzionale	Un biglietto è venduto da un bigliettaio.

## Data Properties

Le Data Properties sono uno dei tipi fondamentali di proprietà utilizzate nelle ontologie OWL (Web Ontology Language). Servono per definire una relazione tra un'entità (un individuo o un'istanza di una classe) e un dato letterale come un numero, una stringa, una data o un altro tipo di valore. Ogni Data Property è caratterizzata da due elementi principali:

- **Dominio:** Indica la classe di appartenenza delle entità che possono utilizzare quella proprietà.
- **Range:** Specifica il tipo di dato associato alla proprietà, ad esempio numeri interi, stringhe, date, o altri tipi definiti da XML Schema Datatypes.

Proprietà	Dominio	Tipo	Descrizione
<b>nomeMuseo</b>	Museo	String	Indica il nome del museo.
<b>cittàMuseo</b>	Museo	String	Indica la città in cui risiede il museo.
<b>statoMuseo</b>	Museo	String	Indica lo Stato in cui risiede il museo.
<b>descrizioneMuseo</b>	Museo	String	Breve descrizione del museo.

<b>nomeStanza</b>	Stanza	String	Indica il nome della stanza.
<b>accessibilitàStanza</b>	Stanza	Boolean	Indica l'accessibilità della stanza (0 no, 1 sì).
<b>descrizioneStanza</b>	Stanza	String	Breve descrizione della stanza.
<b>materialeInterattivoStanza</b>	Stanza	Boolean	Indica la presenza di materiale interattivo nella stanza (0 no, 1 sì).
<b>numeroStanza</b>	Stanza	Integer	Indica il numero della stanza nel museo.
<b>tipoStanza</b>	Stanza	String	Indica il tipo di stanza (corridoio, stanza tematica o stanza interattiva).
<b>nomeOpera</b>	OperaDArte	String	Indica il nome dell'opera d'arte.
<b>autoreOpera</b>	OperaDArte	String	Indica il nome dell'autore.
<b>dataDiCreazioneOpera</b>	OperaDArte	dateTime	Indica la data di creazione dell'opera.
<b>descrizioneOpera</b>	OperaDArte	String	Breve descrizione dell'opera.
<b>dimensioneOpera</b>	OperaDArte	String	Indica la dimensione dell'opera.
<b>interattivitàOpera</b>	OperaDArte	Boolean	Indica l'interattività dell'opera (0 no, 1 sì).
<b>materialeOpera</b>	OperaDArte	String	Indica il materiale dell'opera.
<b>periodoStoricoOpera</b>	OperaDArte	String	Indica il periodo storico dell'opera.
<b>tecnicOpera</b>	OperaDArte	String	Indica la tecnica con cui è stata realizzata l'opera.
<b>posizioneOpera</b>	OperaDArte	String	Indica le coordinate esatte in cui si trova l'opera.
<b>idAgente</b>	Agente	Integer	Il numero che identifica l'agente.
<b>nomeAgente</b>	Agente	String	Indica il nome dell'agente.
<b>tipoAgente</b>	Agente	String	Indica il tipo di agente (guida del museo o bigliettaio).
<b>nomeVisitatore</b>	Visitatore	String	Indica il nome del visitatore.
<b>cognomeVisitatore</b>	Visitatore	String	Indica il cognome del visitatore.
<b>dataDiNascitaVisitatore</b>	Visitatore	dateTime	Indica la data e l'ora di nascita del visitatore.
<b>dataVisita</b>	Visitatore	dateTime	Indica la data e l'ora della visita.
<b>idBiglietto</b>	Biglietto	Integer	Il numero che identifica il biglietto.
<b>prezzoBiglietto</b>	Biglietto	Float	Indica il prezzo del biglietto.

<b>tipoBiglietto</b>	Biglietto	String	Indica il tipo di biglietto (biglietto con guida o biglietto senza guida).
<b>idEvento</b>	Evento	Integer	Il numero che identifica l'evento.
<b>nomeEvento</b>	Evento	String	Indica il nome dell'evento.
<b>dataEvento</b>	Evento	dateTime	Indica la data e l'ora in cui si è tenuto l'evento.
<b>descrizioneEvento</b>	Evento	String	Breve descrizione dell'evento.
<b>tipoEvento</b>	Evento	String	Indica il tipo di evento (tour guidato o tour non guidato).

## Individuals

Gli individuals rappresentano le entità specifiche o istanze di una classe all'interno di un'ontologia OWL (Web Ontology Language). Sono gli oggetti "concreti" del dominio modellato, cioè gli esempi reali o immaginari che appartengono alle classi definite nell'ontologia. Ogni Individual è caratterizzata da elementi principali:

- **Appartenenza a una classe:** ogni individual è istanza di una o più classi definite nell'ontologia.
- **Unicità:** ogni individual ha un URI (Uniform Resource Identifier) che lo identifica univocamente all'interno dell'ontologia.
- **Proprietà associate:** gli individuals possono avere Data Properties (che collegano l'individuo a un valore letterale) e Object Properties (che collegano l'individuo ad altri individui).
- **Partecipazione alle inferenze:** gli individuals sono utilizzati dai reasoner per inferire nuove conoscenze in base alle regole e alle relazioni definite nell'ontologia.

In questo progetto sono state definite specifiche istanze poiché l'intera base di conoscenza è integrata all'interno dell'ontologia. Questo approccio ha reso superfluo l'utilizzo di software come *GraphDB* o *Neo4j* per la gestione di database a grafo. Al loro posto, è stata adottata la libreria *RDFlib* che offre strumenti potenti e flessibili per la manipolazione e l'interrogazione di dati RDF. Questo approccio risulta particolarmente adatto per applicazioni che richiedono un'interazione diretta con l'ontologia senza la necessità di architetture di database separate.

### 1. Museo

- museo contiene stanzaInterattiva
- museo contiene stanzaVanGogh
- museo contiene stanzaItalia
- museo contiene stanzaGrecia
- museo contiene corridoio
- museo contiene ingresso
- museo organizza tourGuidato1
- museo organizza tourGuidato2
- museo organizza tourNonGuidato1
- museo organizza tourNonGuidato2

## 2. Stanza

- **Sottoclassi:**

- **StanzaTematica**

- 1. **stanzaGrecia**

- stanzaGrecia espone capitelloIonico
      - stanzaGrecia espone vasoGreco
      - stanzaGrecia espone Discobolo
      - stanzaGrecia espone AfroditeDiMilo
      - stanzaGrecia èContenutoIn museo

- 2. **stanzaItalia**

- stanzaItalia espone David
      - stanzaItalia espone Gioconda
      - stanzaItalia espone AmoreEPische
      - stanzaItalia èContenutoIn museo

- 3. **stanzaVanGogh**

- stanzaVanGogh espone Autoritratto
      - stanzaVanGogh espone RamoDiMandorloFiorito
      - stanzaVanGogh espone LaNottiStellata
      - stanzaVanGogh èContenutoIn museo

- **StanzaInterattiva**

- 1. stanzaInterattiva espone globoTerrestre
      - 2. stanzaInterattiva espone pendolo
      - 3. stanzaInterattiva espone sistemaSolare
      - 4. stanzaInterattiva èContenutoIn museo

- **Corridoio**
    1. corridoio èContenutoIn museo
  - **Ingresso**
    1. ingresso èContenutoIn museo
3. **Opera d'Arte**
- **Sottoclassi:**
    - **Statua**
      1. AfroditeDiMilo èEspostoIn stanzaGrecia
      2. AmoreEPische èEspostoIn stanzaItalia
      3. David èEspostoIn stanzaItalia
      4. Discobolo èEspostoIn stanzaGrecia
    - **Quadro**
      1. Autoritratto èEspostoIn stanzaVanGogh
      2. Gioconda èEspostoIn stanzaItalia
      3. LaNotteStellata èEspostoIn stanzaVanGogh
      4. RamoDiMandorloFiorito èEspostoIn stanzaVanGogh
    - **Artefatto**
      1. capitelloIonico èEspostoIn stanzaGrecia
      2. vasoGreco èEspostoIn stanzaGrecia
    - **InstallazioneInterattiva**
      1. **globoTerrestre**
        - globoTerrestre èEspostoIn stanzaInterattiva
        - globoTerrestre èInteragitaDa visitatore1
        - globoTerrestre èInteragitaDa visitatore2
        - globoTerrestre èInteragitaDa visitatore3
        - globoTerrestre èInteragitaDa visitatore4
        - globoTerrestre èInteragitaDa gruppo1
        - globoTerrestre èInteragitaDa gruppo2
      2. **pendolo**
        - pendolo èEspostoIn stanzaInterattiva
        - pendolo èInteragitaDa visitatore1
        - pendolo èInteragitaDa visitatore2
        - pendolo èInteragitaDa visitatore3
        - pendolo èInteragitaDa visitatore4

- pendolo èInteragitaDa gruppo1
- pendolo èInteragitaDa gruppo2

### 3. sistemaSolare

- sistemaSolare èEspostoIn stanzaInterattiva
- sistemaSolare èInteragitaDa visitatore1
- sistemaSolare èInteragitaDa visitatore2
- sistemaSolare èInteragitaDa visitatore3
- sistemaSolare èInteragitaDa visitatore4
- sistemaSolare èInteragitaDa gruppo1
- sistemaSolare èInteragitaDa gruppo2

## 4. Agente

- Sottoclassi:

- GuidadelMuseo:

- 1. GuidaPersona

- guidaPersona conduce tourGuidato1
    - guidaPersona conduce tourGuidato2

- 2. GuidaRobot

- guidaRobot conduce tourGuidato1
    - guidaRobot conduce tourGuidato2

- 3. NessunaGuida

- nessunaGuida conduce tourNonGuidato1
    - nessunaGuida conduce tourNonGuidato2

- Bigliettaio:

- 1. BigliettaioPersona

- bigliettaioPersona vende bigliettoConGuida
    - bigliettaioPersona vende bigliettoSenzaGuida

- 2. BigliettaioGuida

- bigliettaioRobot vende bigliettoConGuida
    - bigliettaioRobot vende bigliettoSenzaGuida

## 5. Visitatore

- Sottoclassi:

- VisitatoreIndividuale

- 1. visitatore1

- visitatore1 partecipaA tourGuidato1

- visitatore1 partecipaA tourGuidato2
- visitatore1 partecipaA tourNonGuidato1
- visitatore1 partecipaA tourNonGuidato2
- visitatore1 interagisceCon globoTerrestre
- visitatore1 interagisceCon pendolo
- visitatore1 interagisceCon sistemaSolare

## 2. visitatore2

- visitatore2 partecipaA tourGuidato1
- visitatore2 partecipaA tourGuidato2
- visitatore2 partecipaA tourNonGuidato1
- visitatore2 partecipaA tourNonGuidato2
- visitatore2 interagisceCon globoTerrestre
- visitatore2 interagisceCon pendolo
- visitatore2 interagisceCon sistemaSolare

## 3. visitatore3

- visitatore3 partecipaA tourGuidato1
- visitatore3 partecipaA tourGuidato2
- visitatore3 partecipaA tourNonGuidato1
- visitatore3 partecipaA tourNonGuidato2
- visitatore3 interagisceCon globoTerrestre
- visitatore3 interagisceCon pendolo
- visitatore3 interagisceCon sistemaSolare

## 4. visitatore4

- visitatore4 partecipaA tourGuidato1
- visitatore4 partecipaA tourGuidato2
- visitatore4 partecipaA tourNonGuidato1
- visitatore4 partecipaA tourNonGuidato2
- visitatore4 interagisceCon globoTerrestre
- visitatore4 interagisceCon pendolo
- visitatore4 interagisceCon sistemaSolare

## ▪ GruppoDiVisitatori

### 1. gruppo1

- gruppo1 partecipaA tourGuidato1

- gruppo1 partecipaA tourGuidato2
- gruppo1 partecipaA tourNonGuidato1
- gruppo1 partecipaA tourNonGuidato2
- gruppo1 interagisceCon globoTerrestre
- gruppo1 interagisceCon pendolo
- gruppo1 interagisceCon sistemaSolare

## 2. **gruppo2**

- gruppo2 partecipaA tourGuidato1
- gruppo2 partecipaA tourGuidato2
- gruppo2 partecipaA tourNonGuidato1
- gruppo2 partecipaA tourNonGuidato2
- gruppo2 interagisceCon globoTerrestre
- gruppo2 interagisceCon pendolo
- gruppo2 interagisceCon sistemaSolare

