UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica

Universitat de Barcelona

Interaccions Conversacionals a un Joc Seriós

Autor: Carles Toujouse Machado

Directora: Dra. Inmaculada Rodríguez Santiago

Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 27 de juny de 2018

Abstract

This project is located, on one hand, in the field of the conversationa l'agents, artificial intelligences prepared and focused on maintaining dialogues with a user, and on the other hand, in the field of video games, specifically integrating these conversationa l'agents into the game of Fracsland .

A conversationa l'agent is able to maintain a conversation in natural language with a user, either by text or by voice, and to extract information to be able to apply functionalities to the software where it's used, as well as to access information from third-party services, with which to answer the users.

On the other hand, the game of Fracsland is a project of educational character whose objective is to help consolidate the concept of fraction to the students of primary studies. Fracsland was developed in a previous project.

At the end of the project, these new agents are integrated into the Fracsland VR (Virtual Reality) project, developed in parallel to this project, in which voice interaction is a necessary condition due to lack of physical keyboard on the VR platforms.

Resum

Aquest projecte se situa, per una banda, en l'àmbit dels agents conversacionals, intel·ligències artificials preparades i enfocades a mantenir diàlegs amb un usuari, i per l'altra banda, en l'àmbit dels videojocs, concretament en integrar aquests agents conversacionals al joc de Fracsland.

Un agent conversacional és capaç de mantenir una conversa en llenguatge natural amb un usuari, sigui per text o per veu, i poder extreure'n informació per a poder aplicar funcionalitats al software on és utilitzat, així com accedir a informació de serveis de tercers, amb la que contestar als usuaris.

Per l'altra banda, el joc de Fracsland és un projecte de caràcter educatiu que té com a objectiu ajudar a consolidar el concepte de fracció als alumnes del Cicle Superior de Primària.

En el punt final del projecte, s'integren aquests nous agents al projecte Fracsland VR (Realitat Virtual), desenvolupat en paral·lel a aquest projecte, en el qual la interacció per veu és una condició necessària a causa de la falta de teclat físic a les plataformes VR.

Índex

Índex de figures

1 Introducció

Aquest projecte se situa en l'àmbit dels agents conversacionals, intel·ligències artificials preparades i enfocades a mantenir diàlegs amb un usuari, i per l'altra banda, en l'àmbit dels videojocs, concretament el joc Fracsland, un projecte previ[1], el qual ens servirà per a experimentar amb l'experiència d'aplicar els comentats agents conversacionals en un joc seriós i veure'n els avantatges i inconvenients. En les pròximes subseccions es tracten ambdues parts en detall.

Un agent o bot conversacional és un software que simula mantenir una conversació amb un usuari responent automàticament a entrades o peticions fetes per l'usuari. Habitualment les conversacions es mantenen a través de text, però recentment han guanyat valor les opcions amb una interfície multimèdia, normalment amb reconeixement de veu i la capacitat per respondre de la mateixa forma, utilitzant programes que converteixen el text a so i viceversa. Per tant, el diàleg es realitza en llenguatge natural, fet pel qual els agents ja estan preparats per a processar, gràcies als algoritmes de *Natural Language Processing* (NLP).

Per establir un diàleg, s'espera que s'utilitzin frases fàcilment comprensibles i coherents. Pel general, els bots no aconsegueixen comprendre del tot del que s'està parlant, sinó que tenen en compte un seguit de paraules o frases clau de l'interlocutor, que els hi permet retornar una sèrie de respostes preparades per la situació detectada.

Alguns agents conversacionals són integrats en sistemes de diàleg com 'assistents virtuals' i cada vegada més empreses estan utilitzant-los en lloc dels coneguts '*call centers*'. Els bots poden respondre a l'usuari i assistir-lo en múltiples activitats; entre les seves capacitats bàsiques estan les d'aprendre, buscar, recordar i connectar amb altres sistemes o serveis. Actualment, els bots conversacionals operen basats completament en intel·ligències artificials, amb un creixent interès a utilitzar la 'computació basada en humans' (HBC) per assolir un servei més eficient.

És cada cop més comú, per als agents més potents, el fet de millorar al llarg de les interaccions amb els usuaris i ha esdevingut una característica pràcticament obligatòria. S'utilitzen d'algoritmes d"aprenentatge automàtic', juntament amb una tècnica coneguda com a computació amb humans en bucle, o '*human-in-the-loop computing*', que arriben a ser la forma més efectiva per a entrenar als bots, a l'estar en un continuat bucle de retroalimentació. A més persones interactuant amb ells i aportant feedback de com es resolen els problemes, més millora l'algoritme d'aprenentatge automàtic, augmentant la capacitat per a transmetre una experiència més productiva i agradable per a l'interlocutor.

Degut a aquesta nova tecnologia, ha sorgit la necessitat de diferenciar els bots (bàsics) que segueixen unes regles simples, dels bots 'intel·ligents'. En els primers, el desenvolupador defineix el diàleg i el rang de respostes possibles, mentre que els segons permeten a l'usuari d'interaccionar lliurement amb el bot, processant la informació que li transmet, per a proposar una resposta. En l'àmbit dels agents amb intel·ligència artificial, la indústria tecnològica és la primera que ha començat a apostar per ells; sistemes com Watson d'IBM, LUIS de Microsoft o API.ai de Google (comunament conegut com a DialogFlow) són exemples d'aquest tipus d'agents, on la clau de la intel·ligència del bot resideix en l'entrenament que es fa d'ell. Concretament en aquest projecte utilitzarem DialogFlow per a incloure al joc de Fracsland un agent conversacional. La principal raó es degut a ser l'única plataforma que oferia un servei gratuït i que disposava de recursos i suport per a ser utilitzada des de el editor Unity, amb el que està fet el joc Fracsland. DialogFlow disposa de dos versions diferents (v1 i v2), la nova versió (v2) està encara

en fase 'beta' i no disposa de suport amb Unity, es per aquesta raó que aquest projecte utilitza la primera versió de DialogFlow. Existeixen altres raons que secunden l'ús de la primera versió de DialogFlow, en parlarem a la secció 4.4 Requisits tecnològics.

Fracsland és un projecte de caràcter educatiu que té com a objectiu ajudar a consolidar el concepte de fracció als alumnes del Cicle Superior de Primària i com aquest concepte es pot relacionar amb coses del món i de la vida quotidiana, ja que a priori sembla que és un concepte que en general costa d'assumir[1].

Des d'un entorn còmode com pot ser el d'un joc, s'introdueixen als alumnes diversos reptes i situacions a superar utilitzant les matemàtiques, concretament les fraccions, que són un element clau per al joc. Alhora, proporciona una eina al professor per a mantenir un seguiment sobre els resultats de l'alumne. El projecte també disposa d'un recurs, en forma de pàgina web per al professor, on es permet canviar els paràmetres dels problemes, per a poder oferir un marge de personalització als exercicis, en com es cregui que ha de ser el nivell de dificultat.

1.1 Conceptes relacionats

A continuació es presenten els conceptes relacionats amb els agents de DialogFlow, segons la documentació oficial[2]. Per estructurar la manera en la qual el bot interpreta i gestiona el diàleg, s'han definit una sèrie de conceptes com **intencions** (intent), **entitats** (entity), **contexts** (context). Aquests conceptes, i d'altres, necessaris per entendre el que s'explicarà en els pròxims apartats estan definits a continuació:

- **Agents**: Molts cops descrits com a mòduls de NLU (Natural Language Understanding), poden ser implementats en una aplicació, formar part d'una web o servei i transformar expressions naturals dels usuaris en accions i informació. Normalment dissenyats per a portar el flux de la conversa d'una manera específica, utilitzant contexts, intents, entities, etc...

- **Intents**: Un intent representa un 'mapeig' entre el que l'usuari diu i l'acció que ha de prendre l'agent. A DialogFlow, un intent, dit d'una altra forma, representa l'objectiu que l'usuari vol aconseguir, i que ha de detectar l'agent. Consta de les següents seccions:

  - **Frases de formació**: Es tracta del conjunt de frases exemple que utilitzarà l'agent per a detectar quan es tracta d'aquest *intent*. Per exemple, les frases d'un intent de salutació podrien ser: 'Hola', 'Hey' o 'Bon dia', d'entre moltes altres.
  - **Acció**: Es retorna juntament amb la resposta, és un camp pensat per a ser l'element que seleccioni quina acció s'ha de dur a terme en el software que està utilitzant a l'agent. Seguint l'exemple anterior, l'acció podria ser: '*Social.Salutacio*'. La nomenclatura està completament en les mans del desenvolupador. Aleshores el software que estigui utilitzant a l'agent, podria reproduir algun só o mostrar algun element de feedback de resposta a una salutació de l'usuari.
  - **Resposta**: El conjunt de respostes possibles, de les quals l'agent en tria una per a respondre a l'usuari. Si un mateix *intent* s'activa múltiples vegades, l'agent anirà triant les respostes que encara no han sigut triades, fins que ho

hagin estat totes. Seguint amb l'exemple, algunes possibles respostes a l'intent de salutació podrien ser: 'Hola! Com va?' o 'Hey, en què puc ajudar-te?'.

  - **Contextos**: (Explicats a continuació.)

- **Entities**: S'utilitzen per extreure paràmetres d'entrades en llenguatge natural, per exemple, a la frase: 'Com està el *temps* avui a *Barcelona*?', tant 'temps' com 'Barcelona' serien entitats. Qualsevol informació d'interès que es desitgi saber de l'usuari, tindrà la seva corresponent entitat. Les entitats poden tenir tipologies definides, és a dir, diferents tipus d'aquella mateixa entitat. Per a cada una de les tipologies, es poden definir sinònims per a detectar aquest mateix tipus dins de l'entitat.

  En les frases de formació de cada intent, es poden definir models que continguin entitats, d'aquesta forma es detecta quan l'usuari ho menciona, i quin de tots els valors de l'entitat ha pronunciat.

  Un exemple d'entitat podria ser una *destral*; l'entitat es la destral, però hi ha de diverses característiques, per exemple, de coure, de ferro, d'acer... Aquestes serien les tipologies de l'entitat destral. Cada una d'elles pot tenir diferents sinònims, per exemple, la destral d'acer també podria ser coneguda com a 'millor destral' o la destral 'més forta'.. Així s'aconsegueix que, diverses maneres de referir-se a un mateix objecte per a un usuari, acabin donant el mateix valor a l'agent, per tant, l'usuari rep la resposta esperada, indiferentment del sinònim que utilitzi.

- **Actions** i **Parameters**

  Una acció correspon al pas que ha de dur a terme l'aplicació que utilitza el bot quan un *intent* específic ha sigut detectat. Les accions poden tenir paràmetres per extreure informació de les peticions de l'usuari. Els paràmetres són elements generalment utilitzats per a connectar paraules, en les frases de l'usuari, a entitats.

En les frases de formació de cada intent, seleccionem les paraules que poden ser reconegudes com a entitats, aleshores de forma automàtica, es marquen com a valors de paràmetre, és a dir, els paràmetres guarden valors d'entitats. Per tant, a la secció de 'Action & parameters' es relacionen aquests paràmetres amb l'entitat desitjada com es pot apreciar a la Figura 1.
Els paràmetres poden ser del tipus variable o del tipus llista, poden ser definits utilitzant tipologies ja definides per DialogFlow (p.ex. Date, Localitzation, Any,...) i poden ser definits com a 'required', juntament amb una frase que demani explícitament aquest camp, en el cas que no es detecti el paràmetre (requerit) en la frase de l'usuari.