## 6. **Biglietto**

- **Sottoclassi:**

- **BigliettoConGuida**

1. bigliettoConGuida èVendutoDa bigliettaioPersona
2. bigliettoConGuida èVendutoDa bigliettaioRobot

- **BigliettoSenzaGuida**

1. bigliettoSenzaGuida èVendutoDa bigliettaioPersona
2. bigliettoSenzaGuida èVendutoDa bigliettaioRobot

## 7. **Evento**

- **Sottoclassi:**

- **TourGuidato**

- 1. **tourGuidato1**

- tourGuidato1 èCondottoDa guidaPersona
- tourGuidato1 èCondottoDa guidaRobot
- tourGuidato1 èOrganizzatoDa museo
- tourGuidato1 èPartecipatoDa visitatore1
- tourGuidato1 èPartecipatoDa visitatore2
- tourGuidato1 èPartecipatoDa visitatore3
- tourGuidato1 èPartecipatoDa visitatore4

- tourGuidato1 èPartecipatoDa gruppo1
- tourGuidato1 èPartecipatoDa gruppo2

## 2. tourGuidato2

- tourGuidato2 èCondottoDa guidaPersona
- tourGuidato2 èCondottoDa guidaRobot
- tourGuidato2 èOrganizzatoDa museo
- tourGuidato2 èPartecipatoDa visitatore1
- tourGuidato2 èPartecipatoDa visitatore2
- tourGuidato2 èPartecipatoDa visitatore3
- tourGuidato2 èPartecipatoDa visitatore4
- tourGuidato2 èPartecipatoDa gruppo1
- tourGuidato2 èPartecipatoDa gruppo2

## ▪ TourNonGuidato

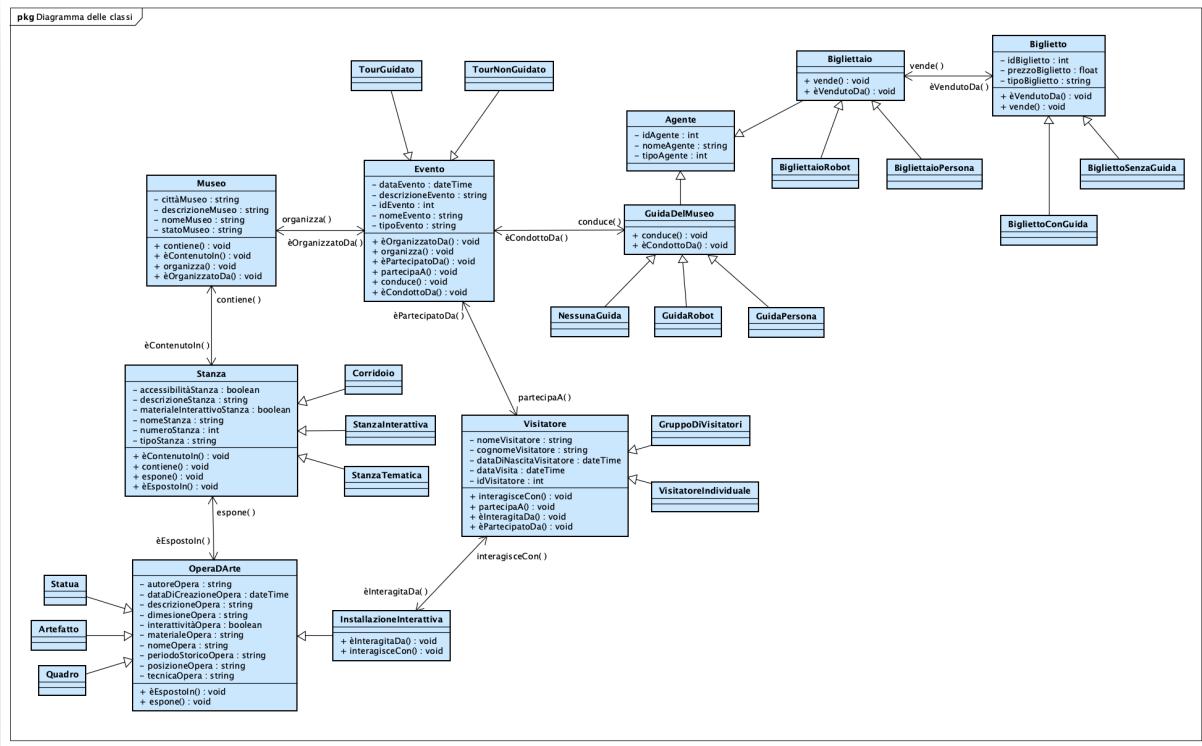
### 1. tourNonGuidato1

- tourNonGuidato1 èCondottoDa nessunaGuida
- tourNonGuidato1 èOrganizzatoDa museo
- tourNonGuidato1 èPartecipatoDa visitatore1
- tourNonGuidato1 èPartecipatoDa visitatore2
- tourNonGuidato1 èPartecipatoDa visitatore3
- tourNonGuidato1 èPartecipatoDa visitatore4
- tourNonGuidato1 èPartecipatoDa gruppo1
- tourNonGuidato1 èPartecipatoDa gruppo2

### 2. tourNonGuidato2

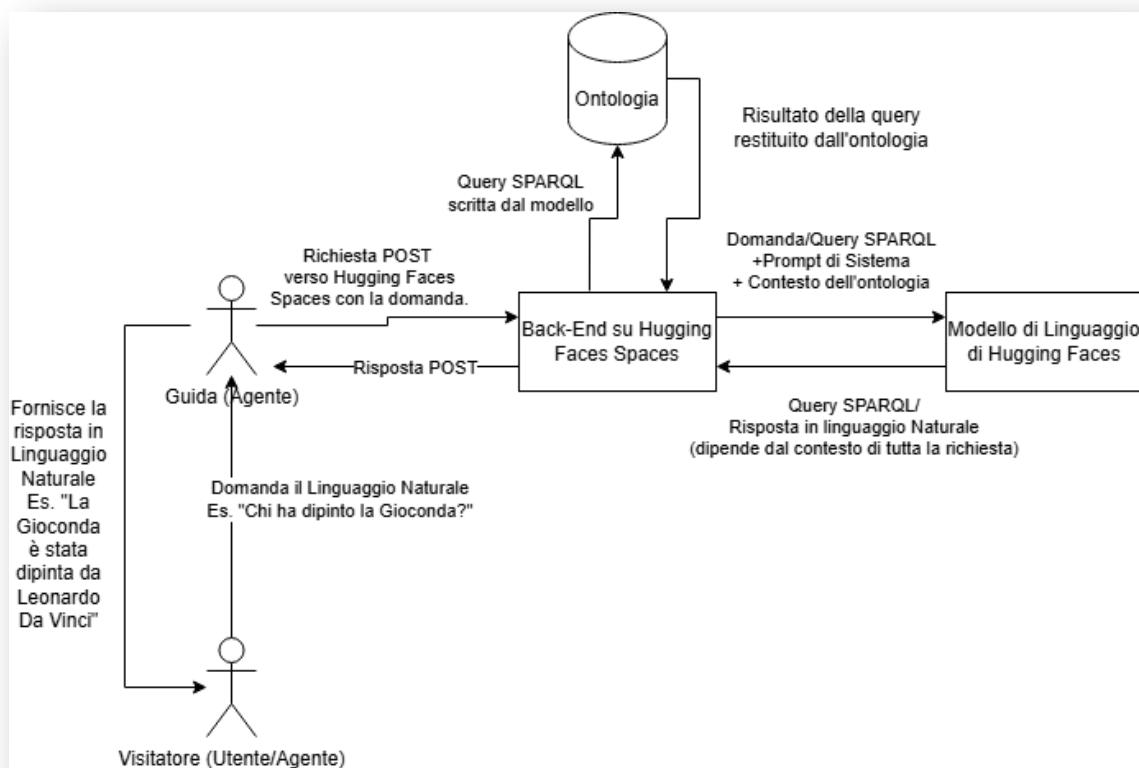
- tourNonGuidato2 èCondottoDa nessunaGuida
- tourNonGuidato2 èOrganizzatoDa museo
- tourNonGuidato2 èPartecipatoDa visitatore1
- tourNonGuidato2 èPartecipatoDa visitatore2
- tourNonGuidato2 èPartecipatoDa visitatore3
- tourNonGuidato2 èPartecipatoDa visitatore4
- tourNonGuidato2 èPartecipatoDa gruppo1
- tourNonGuidato2 èPartecipatoDa gruppo2

## Diagramma delle classi



Il diagramma delle classi rappresenta l'ontologia del museo, mostrando le principali entità coinvolte e le loro relazioni. L'insieme delle classi e delle relazioni definite permette di modellare il funzionamento del museo, includendo la gestione degli eventi, dei visitatori e delle opere esposte.

# Processo di Recupero dei Dati della Base di Conoscenza e Generazione della Risposta



Il sistema di guida robotica sviluppato per il museo, integrato con ontologie semantiche, mira a migliorare significativamente l'interazione uomo-macchina in un contesto educativo e culturale. La guida robotica è progettata per comprendere e interpretare le domande dei visitatori espresse in linguaggio naturale, elaborandole attraverso un avanzato processo di analisi semantica e fornendo risposte chiare, pertinenti e coinvolgenti. Questo risultato è reso possibile grazie alla sinergia di diverse tecnologie avanzate, tra cui l'elaborazione del linguaggio naturale (NLP), l'uso di ontologie RDF e l'interrogazione della base di conoscenza mediante query SPARQL. Inoltre, il sistema sfrutta modelli di linguaggio di ultima generazione per affinare ulteriormente la comprensione e la contestualizzazione delle richieste degli utenti.

## 1. Ricezione della domanda in linguaggio naturale

L'interazione ha inizio quando un visitatore si ferma davanti a un'opera d'arte o a una statua e pone una domanda. Grazie a microfoni direzionali e tecnologie avanzate di riduzione del rumore, la guida robotica cattura l'input vocale, lo converte in testo e invia la stringa risultante tramite una richiesta HTTP POST a un endpoint remoto ospitato su Hugging Face Spaces. Questo approccio centralizzato

non solo facilita la scalabilità, ma garantisce anche una maggiore precisione nell'elaborazione delle domande, assicurando che ogni richiesta venga interpretata con la massima accuratezza.

## 2. Elaborazione della domanda tramite prompt di sistema

Un elemento chiave della pipeline è il prompt di sistema, un insieme strutturato di istruzioni progettate per guidare il comportamento del modello di linguaggio. Nello specifico, il prompt definisce:

- **Ruolo del modello:** Il modello deve agire come un assistente museale esperto in ontologie RDF.
- **Regole di risposta:** Sono previste direttive stringenti che richiedono la generazione di query SPARQL concise, in un'unica riga, utilizzando prefissi predefiniti e seguendo uno schema rigoroso conforme alle proprietà semantiche del dominio museale.
- **Contesto della base di conoscenza:** Il prompt include informazioni sintetizzate sull'ontologia RDF, comprese classi, proprietà ed entità rilevanti, per consentire al modello di operare in un contesto ben definito.
- **Esempi guida:** Per agevolare l'interpretazione, il prompt fornisce esempi esplicativi di domande e risposte, come la trasformazione della domanda "Chi ha creato la statua del Discobolo?" in una query SPARQL precisa e concisa.

### ### Istruzioni ###

Sei un assistente museale esperto in ontologie RDF. Utilizza le informazioni fornite per generare query SPARQL precise e pertinenti.

### ### Ontologia ###

[Dettagli sull'ontologia RDF, incluse classi, proprietà ed entità rilevanti.]

### ### Regole ###

1. La query SPARQL deve essere scritta su una sola riga.
2. La risposta deve essere chiara, diretta e basata sui dati RDF.
3. In caso di fallimento, genera una query alternativa.
4. La risposta finale deve includere i risultati della query interpretati in linguaggio naturale.

### ### Conversazione ###

*Utente:* Chi ha creato la statua del Discobolo?

*Assistant:* PREFIX base: <<http://www.semanticweb.org/ontologies/museo#>> SELECT ?autore WHERE { ?statua base:hasName "Discobolo" . ?statua base:creatoDa ?autore . }

### 3. Generazione della query SPARQL

Il modello di linguaggio elabora la domanda ricevuta e, grazie al prompt di sistema, genera una query SPARQL conforme all'ontologia del museo. Questa query è progettata per interrogare specifici campi della base di conoscenza, come il periodo storico di un'opera o il suo autore.

*Esempio:*

- **Domanda:** "Qual è il periodo storico della statua del Discobolo?"
- **Query generata:**

```
PREFIX base: <http://www.semanticweb.org/ontologies/museo#> SELECT ?periodo WHERE {
?statua base:hasName "Discobolo" . ?statua base:appartieneA ?periodo . }
```

Il modello è inoltre programmato per riconoscere espressioni come "fammi vedere", "portami all'opera X" o "dove si trova", integrando nelle query la proprietà relativa alla posizione delle opere (ad esempio, `progettoMuseo:posizioneOpera`). Questo consente di fornire al sistema di navigazione informazioni precise per la ripianificazione del percorso.

### 4. Esecuzione della query sulla base di conoscenza

Una volta generata, la query viene sottoposta a un processo di correzione sintattica tramite algoritmi euristici come la funzione di correzione avanzata che elimina eventuali errori di formattazione, tra cui la rimozione di newline, la gestione degli spazi e l'aggiunta dei prefissi necessari. Successivamente, il sistema verifica la validità della query utilizzando un parser basato su RDFLib e, se correttamente formulata, la esegue sul motore RDF (come GraphDB) che ospita l'ontologia del museo. I dati restituiti, contenenti dettagli quali nomi, autori, date, materiali ed eventualmente posizioni, vengono raccolti e trasformati in una risposta in linguaggio naturale. Nel caso in cui la query, sebbene valida, non produca risultati, il sistema adotta un meccanismo di fallback: la risposta viene generata utilizzando le conoscenze pregresse del modello di linguaggio, garantendo comunque un output informativo per l'utente.

### 5. Interpretazione e presentazione dei risultati

I dati ottenuti dall'esecuzione della query vengono elaborati e formattati per essere facilmente comprensibili ai visitatori. Un secondo prompt di sistema, generato da una funzione dedicata, istruisce

il modello a rispondere come una "guida museale virtuale", utilizzando un linguaggio discorsivo e naturale. Questo prompt include esplicitamente:

- La domanda originaria dell'utente.
- La query SPARQL generata (se disponibile).
- I risultati ottenuti dalla query o, in assenza di questi, un'indicazione che non sono stati trovati dati pertinenti.

Oltre a fornire informazioni storiche o descrittive sull'opera, la risposta guida include anche un invito all'azione. Quando la domanda riguarda la posizione di un'opera (ad esempio, "fammi vedere", "portami all'opera X"), il sistema interpreta i risultati della query e genera una risposta che esorta il robot a ripianificare il proprio percorso. Questo comando viene elaborato dal modulo di navigazione, che sfrutta algoritmi come A\* e il sistema NavMesh per ricalcolare il tragitto ottimale e adattare dinamicamente il percorso in base ai dati ricevuti.

#### ***Esempio di risposta guidata:***

"L'autore del Discobolo è Mirone."

### **6. Comunicazione della risposta al visitatore**

La risposta finale viene trasmessa al visitatore attraverso due canali:

- **Sintesi vocale:** La guida robotica comunica verbalmente la risposta, offrendo un'esperienza interattiva e immediata.
- **Display visivo:** Il testo della risposta viene mostrato su uno schermo o in una finestra di dialogo, garantendo accessibilità e chiarezza delle informazioni.

Questo approccio multicanale assicura che, indipendentemente dalle condizioni ambientali o dalle preferenze dell'utente, le informazioni siano sempre fruibili.

### **7. Traduzione**

Una volta generata la risposta finale, il sistema procede con un'attività aggiuntiva fondamentale:

- **Traduzione:** Il modulo di traduzione, integrato all'interno della pipeline, verifica se la lingua della risposta corrisponde a quella della domanda.

- Utilizzando un modello di rilevamento linguistico (ad es. "xlm-roberta-base-language-detection"), il sistema identifica la lingua in cui è stata formulata la domanda e quella della risposta.
- Se le lingue non coincidono, viene scelto dinamicamente il modello di traduzione appropriato.

## **Ottimizzazioni tecniche e ruolo del prompt di sistema**

Il prompt di sistema è fondamentale per garantire un'interazione fluida e coerente tra il modello di linguaggio e il dominio specifico del museo. Le ottimizzazioni tecniche e le strategie integrate nel prompt sono progettate per:

### **1. Mantenere la coerenza nel contesto museale**

- **Definizione del ruolo e del linguaggio:** Il prompt stabilisce chiaramente che il modello agisce come un assistente museale esperto in ontologie RDF, assicurando che il linguaggio utilizzato sia pertinente e adatto alla circostanza culturale del museo, evitando errori di ambiguità o risposte fuori contesto.
- **Utilizzo di esempi e istruzioni:** Esempi concreti (ad esempio la trasformazione di “Chi ha creato il Discobolo?” in una query SPARQL) vengono forniti per aiutare il modello a produrre risposte coerenti e precise, facilitando l'adozione del linguaggio specifico del dominio.
- **Linee guida linguistiche:** Regole interne sono definite per garantire la consistenza, come la generazione di query SPARQL su una singola riga, l'uso di prefissi predefiniti e la gestione standardizzata degli spazi. Questo permette di mantenere una struttura uniforme e facilmente leggibile.

### **2. Garantire la precisione nelle query SPARQL**

- **Struttura e formattazione rigorosa:** Il prompt specifica in dettaglio la sintassi corretta delle query SPARQL, assicurando che il modello generi query precise e conformi agli standard. Include l'obbligo di terminare ogni tripla con un punto e di separare correttamente i prefissi dalle variabili.
- **Contesto dell'ontologia:** Il prompt fornisce una sintesi completa dell'ontologia RDF, inclusi dettagli sulle classi, proprietà e entità pertinenti per garantire che le query siano contestualizzate correttamente.
- **Gestione delle richieste di posizione:** Il prompt guida l'inclusione di proprietà relative alla posizione nelle query, garantendo che la geolocalizzazione o altre variabili

spaziali siano correttamente integrate nei risultati. Se la domanda include richieste specifiche come “fammi vedere l'opera Discobolo” o “portami all'opera Discobolo”, il prompt orienta il modello affinché includa nella query la proprietà relativa alla posizione (`progettoMuseo:posizioneOpera`).

### 3. Abilitare strategie di fallback e gestione dei fallimenti

- **Fallback integrato:** In presenza di ambiguità, quando la domanda non è pertinente o ci sono errori nella formulazione della query, il modello è programmato per restituire comandi alternativi come il token "NO\_SPARQL". Questo meccanismo di fallback è essenziale per garantire che il sistema possa comunque fornire una risposta, anche se generica, quando la domanda non rientra perfettamente nel dominio dell'ontologia.
- **Gestione delle ambiguità:** Le istruzioni nel prompt includono meccanismi per gestire discrepanze nei dati o richieste incomplete, orientando il modello verso soluzioni appropriate ed evitando che produca output incompleti o errati. In presenza di dati parziali, il sistema è programmato a utilizzare conoscenze pregresse per integrare la risposta finale.

### 4. Integrazione con strumenti tecnici e modelli avanzati

- **Interazione con il motore RDF:** Il prompt fornisce le linee guida necessarie affinché la query generata sia compatibile con il motore RDF utilizzato (ad esempio, GraphDB). Questo accorgimento assicura che l'interrogazione sulla base di conoscenza venga effettuata in modo efficiente e preciso.
- **Sinergia con il modulo di traduzione:** Il modello di linguaggio non è limitato a produrre solo query e risposte in un'unica lingua. Le istruzioni del prompt, insieme ai successivi processi di rilevamento linguistico e traduzione, garantiscono che l'output sia sempre consegnato nella lingua dell'utente.
- **Explainability e debug:** Ogni fase del processo (generazione, correzione e validazione della query, esecuzione, e traduzione) è tracciata in modo dettagliato tramite un dizionario di explainability. Il prompt, infondendo regole precise, rende più semplice identificare eventuali errori o aree di miglioramento, facilitando il debug e l'ottimizzazione del sistema.

## Generazione della Query SPARQL

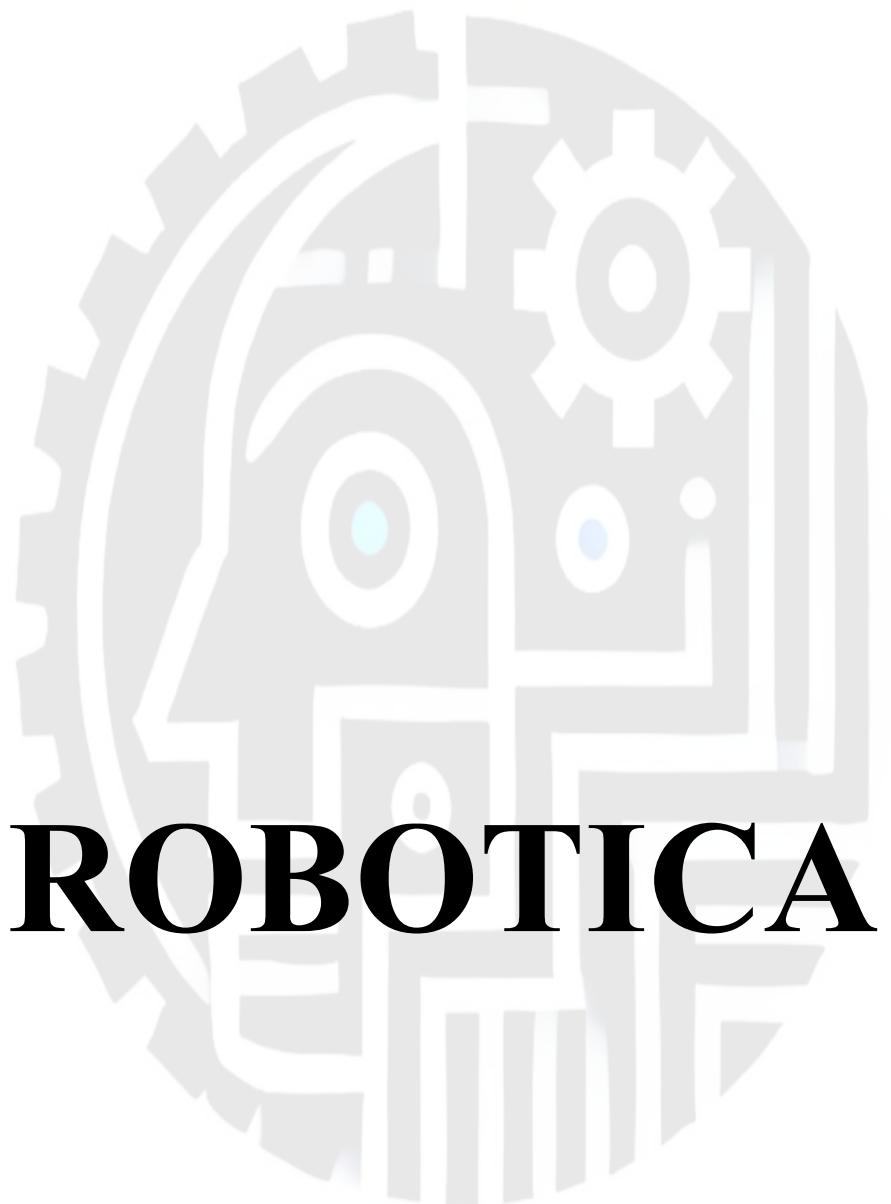
### 1. Decisione e generazione della query

- Se la domanda è pertinente all'ontologia:
  - Il sistema procede a generare la query SPARQL.

- Viene creato un prompt di sistema specifico che include:
  - il contesto dell'ontologia RDF;
  - regole stringenti per la formattazione della query su una sola riga;
  - esempi guida e istruzioni per includere la proprietà di posizione, se richiesto (ad esempio, in domande come "fammi vedere" o "portami", che implicano una localizzazione dell'opera).
- Il prompt, insieme alla domanda, viene inviato al modello generativo (ad esempio, tramite il client Hugging Face) per ottenere come output la query SPARQL.
- Se la domanda non è pertinente all'ontologia:
  - Il sistema interrompe il processo di generazione della query.
  - Il modello restituisce una risposta speciale, NO\_SPARQL, indicando che non è possibile formulare una query coerente con l'ontologia per una richiesta fuori dal dominio museale.