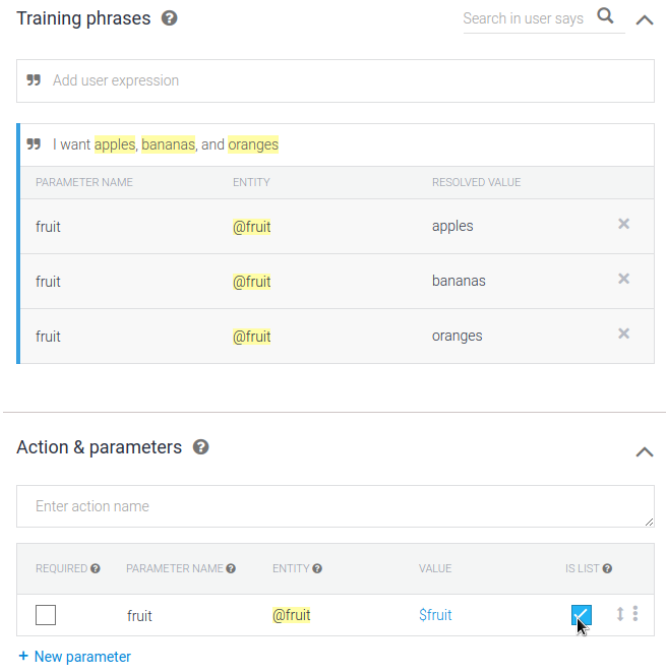

Figura 1: Relació entre entitats i paràmetres a DialogFlow.

Un exemple de paràmetre requerit podria ser el següent: L'usuari demana 'Com està el temps avui?' a un intent on es demana saber la ciutat (paràmetre requerit) per on es pregunta el temps. L'agent contestarà 'A quina ciutat?'.

- **Contexts**

  Són utilitzats per a transmetre informació d'altres moments de la conversació o de fonts externes. Els contexts utilitzen els paràmetres per a guardar informació, de fet, guarden únicament els paràmetres declarats. Cada intent disposa de l'opció de tenir contexts d'entrada i de sortida. En un *intent* que hi ha un context d'entrada, aquest passa a ser condició obligatòria per accionar el *intent* en qüestió.

  Els contexts tenen un temps de vida, conegut com a 'lifespan', el qual es defineix quan es crea el context, i pot ser modificat en qualsevol intent que agafi el context com entrada i el tingui de sortida, serà en aquest punt on se li assigna un nou lifespan. El temps de vida es comptabilitza a través d'intents accionats.

  A la Figura 2 es pot veure el cas de l'intent 'Social.Bye (w purchase)' el qual té el propòsit d'acomiadar al jugador, però en el cas d'haver realitzat una compra. Val a dir que el context 'objectSelected' es crea en un altre intent orientat a la compra, i guarda l'objecte comprat per a poder donar una resposta personalitzada a l'acomiadament, com es pot apreciar a la Figura 3[1], sempre i quan estiguin els paràmetres declarats. Per a que DialogFlow pugui extreure el valor dels paràmetres guardats al context es necessari afegir '$' seguit del nom del paràmetre

[1]Per a poder capturar i mostrar els paràmetres del context a la resposta és necessari declarar els paràmetres dins de l'intent en qüestió, de la mateixa forma que es fa a la Figura 1.

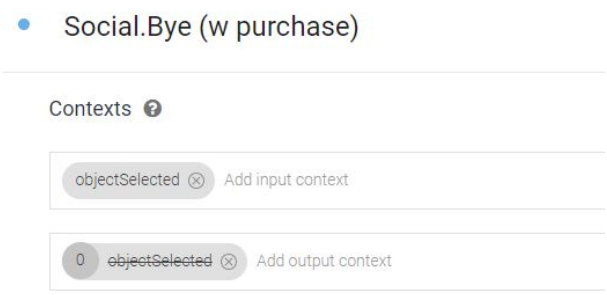

Figura 2: Relació entre entitats i paràmetres a DialogFlow.

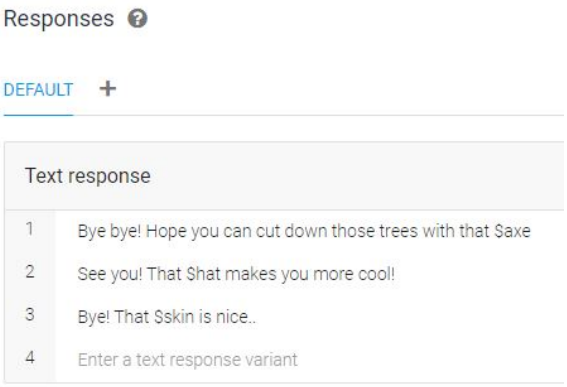

Figura 3: Respostes amb paràmetres capturats del context de entrada.

- **Dialogs**

  A DialogFlow existeixen dos tipus de diàlegs quan definim el procedir d'un agent:

  – Diàlegs **linears**, l'objectiu del qual és recopilar informació, normalment de manera seqüencial, per a completar alguna acció requerida. Utilitzen la tècnica coneguda com a '*Slot Filling*' (ompliment de ranures) per a obtindre tots els paràmetres que necessita per a procedir.

  Per il·lustrar-ho utilitzarem l'exemple d'un agent per a fer reserves a hotels; per tant, es necessita certa informació mínima: 'a quin hotel?', 'data d'entrada?' i 'data de sortida?', es designen doncs, 3 paràmetres que considerarem necessaris o requerits, i l'agent preguntarà de forma seqüencial per a cada un d'ells. Com que l'agent només te l'ús de reservar un hotel, es necessita només un intent. Quan tots els paràmetres requerits hagin estat respostos, l'agent farà la reserva i haurà acabat el diàleg.

  – Diàlegs **no-linears**, els quals poden tenir diverses branques per on el diàleg pot fluir, depenent de les respostes que l'usuari decideixi donar.

  Un exemple de diàleg no linear seria, per exemple, el cas d'un agent amb la finalitat de preguntar opinions sobre l'estància a un hotel. Les preguntes es fan en ordre, però l'usuari pot contestar de nou alguna pregunta ja passada per a canviar-ne la resposta sense que l'agent es perdi, ja que la característica bàsica dels diàlegs no lineals és que pot canviar el flux del diàleg, fent que la conversació per a l'usuari sigui més natural i satisfactòria, però afegeix una complexitat considerable respecte als linears.

Per il·lustrar les diferencies en el flux de la conversa, entre els agents amb diàleg lineal i els de diàleg no lineal, es pot veure a la Figura 4 com procedeixen en el mateix tipus de conversa. En el cas de l'agent no lineal, no entén el que l'usuari li diu pel fet que es troba en el pas de detectar el 'When' com a paràmetre requerit, i no entendrà res que no pugui relacionar amb el paràmetre darrere 'when'.

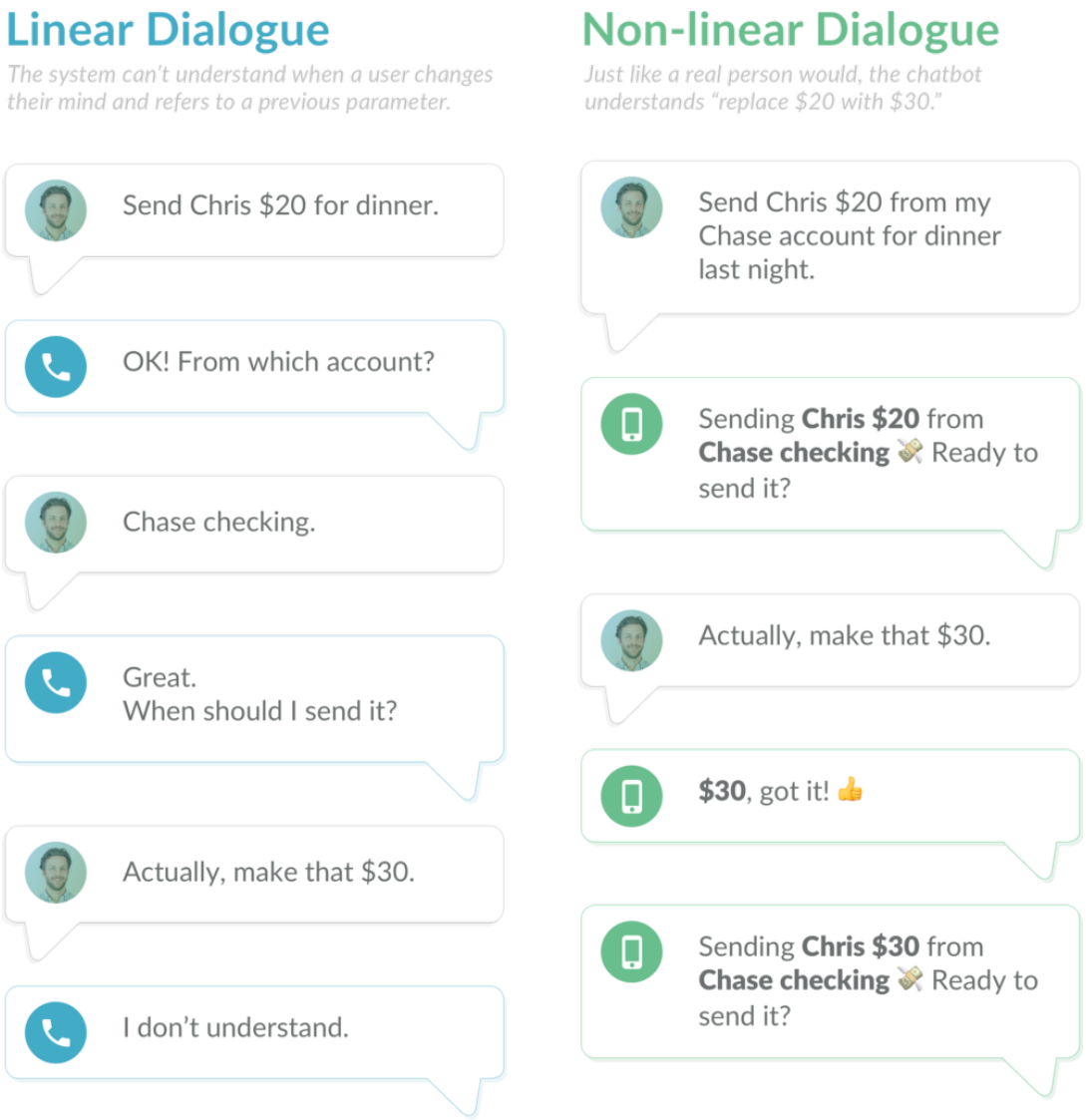

Figura 4: Comparativa entre els diàlegs linears i els no linears.

Els agents creats en aquest projecte pretenen ser, i són, agents amb diàlegs no linears; no hi ha ordre per preguntar res, i qualsevol branca de diàleg es pot tallar per parlar d'un altre tema sense problemes.

2 Objectius

Els objectius d'aquest projecte radiquen al voltant dels agents conversacionals, la seva creació, comprensió i el seu ús. Conèixer els diferents serveis disponibles per a la creació d'aquests agents i generar-ne un per, a posterior, integrar-lo en un joc VR (Virtual Reality o Realitat Virtual). Aquest projecte es desenvolupa en paral·lel amb un altre projecte que té la fi de portar el joc Fracsland[1] a la realitat virtual.

Així doncs, l'última part d'aquest projecte consisteix a integrar l'agent conversacional a FracslandVR com a millor solució per a la comunicació amb NPCs, importants o destacats per a la jugabilitat, ja que en les tecnologies VR no sempre és bona practica, o una solució eficient, el fet de tenir teclats físics/virtuals a l'abast. Així doncs, els objectius d'aquest treball queden dividits en dos grans subobjectius que es detallen a les següents subseccions:

2.1 Aprendre i crear un agents conversacionals

Aquest objectiu comprèn tot el referent al necessari per a tenir a un agent de DialogFlow implementat i funcionant en un projecte de Unity. Des d'aprendre de zero la creació i les possibilitats dels agents DialogFlow, fins a trobar, actualment, quines d'aquestes funcionalitats són accessibles des d'un joc programat amb Unity, i implementar-les.

- **Investigar i aprendre els serveis que ofereix DialogFlow**

  Un cop vist que DialogFlow oferia el servei més adient per aquest projecte, el següent pas és documentar-se i aprendre com funciona la plataforma i com utilitzar els recursos que ofereix per a treure el màxim potencial possible entre els agents de Google i l'entorn Unity.

- **Saber crear agents conversacionals de DialogFlow**

  Utilitzar els coneixements adquirits per a crear un agent conversacional a DialogFlow. L'agent està dirigit, en l'última etapa del projecte, a ser integrat en un personatge en concret de Fracsland, el venedor, així doncs, es dissenyarà per a comunicar-se amb el jugador i parlarà fent servir conversacions orientades a aquest rol. Addicionalment també es crearà un agent amb funcions d'assistència per al jugador amb tot el necessari per a ser funcional.

- **Utilitzar la plataforma DialogFlow a l'editor Unity.**

  Investigar i trobar recursos oficials, o no oficials, per a utilitzar els serveis de DialogFlow des de Unity. Implementar en una escena d'un projecte de Unity una demostració bàsica de comunicació amb l'agent "venedor de Fracsland".

- **Trobar solucions per al reconeixement de veu en Unity.**

  Una de les capacitats més interessants dels agents és el fet d'entendre el llenguatge natural i respondre de la mateixa forma, per tant, té molt valor si es pot parlar amb ell i que et respongui de veu. La interacció amb DialogFlow v1 s'ha de dur a terme a través de text i respon amb una estructura JSON, que inclou un paràmetre amb la resposta, per tant, les solucions necessàries han de ser des de Unity. En primera instància, i per prioritat, buscar i implementar una solució de reconeixement de veu en Unity, que ens retorni un text amb la frase pronunciada per l'usuari. Inclosa

en aquesta tasca, està la de trobar com seleccionar l'idioma del reconeixement. La solució final ha sigut la de trobar les funcions necessàries per a comunicar amb l'eina pròpia de Windows, ocasionant que aquesta funcionalitat només funcioni en aquest Sistema Operatiu.

- **Trobar solucions per a què la resposta de l'agent, al Unity, sigui de veu**

  L'agent amb tecnologia de Google en l'entorn Unity3D ja reconeix la veu de l'usuari, obté i mostra satisfactòriament una resposta de DialogFlow per pantalla.

  Només resta, per assolir aquest objectiu, que la resposta sigui pronunciada de veu. Un cop investigades les possibles opcions, es troba que la manera més accessible, contant que ja existeix la limitació de només funcionar en Windows el reconeixement de veu, és un cop més, utilitzar els recursos de Windows per a utilitzar les seves funcions de *Text-to-Speech*.

2.2 Integrar agents conversacionals a FracslandVR

En aquest punt, ja es coneixen i comprenen les funcionalitats de DialogFlow juntament amb l'entorn Unity3D, aquest objectiu doncs, es desglossa en tot el referent a la integració d'aquest tipus d'agents en personatges destinats a ser integrats a l'altra branca del projecte, FracslandVR.

- **Integrar un agent de DialogFlow al Venedor (NPC) de Fracsland.**

  Extreure, del projecte original de Fracsland, al venedor i situar-lo en un projecte nou de Unity. Implementar-hi el agent de DialogFlow "venedor de Fracsland"al propi personatge venedor i programar un sistema de obtenció de dades de la resposta JSON que ens retorna a cada petició DialogFlow, per a accionar funcions del venedor o provocar diferents efectes que aportin coherència a la interacció amb l'usuari.

- **Crear un nou personatge assistent i integrar-li un agent de DialogFlow.**

  Comprèn crear de zero, utilitzant 'assets' ja inclosos en els recursos de Fracsland, un personatge destinat a ser un assistent personal del jugador, el qual ha de tenir competències per a aconsellar en funció de la situació actual o donar 'tips' sobre el joc en general. De la mateixa manera, comprèn la creació, a la plataforma DialogFlow, del agent pertinent a aquest rol.

- **Integrar en FracslandVR els dos agents creats.**

  Un cop creats els dos agents i integrats en models de Unity preparats per a funcionar amb el codi de Fracsland, el següent pas es integrar-los dins les escenes de FracslandVR i trobar solucions d'interacció amb els recursos que plataforma OculusVR ofereix en el seu SDK de Unity.

  El resultat, per tant, és el de tenir la possibilitat de mantenir conversacions en llenguatge natural (gràcies a les capacitats de *Natural Language Processing* o NLP del agent), i de pròpia veu, amb el personatge que ofereix objectes al jugador, què a part, ara disposa d'una base d'acudits i capacitat per mantenir diàlegs més personals, gràcies al modul 'SmallTalk' que DialogFlow ofereix als seus agents; i per altra banda, poder comunicar-se amb un assistent que estarà visiblement a disposició del jugador, per exemple, seguint-lo allà on vagi. L'assistent ha de poder

guiar al jugador si així li demana, o donar-li 'tips' i consells generals sobre el món de Fracsland i/o orientats a la situació/escenari actual. Al joc de Fracsland hi ha un seguit d'escenaris, en els que es desenvolupen un seguit de missions. Les missions són especifiques de cada zona, per tant podem tractar de la mateixa forma, per exemple, la zona 'Granja' amb la missió de 'plantar llavors'. Per tant des de DialogFlow l'assistent donarà consells sobre el mapa-missió actuals.

3 Treballs relacionats

En aquest apartat s'explica com es troba actualment el sector dels agents conversacionals, es a dir, tota aquella IA orientada a mantenir diàlegs amb usuaris i respondre'n amb accions quan es sol·liciti. Es divideix en dos subapartats, un dedicat als agents conversacionals com a tal i a exposar alguns exemples dels més coneguts, un cop vist el concepte, el segon apartat es dedicarà a experiències conversacionals i d'assistents virtuals aplicats/integrats en videojocs.

3.1 Agents conversacionals

En aquest apartat s'exposen alguns dels serveis més coneguts basats en agents conversacionals i a partir de quins recursos han estat creats. Tots els agents comparteixen el fet que la comunicació es realitza per veu, tant les peticions de l'usuari com les respostes del bot, a part de tenir l'opció de comunicar-te per text de manera convencional.

En més d'una ocasió veurem a aquests agents nomenats com 'assistents virtuals', que a priori podríem considerar sinònim d'agent conversacional o 'chatbot', però per a alguns experts en el tema són conceptes propers però amb diferències.

Per a **Rob High**, vicepresident d'IBM, les diferències radicarien en: [3]

- **Chatbot**: És el més senzill de tots, identifica la intencionalitat de la pregunta o la petició i la respon, sense anar més enllà. Per clarificar-ho, l'usuari expressa un enunciat, el bot el reconeix buscant la intenció i finalment busca la rasca mapejada a aquesta intenció. És el que s'anomena 'intercanvis d'un sol torn'. Pel general es tracten d'agents amb diàleg lineal, explicats a la secció 1.1.

- **Agent Conversacional**: Utilitzant les seves pròpies paraules: "*Un agent conversacional és aquell que compromet a l'usuari final a comprendre realment la naturalesa del problema més enllà de la pregunta*". Això significa que el bot ha de determinar quan és necessari aprofundir més i/o demanar més informació a l'usuari que els hi demana o pregunta certa cosa, amb la finalitat d'avançar-se al problema real i donar una resposta amb sentit dins el context de la pregunta, no només quedar-se en la superficialitat d'aquesta.

- **Assistent Virtual**: Citant-lo de nou: "*Un agent conversacional està més centrat en el que necessita per mantenir una conversació. Amb els assistents virtuals o personals, aquests conceptes tendeixen a ser més rellevants en els casos que intentes crear la sensació que l'agent té la seva pròpia personalitat i d'alguna forma té un vincle associat de forma única a l'usuari.*"Així doncs, segons ell, la diferencia d'aquest tercer grup és el fet que transmeten a l'usuari la sensació de tenir un majordom virtual a la seva disposició en qualsevol moment. Aquest concepte i el d'agent conversacional intersequen en el punt en què l'agent conversacional té una pretensió d'establir un tracte personalitzat amb l'usuari.

Aquesta és la forma en què Rob High divideix els diferents conceptes, però per les pròpies descripcions es pot entendre que hi ha zones en què dos conceptes col·lisionen i comparteixen particularitats de més d'un grup dels mencionats anteriorment.

3.1.1 Cortana (Microsoft)

Cortana és un assistent virtual segons la classificació anterior. El seu desenvolupament va començar l'any 2009 i no va ser fins al 2013, a la conferència de Microsoft anomenada BUILD, que va ser demostrada.

Els serveis que ofereix Cortana són amplis, si ens adrecem a la pàgina de Microsoft destinada a explicar que és i que fa la seva assistenta virtual, entre d'altres, es destaquen les següents funcions:

- Facilitar recordatoris de diferents estils i formats, pot utilitzar fotos o informació extreta de contactes per a complementar el recordatori per a cert esdeveniment. Realitzar seguiment de paquets en trajecte a casa, enviar correus o missatges de text, i fer una gestió en general de la informació del calendari. (Figura 5)

- Mantenir una conversa en llenguatge natural i obrir qualsevol aplicació disponible al dispositiu que l'usuari sol·liciti. Els diàlegs són no-linears.

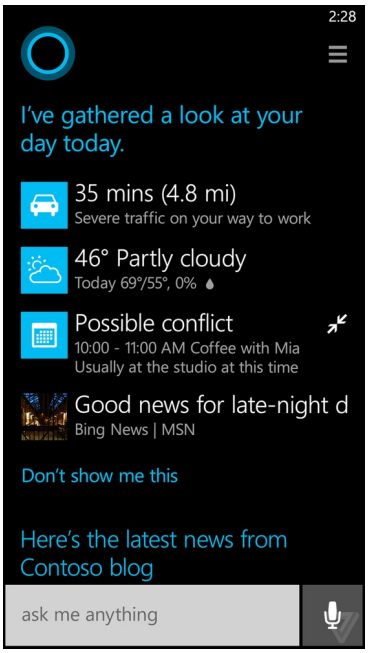

Figura 5: Cortana recopilant informació i informant al usuari de manera autònoma

Com es pot observar, els assistents virtuals poden accedir a informació d'altres aplicacions i serveis per obtenir informació, o inclús afegir-ne, com és el cas d'algun esdeveniment al calendari o d'enviar correus[4].

Aquestes funcions i capacitats, pel general, es mantenen en comú entre la majoria d'assistents virtuals. En els següents apartats no entrarem tan en detall, sinó que es definirà de forma senzilla i només es tractaran les peculiaritats de cada agent.