## 2. Correzione e validazione della query

- La query generata viene passata a un modulo di correzione sintattica, che:
  - rimuove newline e spazi extra;
  - verifica la presenza del prefisso obbligatorio (ad esempio, PREFIX progettoMuseo:  
`<http://www.semanticweb.org/lucreziamosca/ontologies/progettoMuseo#>;`);
  - garantisce la corretta separazione tra proprietà e variabili.
- Successivamente, il sistema utilizza un parser SPARQL (ad esempio, basato su RDFLib) per validare la sintassi della query.
- Se la query è sintatticamente corretta, verrà eseguita sull'ontologia. In caso contrario, il sistema adotta strategie di fallback per tentare una query alternativa o restituisce NO\_SPARQL.



**ROBOTICA**

**ROBOEXHIBIT**

## Struttura del progetto

Il progetto è organizzato in due macrocategorie principali:

- **Assets:** questa categoria include tutti gli elementi necessari per la modellazione dello scenario e l'implementazione delle funzionalità degli agenti. È gestibile manualmente e contiene risorse come modelli 3D, prefabbricati installati dall'asset store o da siti di terze parti, materiali e script. Si distingue una directory di nome *Scenes* dedicata alle componenti create con gli strumenti di Unity.
  - **BehaviorTree:** contiene tutti i Behavior Tree utilizzati dagli agenti.
  - **Materials:** raccoglie tutti i materiali impiegati nel progetto.
  - **MuseoDefinitivo:** directory che include le Nav Mesh per la navigazione degli agenti.
  - **Scripts:** suddivisi in diverse sottodirectory:
    - **Animazioni:** directory che contiene gli script per la gestione delle animazioni delle opere.
    - **Dialog Script:** directory contenente gli script per la gestione del dialogo tra gli agenti.
    - **Tour Data:** directory contenente tutte le classi dedicate alla memorizzazione delle informazioni sul tour.
    - **UI Controller:** directory contenete i controller per le interfacce utente.
    - **Agent Controller e Kalman Filter:** classi adibite per l'implementazione del filtro di Kalman.
    - **Avatar Controller:** script per il movimento dell'utente.
    - **MacTextToSpeech:** script per la sintesi vocale, basato sulla funzionalità nativa di MacOS.
    - **UserInteraction:** script per la gestione dell'interazione tra l'utente e le opere interattive.
  - **MuseoDefinitivoScenes:** scena principale in cui è ambientato il progetto.
- **Packages:** questa categoria contiene tutte le dipendenze e i pacchetti esterni utilizzati nel progetto. Comprende pacchetti ufficiali di Unity e di terze parti. Non è modificabile manualmente, poiché la gestione avviene automaticamente attraverso l'ambiente di sviluppo.

## Modellazione dell'ambiente

Il progetto è stato realizzato per ricreare l'aspetto di un museo, suddiviso in sei ambienti distinti:

1. **Sala della Biglietteria**
2. **Corridoio**
3. **Sala dei Reperti Greci**
4. **Sala dei Reperti di Van Gogh**
5. **Sala dei Reperti Italiani**
6. **Sala delle Opere Interattive**

Ogni sala, ad eccezione della biglietteria e del corridoio, ospita diverse opere. Alcune di queste sono interattive e dotate di animazioni che si attivano in risposta all'interazione dei visitatori.

## Realizzazione dell'Ambiente

L'ambiente museale è stato costruito sfruttando sia strumenti nativi che asset provenienti dall'Asset Store di Unity e dalla piattaforma Sketchfab.

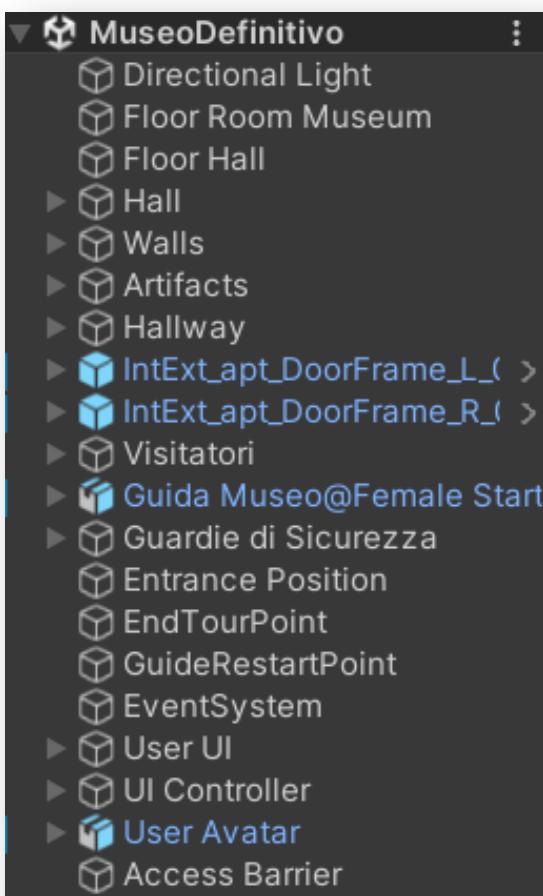
### Asset impiegati:

- **Airport Toilet Restroom Sign**
- **AK Studio**
- **Basic Bedroom Pack**
- **Brick Project Studio**
- **Greek Statues**
- **Greek Columns**
- **Greek Temple**
- **Italy Furniture**
- **Metal Detector**
- **Office Pros Softmax**
- **Van Gogh Museum - Self Portrait**
- **Van Gogh Museum - Almond Blossom**
- **Van Gogh Museum - Starry Night**
- **Wooden Floor**

## Struttura del Museo

Le fondamenta, le pareti e i supporti per le opere sono stati progettati con gli strumenti nativi di Unity:

- **Fondamenta:** realizzate con due *plane* (oggetti nativi di Unity) che fungono da base dell'ambiente, su cui sono posizionati agenti e ostacoli.
- **Pareti:** utilizzate per definire la forma e la dimensione delle stanze e per stabilire i margini di navigazione degli agenti.
- **Supporti e decorazioni:** I sostegni per le opere e gli elementi decorativi sono stati realizzati con strumenti di Unity e arricchiti dagli asset selezionati.



La struttura gerarchica della scena è organizzata come illustrato nell'immagine.

- **Directional Light:** luce direzionale messa a disposizione da Unity per illuminare la scena.
- **Floor Room Museum:** suolo calpestabile del museo su cui sono edificate le sale e posizionate le opere.
- **Floor Hall:** suolo dell'entrata del museo su cui è posizionata la biglietteria.
- **Hall:** macro Gameobject che include tutti i Gameobject posizionati all'ingresso, come il bigliettaio e gli ostacoli statici.
- **Walls:** macro Gameobject che include tutte le pareti che definiscono i contorni delle sale.
- **Artifacts:** macro GameObject che comprende i contenuti delle sale, inclusi opere d'arte e mobili (ostacoli statici).
- **Hallway:** macro GameObject che rappresenta il corridoio composto interamente da ostacoli.
- **DoorFrame:** Gameobject delle porte di ingresso.
- **Visitatori:** Gameobject che incorpora tutti gli agenti visitatori del museo.
- **Guida Museo:** Gameobject dell'agente guida.
- **Guardie di Sicurezza:** GameObject dedicato alle guardie di sicurezza che nella scena assumono il ruolo di ostacoli statici.
- **Entrance Position:** GameObject vuoto che definisce il punto di entrata/uscita del museo.

- **EndTourPoint:** GameObject vuoto che definisce il punto in cui la guida si colloca quando annuncia la fine del tour guidato.
- **GuideRestartPoint:** GameObject vuoto che definisce il punto in cui la guida inizia il tour.
- **User UI:** raccolta delle interfacce utente per l'interazione con l'ambiente circostante e gli agenti all'interno della scena.
- **UI Controller:** raccolta di controller per le interfacce utente, responsabili della gestione della logica dietro le interfacce e del controllo della loro attivazione e disattivazione.
- **User Avatar:** GameObject controllato dall'utente.
- **Access Barrier:** GameObject utilizzato per il controllo dell'accesso dell'utente alle sale del museo. L'oggetto viene disattivato una volta che l'utente ha selezionato la tipologia di biglietto.

Con l'aggiunta degli oggetti decorativi provenienti dagli asset, il risultato ottenuto è il seguente:



## Modellazione degli agenti

Per la realizzazione degli agenti è stata utilizzata la piattaforma Mixamo che offre gratuitamente una vasta gamma di modelli animati. Questo ha permesso di curare l'aspetto estetico degli agenti, garantendo animazioni fluide e coerenti con il contesto museale.

Il comportamento degli agenti è stato implementato attraverso i Behaviour Tree, una struttura gerarchica progettata per gestire in modo ordinato e flessibile le azioni di ciascun agente. Per costruire i Behaviour Tree è stato utilizzato l'asset Behaviour Bricks che fornisce un'ampia gamma di nodi predefiniti e consente la creazione di nodi personalizzati. Ogni nodo è associato a uno script C# dedicato, all'interno del quale viene definita la logica di esecuzione. La sequenza delle azioni è determinata dalla struttura gerarchica e dagli archi che collegano i nodi.

## Strategia Implementativa per gli Agenti

*Guida del Museo:*



La guida è un agente dinamico progettato per condurre i visitatori attraverso un tour organizzato delle opere esposte. È dotata di intelligenza artificiale per rispondere alle domande dei visitatori, gestire il percorso tramite algoritmi di navigazione e interagire in modo dinamico durante il tour.

### 1. **GuideTourData**

- Questo script gestisce tutte le informazioni necessarie per il tour guidato, tra cui il movimento della guida, la gestione delle domande dei visitatori, il riconoscimento dei partecipanti al tour e la terminazione dell'itinerario.
- Contiene gli attributi per monitorare lo spostamento della guida, registrare i visitatori che partecipano al tour, stabilire l'ordine delle stanze da visitare e concludere il tour.

### 2. **WaitVisitor**

- Questo script mette la guida in uno stato di attesa fino a quando almeno un visitatore non acquista un biglietto con guida e si avvicina alla sua posizione.
- Dopo aver rilevato un visitatore idoneo, la guida attende un tempo casuale preimpostato. Se, al termine di questo tempo, non si presenta un altro visitatore, la guida rimane in attesa.
- Il visitatore deve essere registrato nella lista di chi ha acquistato il biglietto con guida per essere rilevato.
- Il visitatore deve trovarsi entro il raggio di rilevamento della guida.

### 3. **GuideInitializerTour**

- Questo script inizializza il tour guidato, preparando la guida per la visita delle stanze.
- Inizializza gli attributi della classe *GuideTourData* e stabilisce l'ordine delle stanze da visitare.

### 4. **GuideNextStand**

- Questo script determina la prossima opera del tour da visitare.
- Il nodo è parte del ciclo iterativo "*Repeat Until Success*".
- Utilizza la ricerca A\* per selezionare il percorso più breve verso l'opera successiva. Durante il tour, lo script seleziona casualmente l'identificativo associato a un visitatore dandogli la possibilità di porre una domanda, mentre gli altri rimangono in silenzio. Una volta raggiunta l'opera, il controllo passa al nodo successivo.

### 5. **WaitForQuestion**

- Questo script pone la guida in uno stato di attesa fino a quando non riceve una domanda da un visitatore o dall'utente.
- L'implementazione prevede che la guida attenda un intervallo di tempo prestabilito per ricevere una domanda posta dall'utente.
- Nel caso in cui questo intervallo di attesa termini, la guida ascolterà una domanda posta dall'agente visitatore.
- La guida esce dallo stato di attesa solo dopo che le viene posta una domanda. Il nodo è parte del ciclo iterativo "*Repeat Until Success*".

### 6. **MakeHttpRequest**

- Questo script gestisce la comunicazione tra la guida e il modello di linguaggio tramite una richiesta HTTP POST.
- La guida rimane in attesa fino a quando la risposta del modello viene mostrata in una finestra di dialogo.
- Il nodo è parte del ciclo iterativo "*Repeat Until Success*".

**7. CheckUserRequest**

- Questo script verifica se la richiesta posta dal visitatore o dall'utente restituisce la posizione dall'ontologia.
- Se la risposta del modello di linguaggio restituisce la posizione, lo script veicola il flusso di esecuzione verso la ripianificazione.
- Se la posizione non viene restituita, il flusso continua a seguire il piano inizialmente generato.

**8. ReplanPath**

- È lo script del nodo che riformula il piano dell'itinerario in base alla richiesta del visitatore.
- Inizialmente, la guida genera un percorso dalla sua posizione attuale fino all'opera da raggiungere.
- Una volta giunta all'opera, la guida esegue la ripianificazione basandosi sulla sua posizione attuale.
- Il nuovo itinerario viene formulato considerando solo le opere che non sono ancora state mostrate ai visitatori.
- Il processo di ripianificazione per la generazione del nuovo itinerario è guidato dalla ricerca A\*.