3.1.2 Siri (Apple)

Siri és una aplicació amb funcions d'assistent virtual i personal, a vegades amb la seva pròpia personalitat, per a sistemes de Apple. Com Cortana, disposa d'un seguit de funcionalitats troncals bàsiques i comuns als bots d'aquest caire, i en els últims anys han invertit esforços a augmentar el conjunt de serveis web amb els quals hi té compatibilitat.

Siri no va ser desenvolupada per Apple, sinó comprada l'any 2010 de mans de '*SRI venture group*', un grup de programadors orientats a la intel·ligència virtual[2] que van crear-la a finals de l'any 2007. Val a mencionar que, entre d'altres, una de les fonts de finançament pel projecte Siri venia de DARPA (Defense Advanced Research Projects Agency)[5].

De les peculiaritats que podríem parlar de Siri, destaquen aquelles en què Apple hi ha posat especial èmfasi; ja que l'objectiu és el fet de transmetre una experiència de comunicació real, hi ha petits detalls que pretenen fer de Siri, un ésser amb pròpia personalitat. Entre d'altres, ens pot cantar un rap, donar-nos una opinió políticament correcte d'Android, el seu competidor, o inclús explicar-nos històries variades.

[2]No confondre amb intel·ligència artificial, la intel·ligència virtual és aquella orientada a mantenir diàlegs naturals amb un usuari.

3.1.3 Alexa (Amazon)

Alexa és l'assistent virtual desenvolupat per Amazon, amb la peculiaritat de què va ser creat per a ser utilitzat en els seus 'altaveus intel·ligents'. Comparteix amb els seus homòlegs pràcticament totes les funcions esmentades, però a l'estar pensada inicialment per a l'ús integrat en els seus altaveus, Alexa pot reproduir música i crear llistes de reproducció fàcilment.

Actualment Alexa només està la comunicació disponible en anglès, alemany i japonès; però es preveu un llançament molt pròxim de compatibilitats amb altres llenguatges, com per exemple, l'espanyol.

A finals de l'any 2017, Alexa comptava amb més de 5000 empleats treballant en ella i en productes/serveis relacionats. Està doncs, en ple creixement. Una de les novetats ha sigut la compatibilitat entre Alexa i un joc recent al mercat, Destiny 2, i en parlarem més en detall a la secció 3.2.1 Alexa i Destiny 2[6].

3.1.4 Google Assistant

Google Assistant és l'assistent virtual desenvolupat per Google disponible principalment per a dispositius mòbils i domèstics intel·ligents, amb capacitat per a mantenir diàlegs bidireccionals.

La seva primera aparició va ser al maig de 2016 per a uns pocs dispositius de forma exclusiva, fins al febrer de 2017 que va fer el salt a la majoria de dispositius Android. Està inclús disponible en iOS en format d'aplicació independent.

Els usuaris interactuen principalment a través de la veu, tot i que també admet entrada a text. Entre les seves capacitats estan les de buscar per internet, programar esdeveniments i alarmes al calendari, ajustar la configuració de hardware al dispositiu de l'usuari i mostrar informació general associada al compte de Google de l'usuari. Els desenvolupadors han anunciat que l'assistent podrà identificar objectes i recopilar informació visual a través de la càmera del dispositiu, admetrà la compra de productes i l'enviament de diners, així com la identificació de cançons[7].

Com es pot comprovar, els assistents virtuals, cada cop més, estan tendint a ser una aplicació 'all-in-one' en el que, idealment, l'usuari pot accedir a tot el que el dispositiu li pugui oferir. La compatibilitat i la inclusió de funcionalitats relacionades amb la Visió Artificial i el reconeixement de música, pel moment, és una exclusivitat que només han promès des de Google.

3.2 Agents conversacionals en videojocs

Hi han diferents maneres de poder utilitzar als agents conversacionals en aquest sector, ja sigui fent el bot extern o completament integrat en les funcionalitats i mecàniques del joc. En els següents subapartats es parla de diferents exemples; en primer lloc, un agent del qual ja s'ha parlat, Alexa, que ha estat afegida com a extra per a un joc que ja existia, convertint-se en un assistent extern amb efectes dins el joc; després es parla de Sam, l'assistent per a la botiga virtual de Ubisoft, per tant, completament extern als jocs, però amb molt coneixement d'ells; i finalment es tracten dos jocs, els quals sense els assistents virtuals, no existirien, ja que tant la mecànica com la història giren entorn de mantenir conversacions amb els agents.

3.2.1 Alexa y Destiny 2

Destiny 2 és un joc que va sortir a finals de l'any 2017, i en cap moment ha disposat de suport de cap tipus per part d'un assistent virtual, ni tan sols d'un 'chatbot', però recentment, i per a promocionar els seus nous productes, Amazon va decidir innovar en aquest camp i va triar a Destiny 2 com al seu punt de partida.

Per als usuaris que disposin d'un altaveu intel·ligent Amazon Echo (altaveu amb la particularitat de disposar de comunicació directa amb Alexa, la seva assistent virtual) podran utilitzar el dispositiu per accedir a diferents funcions dins el joc.

El funcionament és el següent, l'usuari activa l'atenció de l'assistent pronunciant 'Alexa, tell Ghost to...', sent Ghost el nexe entre Alexa i el joc, dins del mateix joc. Les ordres es donen per veu, per tant, no es bloquegen les accions al joc, l'usuari pot seguir disparant o interactuant amb el món del joc.

Figura 6: Imatge de la campanya publicitaria de Alexa per a Destiny 2.

Les ordres que Alexa pot enviar a Ghost són varies, des d'equipar o des-equipar objectes al jugador (ja sigui mencionant-los pel seu nom o fins i tot dient-li que ens equipi amb la millor arma per a la situació actual), organitzar reunions i partides amb coneguts, fins a qüestions menys directes com podrien ser el fet de preguntar per recomanacions sobre quines activitats fer en aquell moment o demanar informació sobre les faccions del joc, inclús dades i fets curiosos sobre l'univers on es troba el personatge. [8]

En essència, es pretén transmetre una experiència de conversació completa, ja que Alexa està preparada, sigui fora o dins el joc, per a gestionar preguntes i respostes personals, com podrien ser 'com estàs?' o 'què fas quan t'avorreixes?'.

3.2.2 Ubisoft i el seu assistent Sam

Actualment, i pel que respecta a aquest sector, en el que cap empresa important vol arriscar-se a quedar-se enrere, Ubisoft no podia ser menys, i han llençat una idea un pèl diferent dels assistents com Alexa, el suficient per a haver decidit que s'explicaria en la secció dels agents aplicats a jocs.

La idea de Ubisoft no ha sigut la de desenvolupar un assistent virtual per a utilitzar dins els videojocs, sinó la de situar-lo a la seva plataforma de compra de jocs, *Ubisoft*

Club. Així doncs, la funció del seu assistent és la de recomanar jocs segons els gustos i les preferències de l'usuari, aprendre amb el seu feedback i donar qualsevol explicació que tingui a l'abast del joc pel qual l'usuari s'interessi.

Entre les seves capacitats, destaca la anomenada '*Daily Login*', que analitzarà els hàbits del jugador i les seves estadístiques per mostrar consells personalitzats aplicades a jocs en concret. Les bases de dades necesaries per a facilitar aquesta funcionalitat són difícils de poblar i sobretot de preparar-ne les recomanacions per a cada joc en concret, de moment només està disponible per al joc *Rainbow Six Siege*, però amb previsió d'abastir una quantitat molt més amplia de títols).

Com es pot veure a les imatges de la Figura 7, com qualsevol agent conversacional complet, està preparat per a transmetre l'experiència completa de comunicació i té respostes ocurrents per a preguntes personals o inclús per a preguntes fora de les seves capacitats (món dels videojocs).

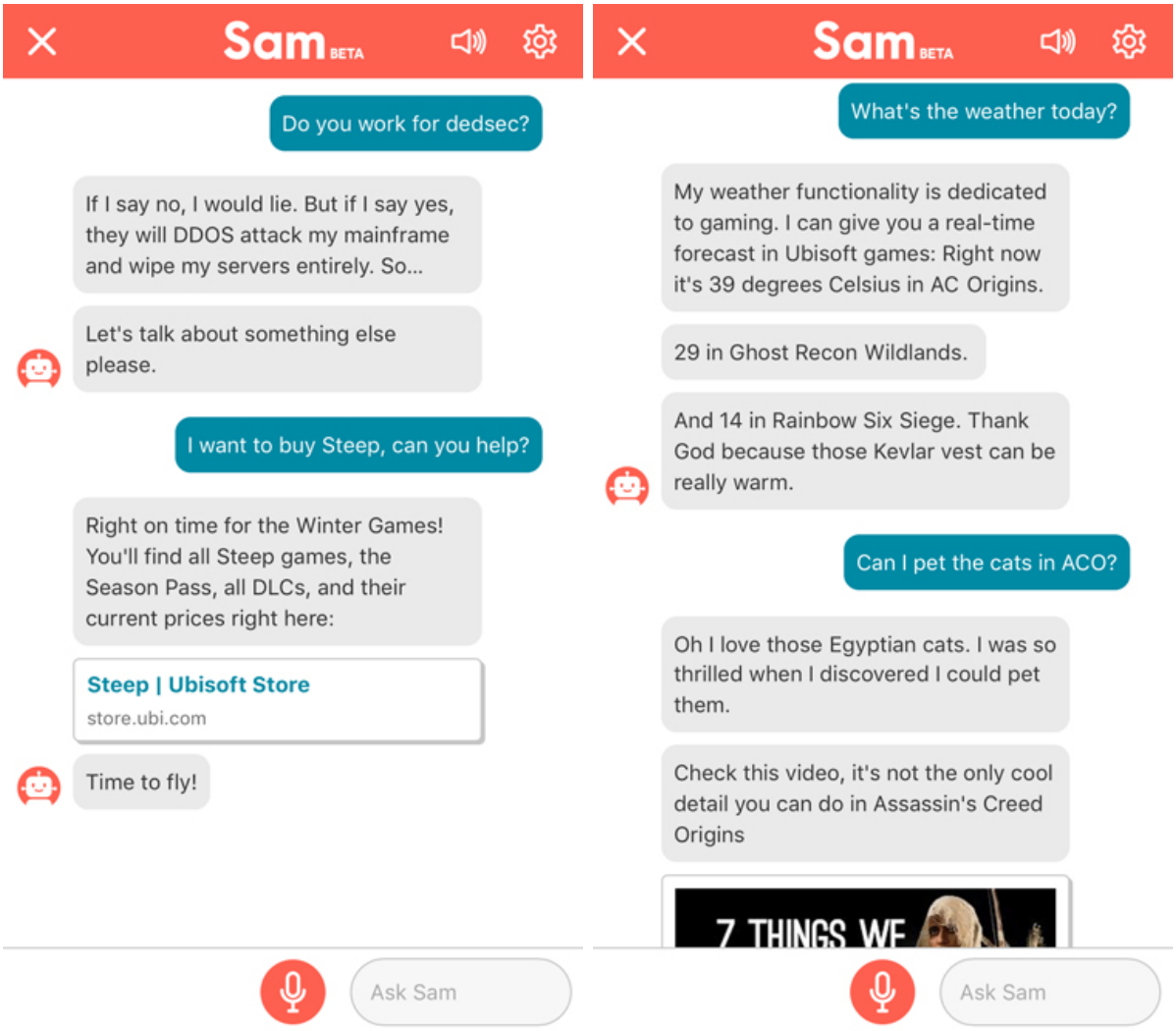

Figura 7: Preguntes personals (a la esquerra) i preguntes fora de context (a la dreta).

Per a concloure, Sam és definit de manera breu com un '**assistent personal de *gaming***' i actualment es troba en fase de proves i només disponible des de *iOS* i *Android* a Canadà, d'aquí que encara no tingui tot el potencial que podria tenir. Està planejat estendre la fase de proves pròximament cap a altres mercats i països. [9]

3.2.3 KOMRAD i Event[0]

Komrad i Event[0] són dos videojocs en què tota la jugabilitat gira entorn d'un agent conversacional. Malgrat que els *chatbots*, i els agents conversacionals en general, no hagin aconseguit encara perpetuar-se a la vida dels usuaris amb naturalitat, resulten, normalment, una experiència de comunicació divertida. Encara ha de passar un temps perquè els usuaris comencin a utilitzar-los per a la seva vida privada, sigui gestionant les seves agendes o inclús les contes bancàries, però al punt on es troben avui en dia tenen un gran potencial per al món dels videojocs. Així ho volen demostrar els creadors de Komrad i de Event[0], dos jocs que van sortir al mercat en tan sols un dia de diferència, els quals els agents conversacionals són el cor de l'experiència[10].

En el cas de **Komrad**, la història del joc situa a l'usuari davant d'un ordinador de l'època de la guerra freda, que ha estat fora de servei més de 30 anys, on se li demana que faci de 'hacker' i n'obtingui informació. És aleshores que el jugador es troba que l'ordinador està basat en una IA Soviètica experimental.

Una decisió de disseny important de Komrad és el fet que les opcions es seleccionen, no dóna plena llibertat a l'usuari per escriure el que desitgi, i això, segons els seus creadors, és per a evitar caure en la situació de què l'usuari no troba el que el bot està esperant per rebre.
Komrad ha estat desenvolupat per Sentient Play, una companyia relativament nova fundada per Brad Becker, director de disseny de Watson, l'assistent virtual d'IBM i que no ha estat mencionat en aquest projecte, ja que està, actualment, a la par amb Cortana o Siri. Per tant es pot deduir la tecnologia amb la qual funciona el joc.
De fet, Becker comenta que el joc va néixer pel seu desig personal d'aconseguir, de manera ràpida, prototipatges d'intel·ligències artificials per a dialogar i aconseguir un públic que hi interactuïn per a fer '*testing*' real. Així doncs, Komrad aporta informació útil sobre la interacció amb usuaris per a millorar el servei de Watson d'una forma pràctica i entretinguda. Tot i així, Komrad no està plenament basat en Watson, ja que com indica el mateix Becker, utilitza un 'mix' de codi i diàlegs '*pre-script*', o predefinits, per a simular un diàleg real sense perdre el flux i el control del joc. Les respostes i reaccions no estan únicament basades en el que s'està dient a l'agent, sinó que també es tenen en compte les respostes passades, i el global de la relació que es va construint amb l'agent per part del jugador.

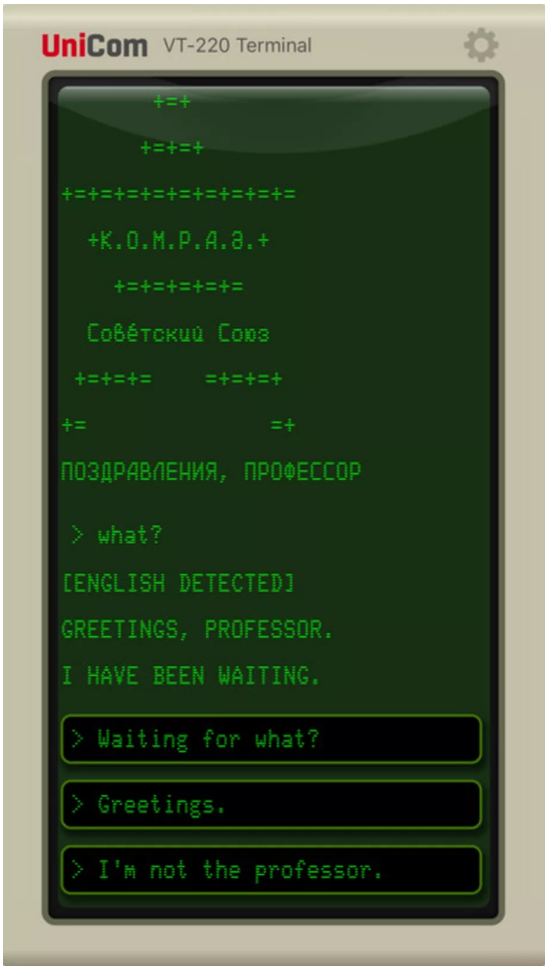

Figura 8: Primer diàleg possible amb l'ordinador de Komrad.

Videojocs moderns, de l'estil de Komrad, no són els primers que brinden al jugador la possibilitat de dialogar. La idea ja va aparèixer, farà uns 40 anys, amb jocs que basaven les seves aventures en text i en la comunicació amb algun tipus d'agent o bot, com per exemple, '*Zork and The Hitchhiker's Guide to the Galaxy*'. La diferència amb els projectes d'avui en dia és la tecnologia darrere aquests bots o agents, la qual permet als desenvolupadors de dissenyar interaccions més creïbles i naturals amb personatges virtuals.

Event[0] parteix del mateix concepte, i el porta un pas més enllà. L'escenari del joc es desenvolupa en una nau espacial, accidentada, on el jugador és enviat per a investigar què ha succeït. L'únic ser o entitat que es troba el jugador és una IA anomenada Kaizen. L'entorn està fet en 3D i és completament explorable, Kaizen és accessible des de diverses computadores, o terminals, repartides per l'escenari, com es pot apreciar a la Figura 9.

Figura 9: Entorn 3D per on el jugador es mou lliurement. La IA es accessible des d'un seguit de terminals, de diferents formes, repartits pel joc.

El joc va ser inicialment creat com a un projecte estudiantil, però després de rebre nombrosos premis l'any 2015, es va decidir llençar-lo com a producte completament comercial, l'estudi que hi posa la firma és anomenat The Ocelot Society. Degut al salt com a projecte comercial, es van renovar pràcticament tots els escenaris i es van afegir noves funcionalitats com la d'un 'scanner' per analitzar objectes i que Kaizen en pogués donar informació[10].

Mentre que Komrad presenta un agent amb una 'personalitat' deteriorada degut al pas dels anys, Kaizen pretén adaptar-se a com el jugador interactua amb ella. A diferencia de Komrad, dota al jugador de plena capacitat per escriure a la IA en comptes d'oferir un seguit d'opcions tancades, sempre comptant amb els problemes i els avantatges de cada una de les opcions.

Kaizen aprèn amb l'usuari i modifica el seu comportament acord a com l'usuari es comporta amb ella, portant molts cops, a situacions o frases curioses, com es mostra en la Figura 10.

Figura 10: Kaizen pregunta a l'usuari si creu que l'homicidi pot estar justificat depenent del context, després que aquest no hagi accedit a fer el que li demanava.

Sergey Mohov, un dels creadors, comenta que inclús als escriptors del bot els hi sorprenia amb freqüència, mentre hi feien proves, per la seva "espontaneïtat"[10].

4 Anàlisi

En aquesta secció es parla de l'anàlisi fet sobre el projecte Fracsland per a determinar com, l'ús d'agents conversacionals, podria facilitar la interacció amb el joc i simplificar-la. Sabent que el destí d'aquests canvis és el d'acabar en la versió de **FracslandVR**, és indispensable que la comunicació es dugui a terme per veu, ja que des d'una plataforma VR és complicat i poc intuïtiu l'ús de teclats, físics o virtuals.

4.1 Introduir un agent conversacional a Fracsland

Al llarg de la partida de Fracsland es poden destacar bàsicament dos personatges NPC (*Non-player character*), un és el venedor o també anomenat Vendor, de l'anglès, i l'altre és Lord Barus.

La funció del Vendor és la d'oferir diversos objectes al jugador a canvi d'or o diamants, les monedes al món de Fracsland, dit d'una altra manera, vendre. Aquests objectes són usualment per a resoldre situacions o completar missions, en altes casos són només visuals per a oferir una sensació de personalització al jugador (a través de barrets i/o roba). Per altra banda, Lord Barus, el governador del món on es troba l'acció del joc, s'encarrega de mostrar al jugador el panell amb les missions disponibles perquè seleccioni les que vulgui.

Es presenten, doncs, dos possibles personatges a Fracsland que podrien ser agents conversacionals. Sabent les possibilitats d'un agent conversacional, el fet que pot interactuar en llenguatge natural, amb conversacions més o menys complicades, i que disposa de facilitats per evitar repetir les mateixes respostes (mentre quedin opcions sense repetir), en aquest projecte es va decidir que el personatge amb més marge per aprofitar aquest potencial seria el Vendor, ja que Lord Barus es limita a mostrar missions disponibles únicament i el jugador no hi necessita tantes interaccions conversacionals per a comunicar-se amb ell.

Així doncs, el venedor o **Vendor**, tindrà capacitats per a rebre input de veu de l'usuari i contestar-li de la mateixa forma. Entre les seves capacitats bàsiques estan:

- Permetre la **compra d'objectes** durant el diàleg, que comprèn:
  - Seleccionar la categoria d'objecte per a veure la llista completa.
  - Seleccionar algun objecte concret perquè el mostri individualment
  - Seleccionar el tipus de moneda amb el que es vol pagar.

  Tots els paràmetres per a realitzar una compra completa es poden assolir en una sola frase o fins en tres de diferents.

- **Explicar acudits** o fer comentaris ocurrents.

  Els agents de DialogFlow permeten de ser creats amb un mòdul predefinit de sortida, entre molts, un d'ells és el mòdul de 'Jokes' o bromes, i va ser seleccionat com a un extra per al venedor per a donar-li una personalitat més divertida.

- Capacitat per a **mantenir una conversa personal** bàsica.

  Un cop més, els agents de DialogFlow ofereixen una opció, que pot ser activada si el desenvolupador ho decideix, anomenada 'SmallTalk', un paquet d'estructures frase-respostes que comprèn des de preguntes bàsiques com 'Qui ets?' o 'Com estàs?'

fins a qüestions més complexes com 'Qui és el teu cap?' o 'On has nascut?'. Les variades categories del paquet SmallTalk són les següents: *About Agent*, *Courtesy*, *Emotions*, *Hello/Goodby*, *About user*, *Confirmation* i *Other*.

- Disposar d'**interaccions proactives**.

  Un interessant potencial d'ajuntar agents conversacionals amb videojocs (inclús en altres softwares) és el fet que existeixin certes accions que activin certes frases o respostes per part de l'agent, en més d'una situació, a ulls de l'usuari, és l'agent qui inicia la conversació.

  En aquest cas s'ha experimentat amb aquesta tècnica en la salutació del personatge al jugador, un cop aquest últim s'apropa a certa distància. Per consegüent, també el comiat és proactiu.

Més endavant, després de l'agent Vendor, es va decidir aprofitar els coneixements, i el temps encara disponible per al projecte, per a la creació d'un segon agent conversacional, aquest però, completament nou a l'univers de Fracsland i amb un rol d'assistent virtual per al jugador. A aquest agent li direm 'Assistant'. Les seves funcions són les d'oferir al jugador un recurs amb el qual comunicar-se en qualsevol moment, i que pugui respondre per a orientar-lo segons en el moment de la partida que es trobi.