**9. CheckUserTerminatesInteraction:**

- È lo script del nodo che verifica se l'utente ha posto una domanda entro il tempo limite di attesa della guida.
- Se l'utente non ha posto alcuna domanda entro il tempo stabilito, la guida prosegue con l'itinerario pianificato ed esce dal nodo "*Repeat Until Failed*".
- Se l'utente pone una domanda entro il tempo limite, la guida rimane in attesa e resta immobile nella sua posizione.

**10. CheckInteractiveRoom**

- Questo script verifica se la prossima stanza da visitare è la stanza interattiva.
- Gestisce l'uscita dal nodo "*Repeat Until Success*".
- Se la stanza è quella interattiva, il nodo esce dal ciclo "*Repeat Until Success*", altrimenti ripete i nodi *GuideNextStand*, *WaitForQuestion*, e *MakeHttpRequest*.

**11. EndTour**

- Questo script permette alla guida di terminare il tour con i visitatori attuali.
- La guida annuncia che i visitatori possono esplorare liberamente la stanza interattiva.

**12. GuideComeBackToRestartPoint**

- Questo script riporta la guida al punto iniziale del tour.

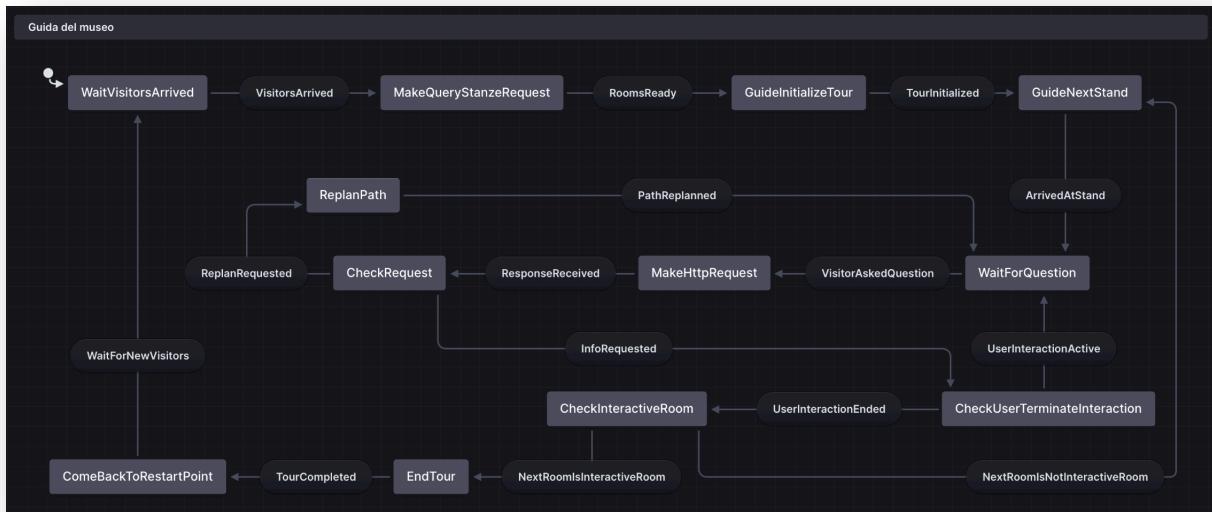
- Sfrutta i metodi della classe *DialogManager* come la comparsa, la distruzione e l'aggiornamento della posizione della finestra.
- Una volta tornata al punto di partenza, la guida riprende il processo dal nodo *WaitVisitors*.

### 13. **GuideDialog**

- Questa classe è adibita alla gestione delle finestre di dialogo tra la guida e il visitatore.
- Sfrutta i metodi della classe *DialogManager*, come la comparsa, la distruzione e l'aggiornamento della posizione della finestra.
- Gestisce l'interazione tra guida e visitatore, assicurando che i messaggi vengano mostrati nei momenti appropriati durante il tour.

## *FSM*

Una FSM (Finite State Machine) è un modello computazionale utilizzato per rappresentare e gestire i comportamenti di un sistema che può trovarsi in uno di un numero finito di stati. Ogni stato rappresenta una condizione specifica del sistema e le transizioni rappresentano le azioni che cambiano lo stato attuale. Questa FSM rappresenta il flusso di controllo di una guida museale automatizzata, gestendo l'interazione con i visitatori e la navigazione tra le diverse stanze del museo. Il processo inizia con lo stato *WaitVisitorsArrived*, in cui il sistema attende l'arrivo dei visitatori. Una volta presenti, viene effettuata una richiesta per ottenere informazioni sulle stanze disponibili (*MakeQueryStanzeRequest*). Dopo aver ricevuto i dati necessari, la guida avvia il tour e procede verso la prima postazione (*GuideNextStand*). Quando il sistema raggiunge una postazione, attende eventuali domande dei visitatori. Se viene posta una domanda, viene effettuata una richiesta HTTP per recuperare le informazioni pertinenti. Durante il tour il sistema può verificare se è necessario ricalcolare il percorso (*ReplanPath*) o controllare se la stanza successiva è interattiva (*CheckInteractiveRoom*). Se l'interazione con i visitatori è attiva, il sistema attende la fine della stessa prima di procedere. Al termine del tour la guida ritorna al punto di partenza e si mette in attesa di nuovi visitatori, riprendendo il ciclo dall'inizio. Questa FSM consente un controllo strutturato ed efficiente del processo di guida museale, garantendo un'interazione dinamica con i visitatori e una gestione ottimale del percorso all'interno del museo.



*Bigliettaio:*



Il bigliettaio è l'unico agente nell'ambiente che non si sposta dalla posizione iniziale. La sua implementazione si basa su tre principali script, ognuno con una funzione specifica:

#### 1. **TicketBoothInteraction**

- Questo script gestisce lo scambio di messaggi tra il bigliettaio e i visitatori.

- Funziona come un contenitore, memorizzando sia il messaggio trasmesso tra gli agenti sia la scelta del tipo di biglietto effettuata dal visitatore.
- Include metodi per ripulire i dati memorizzati, rimuovendo messaggi e scelte precedenti per consentire nuove interazioni.

## 2. **TicketBoothRespond**

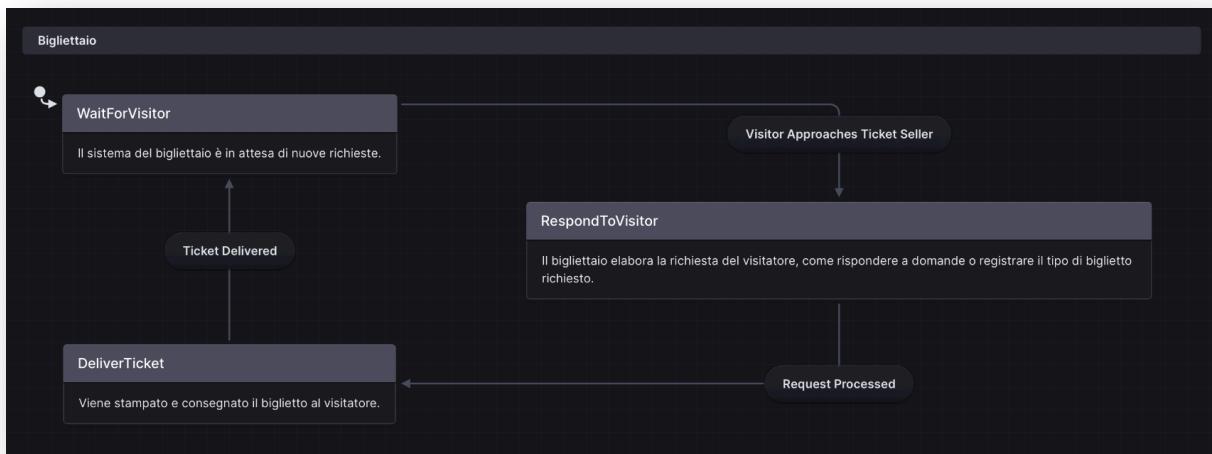
- Questo script regola la comunicazione del bigliettaio con i visitatori che si avvicinano al bancone. Pone il bigliettaio in uno stato di attesa fino a quando un visitatore non gli rivolge la parola. Una volta ricevuto il messaggio, aggiorna il valore dell'attributo corrispondente nella classe `TicketBoothInteraction` e consente al bigliettaio di rispondere.

## 3. **DeliverTicket**

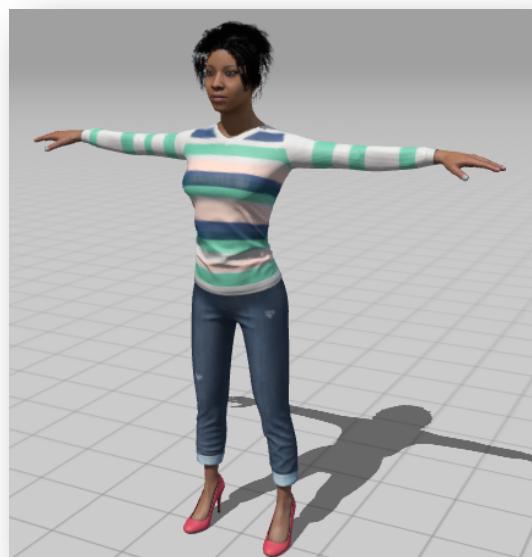
- Questo script consente al bigliettaio di consegnare il biglietto al visitatore.
- Gestisce una lista di registrazione dei visitatori che decidono di acquistare il biglietto con il tour guidato. La lista viene aggiornata ogni volta che un visitatore effettua questa scelta, permettendo alla guida di sapere a quali visitatori prestare attenzione quando questi le pongono domande.
- La registrazione dei visitatori avviene tramite la memorizzazione di un identificativo numerico associato a ciascun visitatore.

## *FSM*

La FSM descrive in modo organizzato il funzionamento del bigliettaio, suddividendo il processo in tre fasi principali. Il ciclo inizia nello stato di attesa, denominato *WaitForVisitor*, in cui il sistema è pronto a rilevare l'approccio di un visitatore. Durante questa fase, il sistema rimane inattivo fino a quando non viene rilevata una richiesta. Quando un visitatore si avvicina, il sistema transita nello stato *RespondToVisitor*, dove analizza la richiesta, risponde a eventuali domande e registra il tipo di biglietto desiderato o altre informazioni necessarie per completare l'operazione. Completata questa fase, il sistema passa allo stato *DeliverTicket*, in cui stampa e consegna il biglietto al visitatore. Dopo la consegna del biglietto, il sistema ritorna allo stato iniziale *WaitForVisitor*, chiudendo il ciclo e preparandosi per gestire una nuova interazione. Questo flusso è progettato per garantire un'interazione fluida, intuitiva ed efficiente con i visitatori, assicurando un alto livello di precisione nella gestione delle richieste e nella consegna dei biglietti.



*Visitatori:*





I visitatori sono gli agenti progettati per interagire attivamente con l'ambiente del museo. La loro implementazione prevede un comportamento dinamico, regolato da diversi script, che permette loro di esplorare il museo in modo realistico. Gli script utilizzati per gestire il comportamento dei visitatori sono i seguenti:

#### 1. **VisitorSpawner**

- Questo script simula l'ingresso del visitatore nel museo.
- Attiva il visitatore nella scena dopo un intervallo di tempo casuale, se il parametro di Spawn Time non è specificato, rendendo l'esperienza più dinamica e simulando che il visitatore sia entrato concretamente nel museo.

#### 2. **VisitorMoveToTicketBooth**

- Gestisce il movimento del visitatore verso il bigliettaio. Permette al visitatore di avvicinarsi al bancone per iniziare l'interazione.

#### 3. **VisitorSendMessage**

- Consente al visitatore di rivolgersi al bigliettaio una volta raggiunto il bancone, mostrando una finestra di dialogo.
- Attiva lo scambio di messaggi tra i due agenti.

#### 4. **VisitorMakeChoice**

- Permette al visitatore di scegliere tra un biglietto con guida o senza guida.
- La scelta può essere specificata a priori, impostandola manualmente, o è generata casualmente se non è specificata.