Les funcions més bàsiques del **Assistant** desglossades són, molt ràpidament, les següents:

- Respondre a preguntes situacionals del jugador

  - Tals com: 'On estem?' o 'On hem d'anar?'

- Respondre a preguntes sobre la missió activa

  - Si el jugador es troba en una missió, l'agent ha de respondre amb pistes o explicacions si es demana ajuda per part del jugador. Tals com: 'Que haig de fer?' o 'On he d'anar?'

- Capacitat per a mantenir una conversa personal bàsica. Amb el ja mencionat recurs de DialogFlow, SmallTalk.

Com es pot veure, la seva tasca serà la d'acompanyar al jugador i resoldre dubtes que el jugador es pugui trobar. És una de les aplicacions més conegudes i esteses dels agents conversacionals, tema el qual ara no hi entrarem, ja que s'aprofundeix molt més a la secció 3.1 Treballs Relacionats. Agents Conversacionals.

Per concloure doncs, tenim la situació en la qual hi ha un NPC venedor, el qual ha estat seleccionat, d'entre els candidats, per a ser el personatge que disposarà de funcionalitats d'agent conversacional. El disseny del diàleg és **no-lineal** (secció 1.1) per a proporcionar a l'usuari la màxima experiència possible de diàleg lliure. L'usuari pot saludar a l'agent, i aquest, de manera simplificada, en la salutació de resposta inclou les possibilitats de què disposa; entre d'elles, l'usuari pot demanar-li al venedor que li expliqui algun acudit, pot mantenir una conversa personal bàsica, disposa de respostes per a certes preguntes situacionals del jugador bàsiques (com podrien ser 'qui mana aquí?' o 'on haig d'anar?') i la principal raó de ser de l'agent, el fet de vendre objectes.

Al decidir que el diàleg no seria lineal, les formes possibles per a obtenir un objecte del venedor són vàries. El jugador pot demanar de veure els objectes disponibles a la botiga, pot demanar de veure els objectes exclusius d'una categoria en concret (hi ha

tres, destrals, barrets i roba) o pot directament preguntar per l'objecte que busca si en sap el nom o inclús si només sap més o menys el que necessita, per exemple: el jugador ha de tallar un arbre, i té bastants diners; pot anar al venedor i dir-li, 'donam la millor destral que tinguis!'.

Altrament, les capacitats del Vendor no estan sempre disponibles, ja que és un personatge que roman estàtic en una escena del joc (Town). Sorgeix la possibilitat que l'usuari es trobi amb problemes de no saber que fer o on anar, estant en qualsevol altra escena.

D'aquest dilema neix la idea de crear un agent assistent al jugador, que el segueixi de forma visual per allà on es mogui i per tant estigui disponible per respondre qualsevol cosa que l'usuari pugui desitjar preguntar-li sobre la situació actual.

4.2 Requeriments funcionals

En aquest apartat es tracten els nous **casos d'ús**, que tant poden afegir-se al projecte de Fracsland com existir de manera independent.

- **Comunicar-se amb el agent (petició a DialogFlow).**

  Aquest cas d'ús es **comú a tots els agents** tractats en aquest projecte, es tracta del flux de comunicació bàsica amb un agent de Google per part d'un usuari que estigui utilitzant l'aplicació.

Nom:	UC1. **Petició a DialogFlow** a través de un **botó** d'escolta.
Actor:	Usuari.
Descripció:	Procés per a que l'usuari envii la seva petició a l'agent de DialogFlow.
Precondicions:	L'aplicatiu ha iniciat la comunicació amb l'agent.
Flux Bàsic: 1. L'usuari activa l'escolta del agent a través de un botó definit. 2. L'usuari pronuncia la frase que desitja que el bot processi. 3. El bot comprova que la frase no estigui buida i l'envia a DialogFlow. 4. L'agent de DialogFlow contesta i es mostra la resposta obtinguda en una bombolla de text. 5. La resposta es pronuncia de veu mitjançant el sistema de Text-to-Speech.	
Flux Alternatiu: 1a. El sistema falla si no hi ha connexió a internet. Es mostra un missatge d'error. 3a. Si la frase es buida, es mostra un missatge d'alerta: "Warning: Empty message"	
Postcondicions:	L'usuari rep la resposta a la frase que ha enviat a l'agent.

Figura 11: Cas d'ús de comunicació amb l'agent utilitzant un input físic.

Nom:	UC2. **Petició a DialogFlow** a través de la salutació **proactiva** de l'agent.
Actor:	Usuari.
Descripció:	Procés per a que l'usuari envii la seva petició a l'agent de DialogFlow.
Precondicions:	L'aplicatiu ha iniciat la comunicació amb el agent.
Flux Bàsic: 1. L'usuari activa, per proximitat, la salutació proactiva del agent. 2. L'agent saluda. 3. L'usuari contesta/parla amb l'agent. 4. El bot comprova que la frase no estigui buida i la envia a DialogFlow. 5. El agent de DialogFlow contesta i es mostra la resposta obtinguda en una bombolla de text. 6. La resposta es pronuncia de veu mitjançant el sistema de Text-to-Speech.	
Flux Alternatiu: 1a. El sistema falla si no hi ha conexió a internet. Es mostra un missatge d'error. 4a. Si la frase es buida, es mostra un missatge d'alerta: "Warning: Empty message"	
Postcondicions:	L'usuari rep la resposta a la frase que ha enviat a l'agent.

Figura 12: Cas d'ús de comunicació amb el agent utilitzant la salutació proactiva de l'agent.

- **Comprar un objecte usant l'agent**

  Aquest cas d'ús mostra com, dins el projecte **Fracsland**, es produeix la compra d'un objecte qualsevol a través de l'agent **Vendor**.

Nom:	UC3. **Comprar un objecte** qualsevol a través de l'agent **Vendor**.
Actor:	Usuari.
Descripció:	Procés per a que l'usuari obtingui un objecte del joc a través del agent
Precondicions:	L'usuari ha iniciat una comunicació amb l'agent Vendor.
Flux Bàsic: 1. L'usuari demana a l'agent, en llenguatge natural, de veure els objectes disponibles. 2. El bot mostra el menú de la botiga. 3. L'usuari pronuncia el objecte que desitja obtenir. 4. El bot li mostra el objecte. 5. El bot demana en quin tipus de moneda vol pagar el objecte. 6. L'usuari contesta amb el tipus de moneda i la compra es fa efectiva al moment.	
Flux Alternatiu: 1a. L'usuari pot demanar veure una categoria en concret. 1b. Es pot demanar el objecte sense veure els disponibles. 3a. Si el objecte no es reconeix o no existeix, el bot ho fa saber a través del diàleg. 4a. Si el objecte no és el desitjat, l'usuari ha de demanar de nou o pronunciar-ne un altre.	
Postcondicions:	L'usuari rep l'objecte que buscava i s'aplica el cost a les monedes triades.

Figura 13: Cas d'ús d'una compra a l'agent Vendor a traves de DialogFlow.

4.3 Requisits no funcionals

Els requeriments no funcionals per a aquest projecte són bàsicament els esperables en la majoria d'aplicacions dels agents conversacionals. Que la comunicació es dugui a terme en llenguatge natural i no sigui possible per a l'agent identificar la intenció de l'usuari. L'aplicació, però, pot fallar, ja sigui per algun problema espontani en alguna de les eines

de conversió veu-text i viceversa, o per algun problema, extern a nosaltres, en el servidor de DialogFlow, que deixaria sense comunicació l'aplicació del servidor on resideix l'agent. De estar relacionat el problema amb algun mal reconeixement de la veu del usuari, podem contar que l'agent tindrà el seu intent dirigit a quan no comprèn el que l'usuari li diu, d'aquesta forma li demanarà que repeteixi indicant-li que no l'ha entès.

4.4 Requisits tecnològics

Els requisits tecnològics queden llistats i desglossats a continuació:

- **Connexió a internet:**

  Ja que es va decidir utilitzar la plataforma de DialogFlow per a la realització de l'agent conversacional, entre d'altres, per ser accessible i gratuïta, un dels requeriments fonamentals pel funcionament del bot és tenir una connexió a internet en tot moment.

  Totes les peticions i frases han de ser enviades al servidor de DialogFlow, on l'agent les rebrà i les processarà per acabar destriant una resposta que serà enviada de nou en format JSON al software que li hagi enviat la petició.

A part de la connexió a internet, indispensable pel funcionament del servei de Google, tenim el requeriment de què les peticions i les respostes es duguin a terme a través de la veu, ja que la implementació final d'aquest bot al joc de Fracsland va destinat a la versió de Realitat Virtual (VR), fet que impossibilita l'accés còmode a un teclat i invalida l'input per text. La resposta del bot en format text és acceptable, però ja que es pretén transmetre la sensació de conversa el més real possible, existeix el requeriment de tractar la resposta del bot (que arriba en format text) amb algun servei 'Text-to-Speech'.

- **Conversió de veu a text:**

  Per a solucionar el requeriment de convertir la veu de l'usuari, en llenguatge natural, a un format text per a enviar a l'agent, s'ha hagut de trobar una solució externa. DialogFlow disposava d'un servei propi de tractament de veu, podia rebre les peticions al bot en format àudio i des del seu servidor s'obtenia el text de manera transparent al desenvolupador. En treure la nova versió de DialogFlow v2, van retirar aquesta funció de la v1 per a donar alguna exclusivitat a la versió 2 i forçar als desenvolupadors a migrar al nou servei. No es va poder utilitzar la v2 des d'un inici en aquest projecte pel fet que es troba en estat BETA i no hi ha suport amb Unity actualment.

  És per aquesta raó que es va fer necessària una eina externa, i després de buscar diferents opcions, la solució que més efectivitat va demostrar va ser el mateix sistema de reconeixement de veu de Windows, compatible amb diferents idiomes, i la qual des de Unity ja hi ha funcions per a accedir-hi. La prioritat era trobar-ne una de multiplataforma, però de manera gratuïta i de lliure ús no existeix cap ara per ara que funcioni correctament.

  En aquesta resposta StackOverflow (https://stackoverflow.com/questions/39611728/how-to-add-speech-recognition-to-unity-project) es presenten opcions gratuïtes per accedir a les eines natives de cada sistema operatiu dedicades al reconeixement de veu, però requereix la creació d'un plugin propi per al seu funcionament. La de

Windows ja ho té implementat en Unity i per falta de temps es van descartar les altres opcions per als altres sistemes operatius.

- **Conversió de text a veu:**

  Un cop el servidor de DialogFlow obté la petició de l'usuari (de veu, i ja convertida a text) la processa i en respon en format JSON, del qual un dels paràmetres és el 'speech'. Aquest paràmetre és la resposta final del bot en format string dirigida a l'usuari. En aquest punt es va buscar alguna eina compatible amb Unity per poder convertir aquest text a veu.

  Un cop més, i després de descartar vàries opcions per a ser de pagament, la solució més efectiva ha sigut utilitzar els serveis Text-to-Speech que Windows dona als desenvolupadors.

  No hi ha una manera directa d'accedir a aquests serveis des de Unity, per tant s'havia de programar un plugin intermedi per a fer crides a les funcions de l'API de Speech de Windows. Per sort, un programador, de manera altruista, al seu blog explica com es faria i deixa la lliure utilització dels exemples que ofereix. [12]

Un cop tractats tots els requeriments de software, tenim connexió al servei del bot, podem comunicar-nos per veu i rebem la resposta per veu. Únicament falta, com a requeriment de hardware, un micròfon o un dispositiu que el porti incorporat.