#### 5. **CheckTicketChoice**

- Verifica il tipo di biglietto acquistato dal visitatore.
- In base alla scelta, determina il comportamento successivo del visitatore, indirizzandolo lungo uno dei due percorsi comportamentali disponibili:
  - con guida
  - senza guida

### Percorso con Guida

#### 6. **FollowGuide**

- Consente al visitatore di seguire la guida durante l’itinerario.
- Lo script rimane attivo fino a quando la guida non raggiunge un’opera. È contenuto all’interno di un nodo “*Repeat Until Failed*”, che permette al visitatore di continuare a seguire la guida dopo aver posto una domanda.

#### 7. **CheckEndedTour**

- Permette al visitatore di recepire quando la guida annuncia la fine del tour.
- Lo script è contenuto all’interno di un nodo “*Repeat Until Failed*”.

#### 8. **VisitorAskQuestion**

- Consente al visitatore di porre una domanda pertinente all’opera mostrata dalla guida.
- Definisce due comportamenti distinti:
  1. Se il visitatore non è stato selezionato per formulare la domanda, rimane immobile e ascolta la risposta fornita dalla guida.
  2. Se il visitatore è quello selezionato per porre la domanda: i quesiti sono predefiniti e il meccanismo con il quale vengono poste è gestito per mezzo di un dizionario di domande predefinite associate a ciascuna opera per estrarre una domanda casuale ma coerente.
- L’esecuzione termina solo quando la guida ha fornito una risposta. Il nodo è contenuto all’interno di un nodo “*Repeat Until Failed*”.

#### 9. **VisitLastRoom**

- Avvia l’esplorazione della sala interattiva una volta che la guida ha concluso il tour.
- La visita delle opere nella sala avviene in modo casuale per dare maggiore dinamicità all’ambiente. Quando il visitatore si trova davanti a un’opera interattiva, interagisce con essa innescando un’animazione.
- L’esecuzione dello script si conclude dopo che il visitatore ha esplorato tutte le opere della sala.

## Percorso senza Guida

### 7. **ExploreMuseum**

- Permette al visitatore di esplorare liberamente il museo seguendo un percorso personalizzato.
- **Logica del percorso:**
  1. Il visitatore seleziona casualmente una sala da visitare.
  2. All'interno della sala viene utilizzato l'algoritmo di ricerca A\* per identificare l'opera più vicina alla posizione attuale del visitatore. Le posizioni delle opere, contenute nella sala selezionata, vengono raccolte in una lista.
  3. Il visitatore si avvicina all'opera selezionata e la ammira per un intervallo di tempo casuale.
  4. L'opera appena visitata viene rimossa dalla lista delle opere nella sala, e il processo si ripete fino a esaurire le opere disponibili.
  5. Una volta completata la visita di una sala, il visitatore seleziona un'altra sala e ripete lo stesso processo fino a completare l'intero itinerario museale.

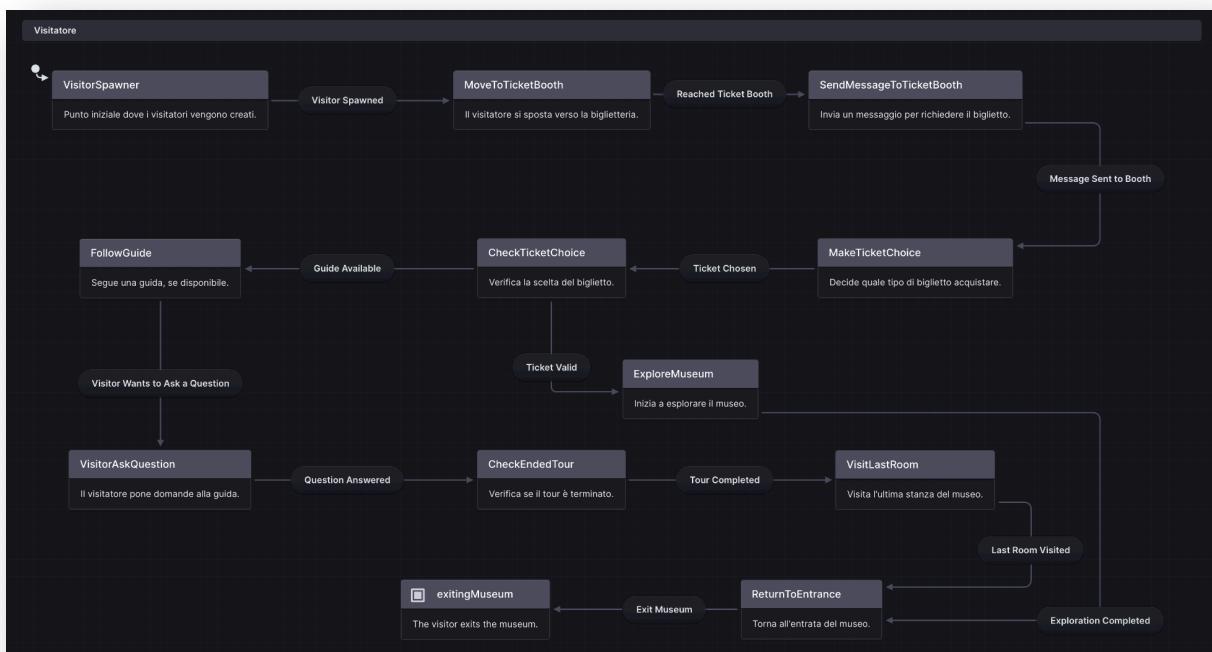
### 8. **ReturnToEntrance**

- Script che permette al visitatore di tornare all'ingresso del museo dopo aver visitato tutte le sale.
- Una volta raggiunto l'ingresso l'agente scompare, simulando l'uscita dal museo.
- Viene eseguito dopo tutti gli altri nodi a prescindere dai due rami comportamentali.

## *FSM*

Questa FSM rappresenta il percorso tipico di un visitatore all'interno di un museo suddividendolo in una serie di stati che descrivono le diverse fasi dell'esperienza. L'obiettivo è mappare ogni interazione del visitatore con il sistema museale, dalla sua entrata fino all'uscita, garantendo un flusso organizzato e chiaro. Il processo inizia con lo stato *VisitorSpawner* dove il visitatore viene generato, rappresentando simbolicamente il suo arrivo al museo. Da qui il visitatore si sposta verso la biglietteria nello stato *MoveToTicketBooth*. Una volta raggiunta la biglietteria, il visitatore invia una richiesta per acquistare un biglietto passando così allo stato *SendMessageToTicketBooth*. Questo rappresenta l'interazione iniziale del visitatore con il sistema in cui viene inoltrata una richiesta specifica. Dopo aver inviato la richiesta, il visitatore entra nello stato *MakeTicketChoice* dove seleziona il tipo di biglietto desiderato che viene successivamente verificato nello

stato *CheckTicketChoice*. Se la scelta è valida, il visitatore può proseguire il suo percorso e accedere alle esposizioni, entrando nello stato *ExploreMuseum*, che rappresenta la fase centrale della visita. Durante l'esplorazione, il visitatore ha l'opportunità di seguire una guida nello stato *FollowGuide* o di interagire ulteriormente ponendo domande alla guida nello stato *VisitorAskQuestion*. Man mano che il visitatore esplora le varie sezioni del museo raggiunge l'ultima stanza nello stato *VisitLastRoom* completando così il tour. A questo punto lo stato *CheckEndedTour* verifica se l'esperienza è terminata. Se il tour è completo, il visitatore ritorna verso l'ingresso nello stato *ReturnToEntrance*. L'esperienza si conclude nello stato finale, *ExitingMuseum*, dove il visitatore lascia fisicamente il museo, segnando la fine del ciclo. Ogni stato e transizione della FSM rappresentano un passo ben definito, progettato per garantire una visita organizzata, intuitiva e senza interruzioni.



## Hardware

Per rendere l'esperienza il più realistico possibile, si è scelto di rappresentare gli agenti con l'aspetto di robot umanoidi. Essendo un ambiente simulato, l'implementazione del progetto non ha richiesto la gestione di componenti hardware come sensori e attuatori. Tuttavia, è interessante considerare i principali componenti hardware che potrebbero essere impiegati in una possibile realizzazione concreta del sistema.

### 1. Sensore di Prossimità Laser

- **Funzione:** consente al robot di muoversi in modo efficiente evitando ostacoli statici e dinamici, fondamentale in un museo affollato.
- **Scopo:** rilevare con precisione l'ambiente circostante e prevenire collisioni con i visitatori o gli oggetti esposti.
- **Caratteristica:** Essendo un sensore percettivo attivo, emette segnali per rilevare la posizione e la distanza degli ostacoli, garantendo una risposta immediata.
- **Posizionamento:** installato all'altezza della vita dello scheletro per garantire una copertura ottimale del campo visivo.

## 2. Speaker

- **Funzione:** permette al robot di comunicare tramite sintesi vocale.
- **Scopo:** fornire spiegazioni sulle opere, rispondere a domande e interagire con i visitatori.
- **Posizionamento:** collocato all'altezza della bocca per simulare una comunicazione naturale.

## 3. Microfono Direzionale

- **Funzione:** cattura la voce dei visitatori che si rivolgono al robot.
- **Scopo:** garantisce che le domande dei visitatori siano recepite chiaramente, anche in presenza di rumore di fondo, utilizzando algoritmi per la riduzione del rumore.
- **Posizionamento:** posizionato sul busto per captare efficacemente le voci provenienti dal fronte.

## 4. Sensore di Riconoscimento dell'Interlocutore

- **Funzione:** permette al robot di distinguere se sta interagendo con il bigliettaio, con la guida o con il visitatore, adattando il proprio comportamento e le risposte di conseguenza.
- **Scopo:** garantire un'interazione precisa e contestualizzata con gli altri agenti presenti nell'ambiente del museo.
- **Caratteristica:** Essendo un sensore percettivo passivo, raccoglie informazioni dall'ambiente senza emettere segnali attivi, analizzando dati già presenti come immagini e suoni.
- **Funzionamento:** la *camera RGB-D* rileva la posizione fisica degli interlocutori nell'ambiente, confrontandola con le posizioni note del bigliettaio, della guida o del visitatore. Il *microfono array* analizza la direzione di provenienza della voce e, tramite algoritmi di riconoscimento vocale, associa il timbro dell'interlocutore al profilo noto

(bigliettaio, guida o visitatore). I dati raccolti vengono elaborati dall'unità di controllo per identificare con precisione chi sta interagendo con il robot.

- **Posizionamento:** la *camera RGB-D* viene installata nella testa o nel busto del robot per monitorare la posizione e la distanza degli interlocutori. Il *microfono array* viene collocato nella parte superiore del busto per catturare in modo chiaro le voci provenienti da diverse direzioni.

## 5. Motore Elettrico

- **Funzione:** abilita il movimento del robot.
- **Scopo:** fornire energia per la locomozione e il funzionamento degli arti.
- **Alimentazione:** collegato a una batteria ricaricabile per garantire autonomia operativa.

## 6. Unità di Controllo

- **Funzione:** gestisce tutte le funzionalità e i comportamenti del robot.
- **Scopo:** elaborare comandi di movimento, ricevere riscontri dai sensori e adattare il comportamento in tempo reale.
- **Implementazione:** utilizza un microcontrollore, ad esempio una *Raspberry Pi*, con il sistema operativo ROS (Robot Operating System) per la gestione della navigazione e il controllo degli attuatori.

## 7. Controller degli Arti

- **Funzione:** riceve comandi dall'unità di controllo e coordina il movimento degli arti.
- **Scopo:** garantisce movimenti precisi e fluidi, essenziali per un'interazione naturale.

## 8. Giunti Articolati

- **Funzione:** abilitano il movimento fluido e realistico degli arti.
- **Scopo:** consentono al robot di gesticolare durante le spiegazioni, simulare una camminata naturale e interagire in modo espressivo con i visitatori.

# Movimento degli agenti

Per garantire il corretto funzionamento della simulazione, è stata posta particolare attenzione al movimento degli agenti nella scena. La navigazione è stata implementata utilizzando l'asset Navigation AI combinato con il sistema NavMesh. Questa integrazione permette di gestire efficacemente gli spostamenti degli agenti evitando collisioni e garantendo percorsi ottimali.

## NavMesh

Il NavMesh è una rappresentazione astratta dei percorsi percorribili dagli agenti, basata su una rete di poligoni che definisce le aree transitabili. Questa rete viene generata analizzando la geometria della scena e considerando parametri come:

- **Presenza di ostacoli** (mobili, oggetti decorativi, ecc.).
- **Altezza degli agenti** (per evitare passaggi troppo bassi).
- **Inclinazione del terreno** (in questo caso è perfettamente piana).