5 Disseny

En aquesta secció es tracta el disseny de l'aplicació dels agents DialogFlow al projecte de Unity de Fracsland. Primer definim conceptes i el disseny dels components del programari que utilitzarà als agents (secció 5.1), en aquest cas, de Unity i del joc Fracsland. Seguidament definim els agents de DialogFlow i el disseny que es requereix en cada un d'ells (secció 5.2).

5.1 Disseny del programari

Començarem definint cadascuna de les parts que participen en aconseguir una comunicació completa amb l'agent, des de que l'usuari pronúncia de veu el que vol dir-li a l'agent, fins que l'usuari rep de veu la resposta de l'agent. A la Figura 14 es mostren aquestes parts. A continuació les detallem.

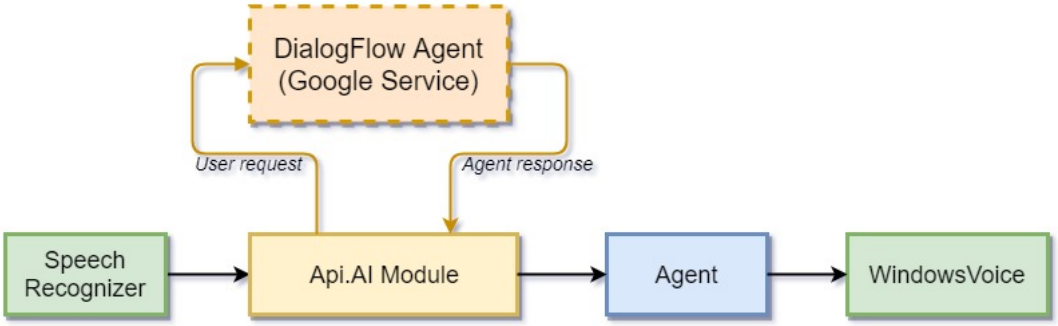

Figura 14: Diagrama bàsic amb els nous components i la seva comunicació

El diagrama que mostra la Figura 14 representa l'estructura de components bàsics necessaris per al funcionament dels agents per veu dins l'entorn de Unity, a continuació estan explicats. A la secció 4.2 es poden veure els casos d'ús específics de Fracsland.

- **Speech Recognizer (Veu a text):**

  Aquest script utilitza l'eina nativa de Windows des de Unity (*UnityEngine.Windows.Speech*) per a efectuar el reconeixement de veu, per tant, el llenguatge que reconeix per defecte, i altres opcions, s'especifiquen a través de la configuració de Windows, a l'apartat de 'Veu, regió, data', com es mostra a la Figura 15.

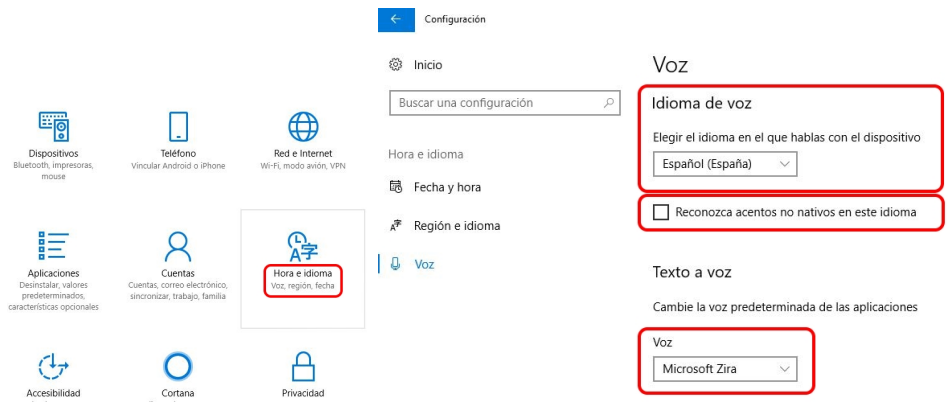

Figura 15: Opcions de veu, a la configuració de Windows 10.

La seva funció és la d'escoltar a l'usuari i transcriure les seves paraules. Per tant està bastant preparat per a treballar de forma independent i de fàcil adaptació a qualsevol projecte.

En el cas d'haver més d'un agent a l'escena, en aquest projecte s'ha decidit que aquest component s'encarregarà de decidir a quin agent va dirigit el text transcrit. Aquesta decisió és deguda al fet que inicialment es va dissenyar per a un sol agent, i l'agent Assistent va plantejar-se amb menys termini abans de la finalització del projecte, quan el disseny ja estava fet. La millor manera de gestionar aquesta situació seria la de definir un nou component 'Agent Controller' per a gestionar aquesta tria del bot actiu, i d'altres plantejaments que podrien sortir en projectes diferents.

- **Api.AI Module (DialogFlow)**

  Aquest component representa a un script anomenat **ApiAiModule.cs**, que juntament amb el 'plugin' que integra l'API de Api.AI (DialogFlow v1)[3] a Unity, disposa de funcionalitats per a enviar, al servei de Google, peticions en format veu i rebre'n la resposta de la mateixa manera; però a l'hora d'utilitzar-les es dispara un missatge d'error que informa que no s'accepta aquest tipus de petició. Tan sols queda la funció per enviar peticions de text i capturar la resposta de la mateixa forma.

  A la documentació del 'plugin' es tracta a ApiAiModule.cs com el script personalitzable que s'ha d'adaptar a les situacions en les quals s'integri. Per aquest motiu és un script més difícil de fer de forma modular. Aquest script està pensat per estar relacionat amb el script que porta la lògica de l'agent dins el joc (representat pel component Agent de la Figura 14), i s'encarrega de trametre la 'action' que retorna DialogFlow per a que l'agent (dins el joc) faci l'acció coordinada amb el que li està dient a l'usuari [4].

  En el cas de tenir més d'un agent al projecte, com és el cas, es necessita d'un *Api.AI Module* per a cada un dels agents, cadascun amb connexió amb l'agent que li pertoca.

- **Agent**

  Aquest component representa el script associat a cada agent, el qual porta la lògica de les accions, moviments i tot el relacionat amb l'objecte que representa l'agent dins el joc. En el cas del Vendor, al tractar-se d'un personatge, amb un script propi, ja existent en el projecte original de Fracsland, s'explica la seva nova funcionalitat a la secció '5.1.3 Canvis en respecte al disseny inicial', on s'explica, entre altres canvis, la funcionalitat de la botiga associada a les respostes que es reben de DialogFlow.

  Per al que respecta a l'agent Assistant, es va decidir que l'agent havia de seguir al jugador allà on anés, i que a l'hora de reproduir les respostes, es situés en un punt concret abans. Totes aquestes mecàniques es gestionen des d'aquest component, juntament amb com les 'actions' rebudes de DialogFlow afecten el comportament de l'agent. Per a l'agent Assistant es consideren tan sols els casos de 'command', on l'agent no pronuncia la resposta de veu ni la mostra en text (s'utilitza de forma

[3]El script ApiAiModule.cs forma part del plugin que integra l'API de DialogFlow V1 a Unity. Disponible en aquest enllaç: *https://github.com/dialogflow/dialogflow-unity-client*

[4]El concepte de 'action' així com la seva utilització estan explicats a la secció 1.1.

interna per a mostrar informació en la fase de programació), i 'bad' i 'good' que s'utilitzen per a mostrar un feedback visual.

- **Windows Voice (Text a veu)**

  En la part del pas de text a veu, un cop més apareix la limitació de la pèrdua de suport, i suspensió, per part de Google, de les funcionalitats relacionades amb la veu a DialogFlow V1. La solució es va trobar gràcies a una web on Chad Weisshaar [12], un programador independent, explicava com comunicar amb l'eina nativa de Windows per a accedir a les funcionalitats de text a veu, tal com s'ha explicat a la secció 4.4 Requisits tecnològics, a l'ítem de 'Conversió de text a veu'.

  El script amb el qual es comunica des de Unity s'anomena **WindowsVoice.cs**[5] i funciona de la següent manera:

  És una classe Singleton, per tant només hi pot haver una sola instància executant-se, és a dir, tots els scripts que requereixin d'executar veu hauran de fer-ho al mateix objecte. Tenint en compte això, el script implementa una cua de tipus 'Queue' per a les frases que ha d'enviar a reproduir per l'eina de Windows.

  Recordar que aquest script és tan sols una interfície per comunicar amb l'eina de Windows i no disposa de cap altra funcionalitat, per tant, és completament modular i aplicable en qualsevol projecte de Unity. La funció per a parlar requereix d'un *string*, i degut a les funcionalitats que Windows ofereix amb la seva eina, aquestes peticions en format text poden contenir 'tags' XML per a definir certs paràmetres, com poden ser, el nom de la veu (a triar entre les que l'usuari hagi descarregat, a la configuració de Veu de Windows) o el pitch de la veu, inclús la velocitat de pronunciació.

  Un exemple de petició podria ser:

```
<VOICE REQUIRED='Gender = Male; Age != Child; language = 409'>
                              + text+
                             </VOICE>
```

  Aquesta petició estaria especificant: Veu masculina, que no sigui de nen petit, i llenguatge 409 és el codi per a l'anglès.

5.1.1 Diagrames de seqüencia

En aquest apartat es troben els diagrames de seqüència per a dues tasques concretes, la primera molt general, és el mecanisme per a enviar un missatge a DialogFlow i obtindre'n la resposta, el segon és més específic, concretament del venedor, és el procés per a comprar un objecte a la botiga utilitzant la veu i l'agent conversacional.

[5] Aquest script, així com l'explicació per a implementar-ne un de personalitzat, es troben en el següent enllaç: *http://www.chadweisshaar.com/blog/2015/07/02/microsoft-speech-for-unity/*

- **Petició a DialogFlow a través d'un botó.**

  A través de components no representats en aquest diagrama, un cop es crida a *SetAction(AIResponse)* de Vendor, també es mostra la resposta en una bombolla de text.