In base alla configurazione generata, il NavMesh consente di calcolare i percorsi migliori per raggiungere una destinazione specifica. Sia la guida che i visitatori possono sfruttare questa struttura per evitare ostacoli statici (come l'arredamento) e dinamici (altri agenti in movimento). Questa funzionalità è particolarmente utile in situazioni di affollamento o quando le traiettorie di due agenti si sovrappongono.

## Ricerca A\*

Oltre al movimento gestito da Navigation AI, viene utilizzato l'algoritmo di ricerca A\* per ottimizzare ulteriormente il calcolo dei percorsi all'interno di un grafo.

- **Nodi:** rappresentano le opere presenti nelle sale.
- **Archi:** indicano i possibili tragitti definiti dal NavMesh.

L'algoritmo A\* utilizza un'euristica per rendere il calcolo dei percorsi più efficiente. In questo contesto, l'euristica è rappresentata dalla distanza euclidea tra la posizione dell'agente e i punti di interesse (le opere). La funzione di valutazione di A\* combina due elementi principali:

1. **Costo accumulato:** rappresenta la distanza totale percorsa dal nodo iniziale (posizione iniziale dell'agente) al nodo corrente (posizione attuale dell'agente), considerando il cammino già effettuato.
2. **Euristica:** stima la distanza tra il nodo corrente e il nodo di destinazione (opera). È fondamentale che questa stima sia ammissibile, ovvero non sovrastimi la distanza reale, per garantire risultati ottimali.

## Superficie Percorribile

Il NavMesh genera una mappa visiva delle aree percorribili all'interno della scena, evidenziate in azzurro.



## Gestione degli Oggetti nella Scena

- Gli oggetti statici (es. mobili, decorazioni) sono associati a una componente *NavMesh Obstacle* che li rende riconoscibili come ostacoli fissi.
- Gli agenti dinamici (visitatori e guida) sono associati a una componente *NavMesh Agent* che consente loro di navigare la scena e comportarsi come ostacoli mobili per gli altri agenti.

Questa configurazione assicura una navigazione fluida, evitando collisioni e garantendo un'esperienza realistica anche in presenza di più agenti che interagiscono simultaneamente.

## Filtro di Kalman

Uno strumento fondamentale per stimare lo stato di un sistema dinamico è il filtro di Kalman, il quale combina i dati provenienti da un modello matematico con quelli campionati dai sensori, tenendo conto dell'incertezza di entrambe le fonti soggette a rumore. Nel contesto del progetto, il filtro di Kalman viene utilizzato per stimare la posizione e la velocità dell'agente durante il movimento verso un punto di interesse, fornendo un segnale quando l'agente ha raggiunto la destinazione. A ogni frame della scena il filtro elabora la nuova posizione dell'agente, monitorando costantemente se l'obiettivo è stato raggiunto. L'integrazione del filtro è motivata dalla staticità dei punti di interesse all'interno dell'ambiente: poiché le posizioni delle opere d'arte sono note con precisione, è possibile confrontarle con la stima fornita dal filtro. Se la differenza tra la stima e la posizione obiettivo rientra entro una determinata soglia, si considera che l'agente abbia raggiunto la destinazione. L'implementazione del filtro di Kalman si articola in tre fasi principali:

1. **Inizializzazione:** in cui vengono definite le matrici necessarie per i calcoli.
2. **Predizione:** viene stimato a priori lo stato del sistema e l'incertezza associata alla predizione.
3. **Aggiornamento:** fase in cui la stima predetta viene confrontata con le nuove misurazioni fornite dai sensori al fine di migliorare l'accuratezza complessiva.

I dati utilizzati nell'algoritmo *KalmanFilter* includono:

- $F \rightarrow$  matrice di transizione
- $H \rightarrow$  matrice di misura
- $Q \rightarrow$  matrice di covarianza del processo
- $R \rightarrow$  matrice di covarianza di misura
- $P \rightarrow$  matrice di covarianza iniziale dell'errore
- $x \rightarrow$  stima iniziale dello stato

Le grandezze considerate nel filtro di Kalman, così come definite, non includono la componente  $y$  che rappresenterebbe lo spostamento verticale dell'agente. Questa scelta è giustificata dal fatto che l'ambiente di riferimento presenta una pavimentazione piana, rendendo superflua la gestione della dimensione verticale.

Nella fase di predizione il comportamento del sistema è descritto attraverso due equazioni fondamentali:

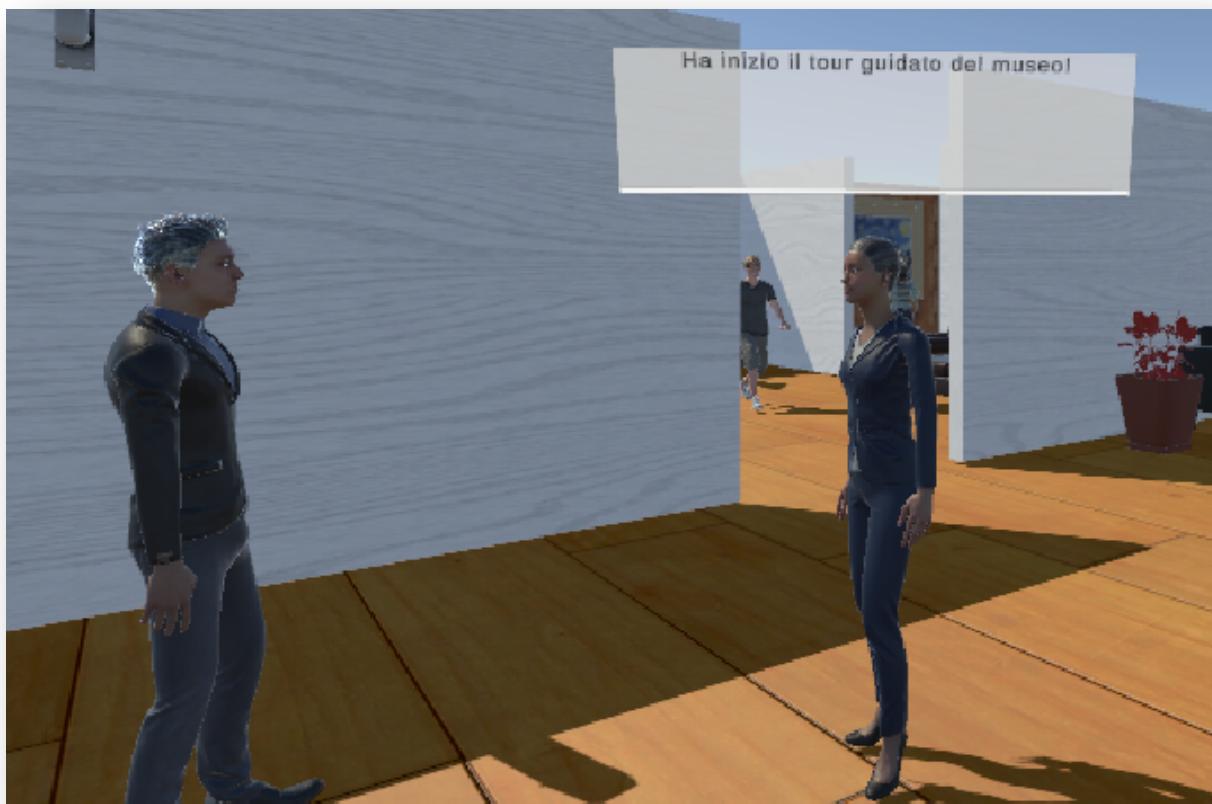
$$\bullet \quad x_{predetto} = Fx$$

- $P_{predetto} = FPF^T + Q$

Nella fase di aggiornamento:

- $z = \{misura sulla x, misura sulla z\}$ 
  - $y = z - Hx_{pred}$
  - $S = HPH^T + R$
  - $K = P_{pred}H^TS^{-1}$
  - $x = x_{pred} + Ky$
  - $P = P_{pred} - KHP_{pred}$

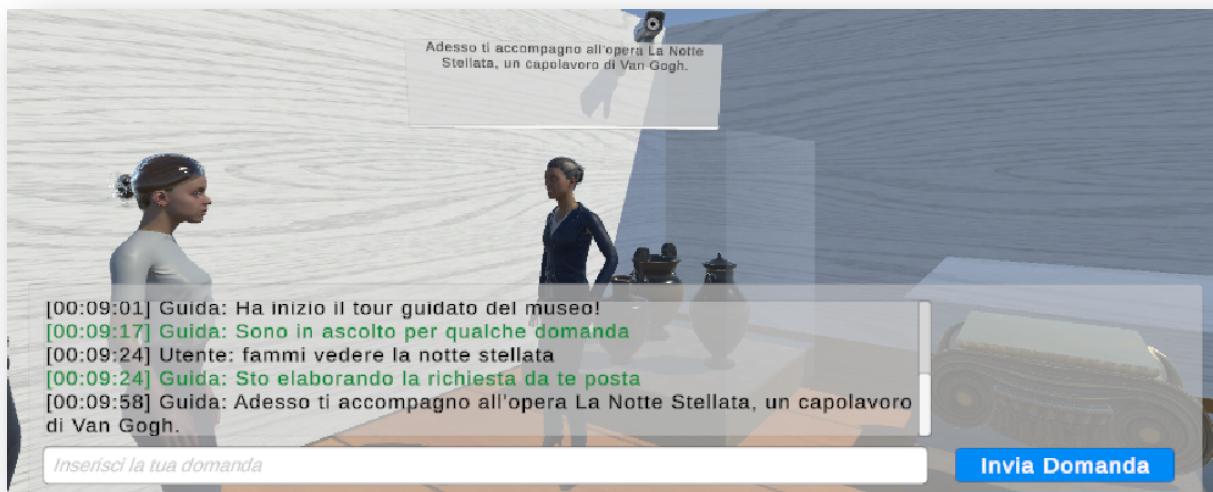
## Pianificazione dell'itinerario



Uno dei comportamenti fondamentali di un agente è la pianificazione del moto, ovvero la capacità di determinare un percorso che gli permetta di raggiungere una destinazione specifica all'interno dell'ambiente in cui opera, rispettando una serie di vincoli. Tra questi rientra la necessità di evitare ostacoli statici, garantendo che il movimento avvenga in sicurezza e senza collisioni. In un

ambiente semidinamico, come quello di un museo, la pianificazione del percorso avviene stabilendo un punto di partenza, corrispondente alla posizione attuale dell'agente, e un punto di arrivo, che coincide con una specifica opera d'arte o punto di interesse. A partire da questi due estremi, l'agente costruisce una sequenza di configurazioni che, se eseguite nell'ordine corretto, consentono di collegare in modo efficace le due posizioni. L'itinerario più breve per l'intero percorso viene calcolato al termine di un periodo di attesa, necessario per consentire l'arrivo di nuovi visitatori. Una volta completata questa fase, l'agente avvia il tour guidato. Poiché le opere d'arte non cambiano posizione nel tempo, consentono all'agente di partire sempre da un punto iniziale prestabilito per avviare il tour. In base alla stanza di partenza scelta, l'itinerario viene pianificato in modo da essere il più efficiente possibile, ottimizzando i tempi e garantendo un percorso rapido e comodo per completare il tour.

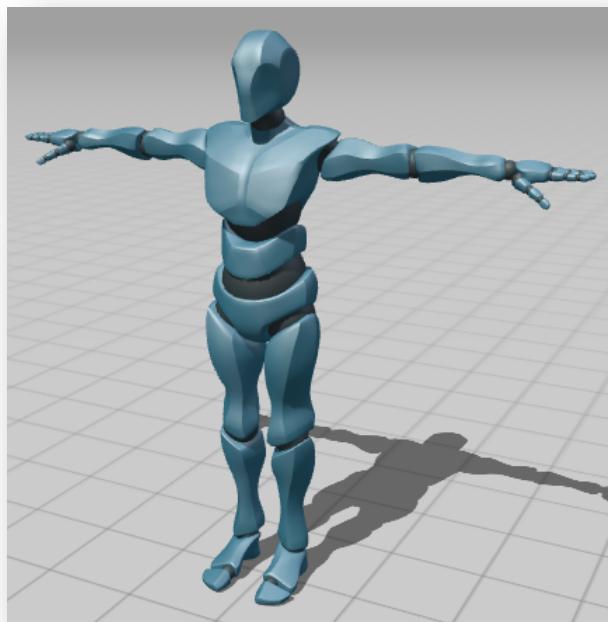
## Ripianificazione dinamica dell'itinerario



La ripianificazione è un comportamento essenziale che conferisce flessibilità all'agente, consentendogli di adattarsi alle richieste dei visitatori e garantire loro la migliore esperienza possibile. Il processo di ripianificazione consiste nel generare un nuovo percorso che l'agente dovrà seguire, in risposta a una richiesta di deviazione da parte di un visitatore desideroso di ammirare una specifica opera all'interno del museo. In questi casi, l'agente interrompe l'itinerario precedentemente pianificato e guida i visitatori verso l'opera richiesta, a condizione che essa sia presente nel museo. Una volta raggiunta l'opera e soddisfatte eventuali domande dei visitatori, l'agente procede a ricalcolare il percorso partendo dalla posizione attuale e tenendo conto delle opere già visitate. Questo accorgimento evita di riproporre opere già ammirate, garantendo un tour ottimizzato e vario in grado di valorizzare al meglio l'esperienza complessiva del pubblico.

## Modellazione dell'utente

L'utente è stato progettato per assumere sembianze umanoidi, analogamente agli altri agenti presenti nella scena. Tuttavia, il suo aspetto è volutamente differenziato per enfatizzare il GameObject che lo guida all'interno dell'ambiente. A differenza dei visitatori, della guida e del bigliettaio, che presentano caratteristiche più realistiche, l'utente ha un aspetto più stilizzato, simile a quello di un manichino, al fine di evidenziare il suo ruolo specifico all'interno della simulazione.



L'utente è dotato di una componente *Character Controller* che definisce un'area specifica attorno all'avatar, rendendolo sensibile al contatto con ostacoli sia statici che dinamici. Il movimento dell'utente è gestito dallo script *Avatar Controller* che permette di esplorare lo scenario e di ruotare la visuale durante la simulazione tramite l'uso della tastiera.

### Elenco comandi:

- **W:** spostamento in avanti
- **S:** spostamento all'indietro
- **A:** spostamento laterale verso sinistra
- **D:** spostamento laterale verso destra
- **Q:** rotazione verso sinistra sull'asse orizzontale
- **E:** rotazione verso destra sull'asse orizzontale

Il punto di vista dell'utente è gestito tramite la Main Camera, un GameObject speciale fornito nativamente da Unity che consente di definire la visuale all'interno della scena. La Main Camera è posizionata all'altezza degli occhi dell'avatar, garantendo così una prospettiva realistica, simile a quella dell'esperienza reale. Per sincronizzare i movimenti dell'avatar con quelli della visuale dell'utente, la Main Camera è impostata come GameObject figlio dell'avatar, assicurando che ogni spostamento dell'avatar venga accompagnato dal corrispondente aggiornamento della prospettiva.

## Modellazione interfaccia utente

Nel corso dell'intera simulazione l'utente interagisce con la scena in modo simile agli altri agenti visitatori, fatta eccezione per alcune specifiche differenze. Per consentire all'utente di interagire con l'ambiente circostante, in punti predefiniti della scena possono apparire delle interfacce utente dedicate. Queste interfacce, di diversa tipologia, offrono funzionalità specifiche a seconda del contesto, permettendo all'utente di accedere a informazioni, compiere azioni o influenzare l'ambiente virtuale in modi mirati.

### *Choose Ticket UI*



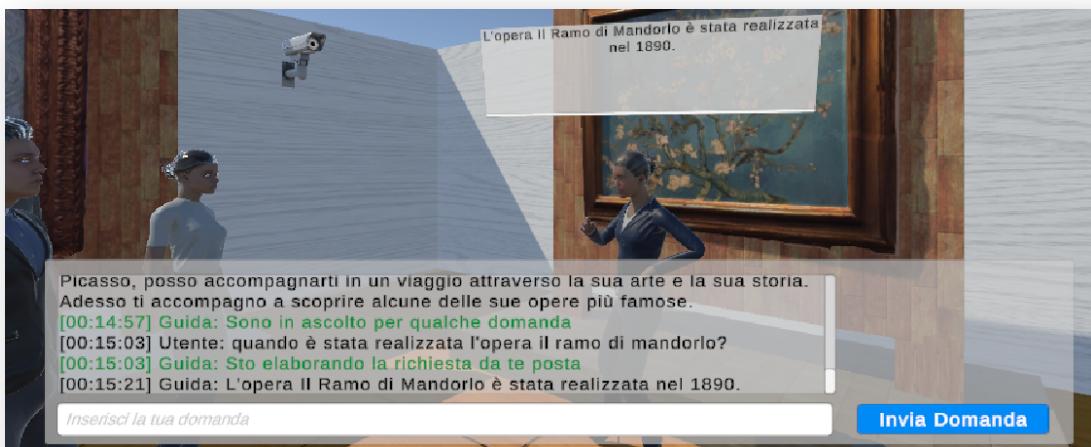
L'interfaccia è progettata per guidare l'utente nella scelta tra due tipologie di biglietto. Utilizzando il cursore, l'utente può selezionare una delle due opzioni disponibili, ottenendo così l'accesso a un tour guidato o a un tour personalizzato. Questa interfaccia compare automaticamente quando l'utente si avvicina all'agente Bigliettaio, raggiungendo una distanza minima predefinita, e rimane visibile fino a quando non viene effettuata una scelta. Se l'utente si allontana dal Bigliettaio senza aver selezionato alcuna opzione, l'interfaccia scompare temporaneamente. Tuttavia, riavvicinandosi nuovamente entro la distanza minima, l'interfaccia viene ripristinata consentendo all'utente di completare la selezione in un secondo momento.

## Block Enter UI

**Non puoi visitare il museo senza avere un biglietto. Recati dal bigliettaio e scegli la tipologia di biglietto.**

L'interfaccia di allerta è progettata per avvisare l'utente nel caso in cui tenti di accedere al corridoio senza essere in possesso di un biglietto. Questo avviso compare automaticamente non appena l'utente prova a varcare l'ingresso del corridoio. A supporto di questa interfaccia è presente una barriera invisibile che impedisce il passaggio dalla sala d'ingresso al corridoio e, di conseguenza, all'intero museo. La barriera rimane attiva finché l'utente non seleziona una delle due modalità di biglietto disponibili. Una volta effettuata la scelta, la barriera viene disattivata consentendo all'utente di proseguire e accedere liberamente alle altre aree del museo.

## Question UI



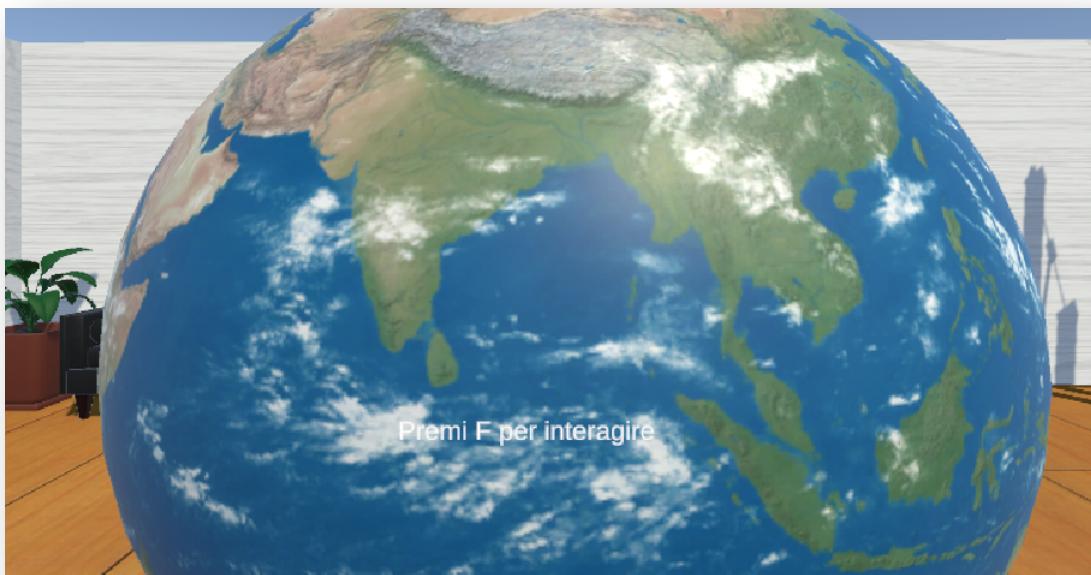
Una delle interfacce principali, caratterizzata dai seguenti elementi:

- **Campo di inserimento testo:** consente all'utente di digitare le domande da porre alla guida.
- **Tasto "Invia Domanda":** permette di inviare la domanda alla guida. Il tasto è attivo solo quando l'utente ha selezionato il biglietto con guida.
- **Finestra scorrevole:** un prompt che tiene traccia di tutte le domande poste alla guida, aggiungendo un timestamp a ciascuna. Inoltre, visualizza le informazioni su come opera

l'agente guida. Include una barra laterale destra per lo scorrimento manuale e un meccanismo di scorrimento automatico verso il basso quando il numero massimo di domande da mostrare è raggiunto.

- **Pulsante "C":** premendo il tasto "C" della tastiera l'interfaccia può essere mostrata o nascosta alternativamente. Questa scelta è stata fatta per permettere all'utente di avere una visione più libera delle opere esposte senza che l'interfaccia oscuri la visuale durante la visita.

## *Interaction UI*



Un'interfaccia che suggerisce all'utente con quali opere tra quelle presenti può interagire, indicando anche il pulsante da premere per attivare l'interazione.

## Finestre di dialogo

Per visualizzare le conversazioni tra gli agenti all'interno della scena, è stato sviluppato un sistema che consente la comparsa di finestre di dialogo direttamente vicino agli agenti coinvolti nella conversazione. Questo approccio offre un riscontro visivo immediato, facilitando la comprensione del contesto della simulazione.



L'aspetto grafico delle finestre di dialogo è stato realizzato utilizzando la componente *Canvas* nativa di Unity che permette la creazione di interfacce utente personalizzabili. Grazie a questa componente, è possibile modificare la formattazione del testo e dimensionare le finestre in modo da garantire una visualizzazione chiara e intuitiva. Le finestre sono progettate con un design semplice: un rettangolo semitrasparente che contiene il testo delle conversazioni degli agenti. Per ottimizzare la leggibilità, le finestre di dialogo sono orientate automaticamente verso la *Main Camera* anche quando quest'ultima è in movimento, assicurando una lettura agevole in ogni situazione. La gestione delle finestre di dialogo è affidata alla classe *DialogManager* che fornisce i metodi necessari per la comparsa e la distruzione dinamica delle finestre all'interno della scena. Questo sistema garantisce un controllo flessibile e scalabile delle interazioni visive durante la simulazione.

## Conclusioni e Sviluppi Futuri

La progettazione e realizzazione di un agente guida per il museo ha rappresentato un'esperienza impegnativa ma al tempo stesso altamente stimolante. Questo progetto ci ha offerto l'opportunità di applicare le conoscenze acquisite in ambito di robotica e intelligenza artificiale durante il percorso di studi, affrontando un problema concreto e reale. Si tratta di un ambito che, sebbene ancora poco sviluppato nelle applicazioni odierne, offre spunti interessanti e merita ulteriori approfondimenti da diverse prospettive. L'integrazione delle due discipline (Robotica e Intelligenza Artificiale 2) ci ha permesso di sviluppare una metodologia di lavoro innovativa, mirata a coniugare gli aspetti fondamentali di entrambe le aree in un sistema funzionale, coeso e avanzato.

Per rendere il progetto ancora più competitivo, le future evoluzioni potrebbero concentrarsi su diversi obiettivi, tra cui:

- **Migliorare l'interazione uomo-macchina** rendendola più naturale e intelligente per affinare la comunicazione tra l'utente e l'agente guida.
- **Incrementare l'immersività dell'esperienza utente** attraverso lo sviluppo di una versione compatibile con la realtà virtuale (VR).
- **Ampliare il numero di opere disponibili**, includendo sia elementi statici che interattivi, per arricchire l'esperienza museale.
- **Rendere le interazioni con le opere interattive più realistiche**, favorendo un coinvolgimento emotivo e cognitivo maggiore da parte degli utenti.

Questi sviluppi rappresentano un passo fondamentale per consolidare l'efficacia del progetto, spingendo i confini dell'innovazione tecnologica applicata al mondo dei musei.