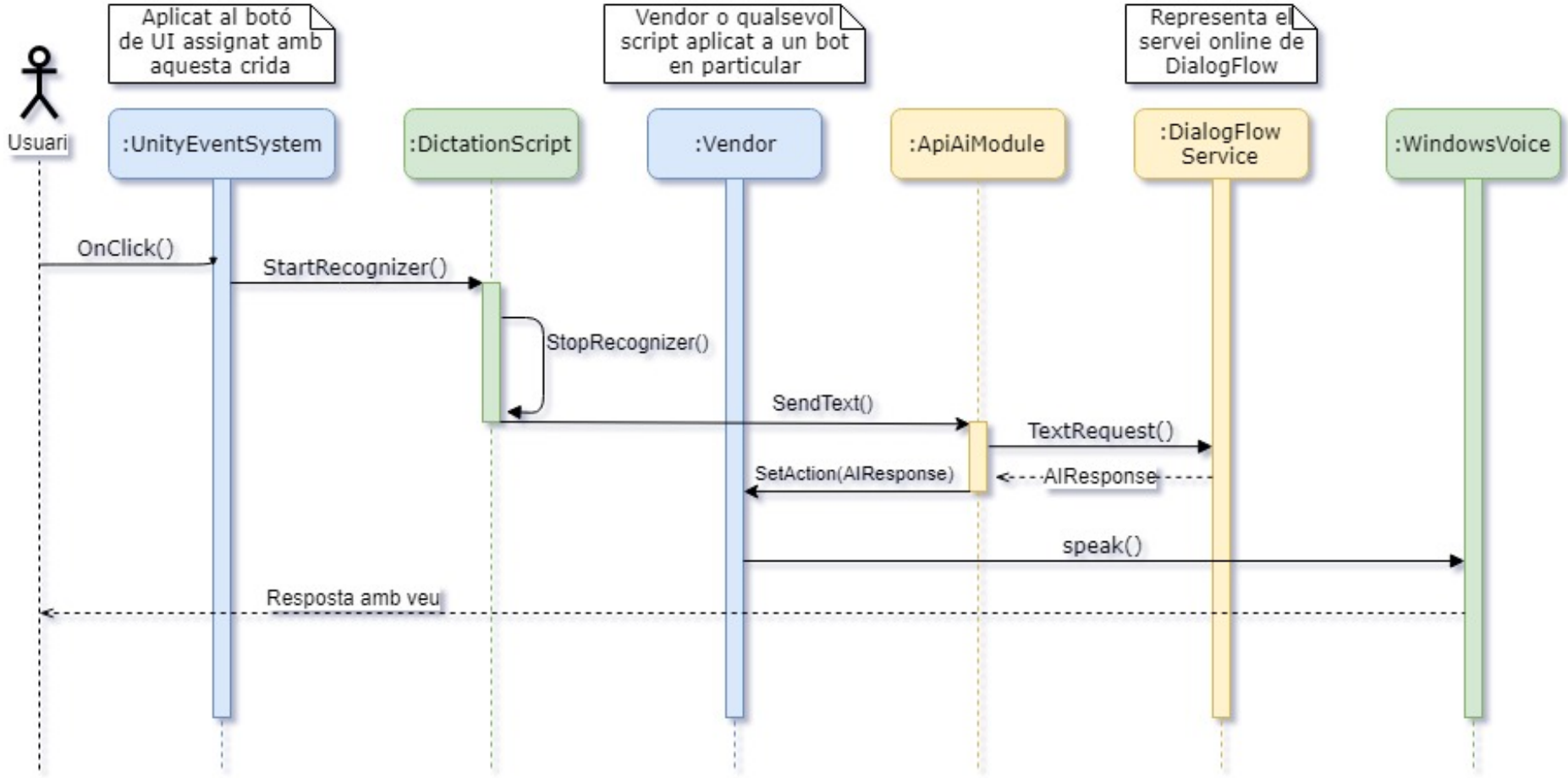

Figura 16: Diagrama de seqüencia del procés d'enviar una petició qualsevol a DialogFlow i obtindre'n la resposta.

- **Comprar un objecte qualsevol a través de l'agent Vendor.**

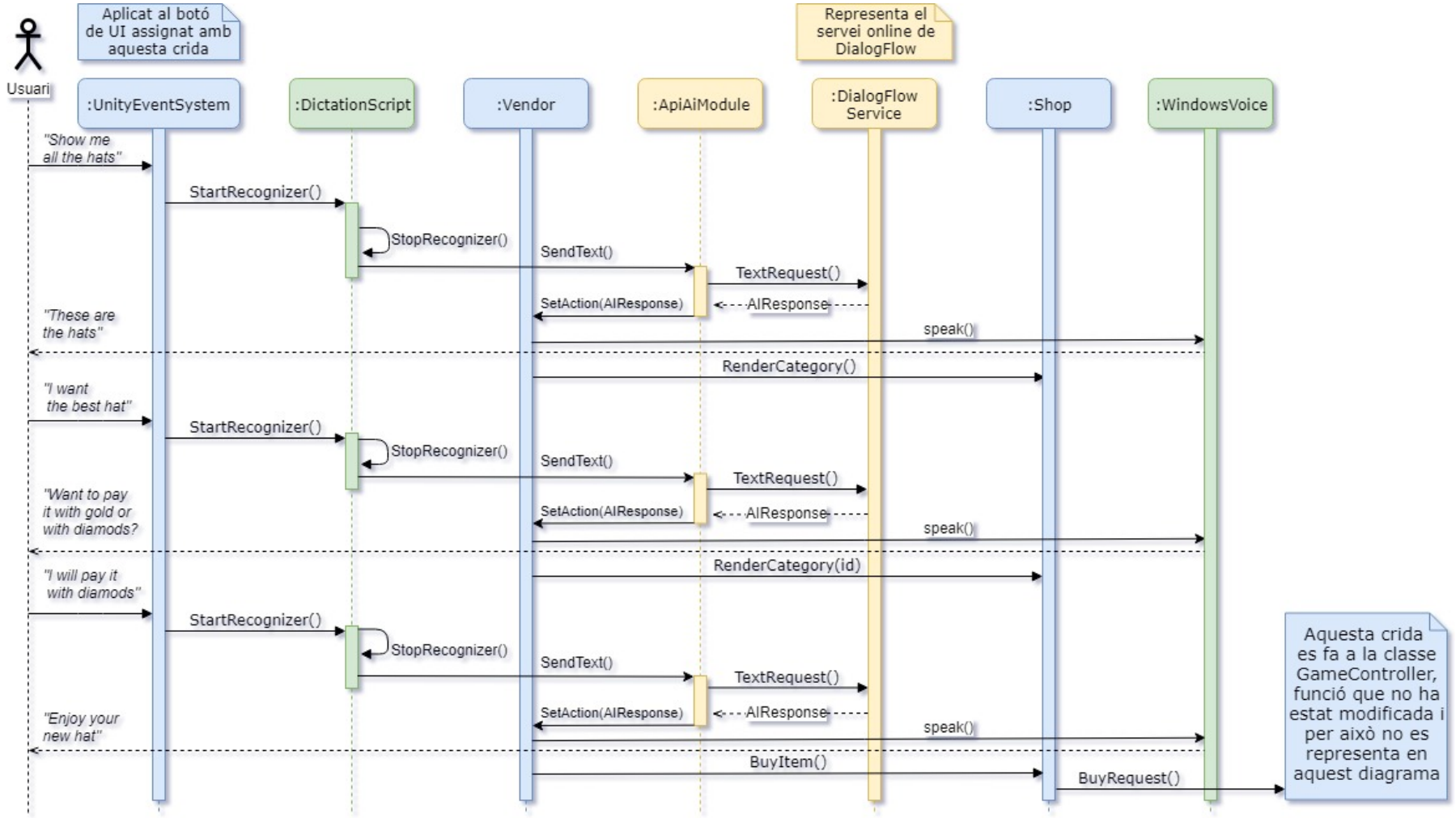

Figura 17: Diagrama de seqüencia del procés de comprar el millor barret, i pagar-lo en diamants.

5.1.2 Diagrama de classes

A continuació es troba a la Figura 18 el diagrama de classes per als nous elements i els ja existents que han patit modificacions. En color verd es troben les classes involucrades en la conversió de veu a text i viceversa (`DictationScript` i `WindowsVoice`); de color groc, els relacionats amb la comunicació amb el servei de l'agent conversacional, on es mostra l'únic script que s'ha de modificar del puglin de DialogFlow (`Assistant ApiAiModule` i `Vendor ApiAiModule`, un per a cada agent); finalment, en blau es troben les classes referents a les lògiques i mecàniques del joc[6] (`Assistant`, `Vendor` i `Shop`).

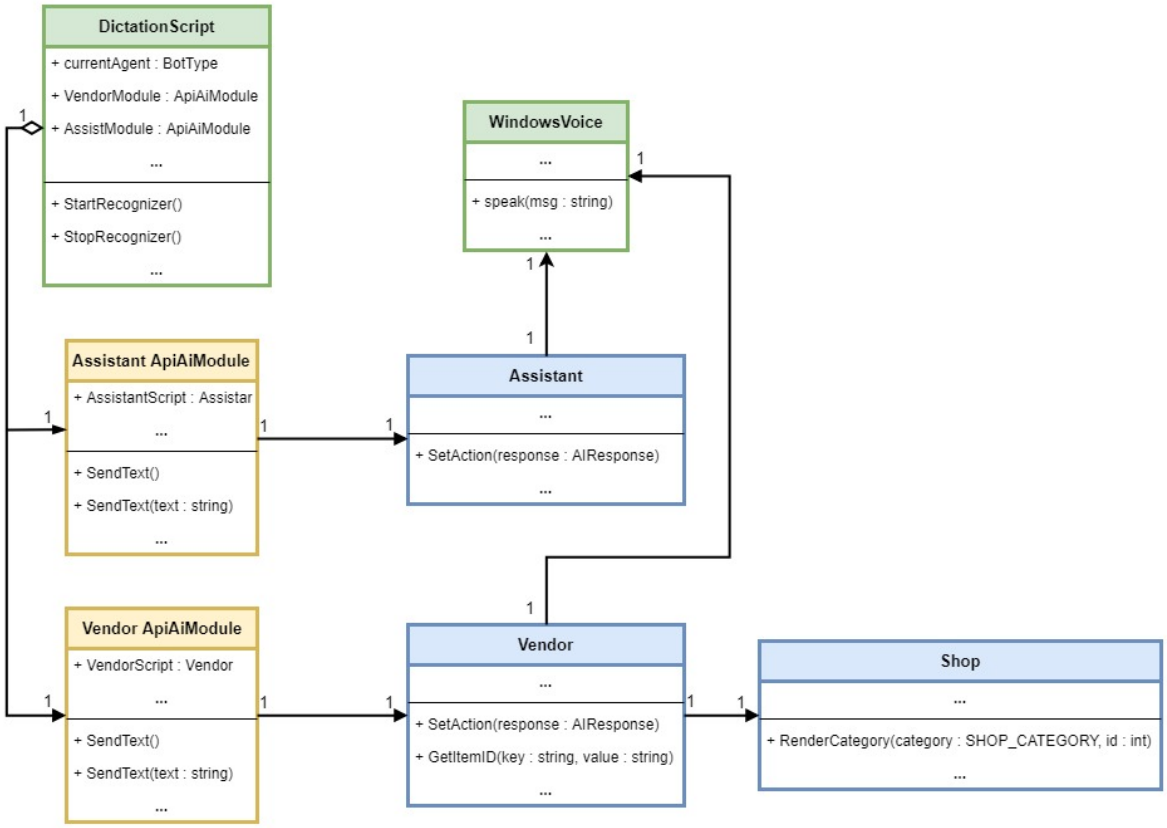

Figura 18: Diagrama de classes dels nous components juntament amb els ja existents a Fracsland).

5.1.3 Canvis respecte al disseny inicial

Un cop presentat el diagrama de classes, en aquest apartat es tracten els principals canvis produïts respecte al projecte original, centrats en les classes representades de color blau.

Els canvis en el codi de Fracsland per a inserir un agent al personatge 'Vendor' són mínims. A part d'afegir nous components al projecte, l'únic punt on hi ha hagut canvis en elements ja existents és al script `Vendor.cs` i per consegüent al script `Shop.cs`.

Aquest script `Vendor.cs`, en un inici, tan sols contenia codi referent a les animacions del personatge, de les quals només s'utilitzaven dos. Una funció per saludar i mostrar un text en una bombolla, i una funció que activava l'animació en bucle d'espera o '*idle*'; obrir la botiga es feia des d'un altre script al clicar al venedor.

[6] Vendor i Shop són dos scripts ja existents al projecte original de Fracslands, per tant seran modificats i adaptats.

Per al funcionament de l'agent d'una manera correcta i controlada es va decidir passar la funcionalitat d'obrir/tancar botiga al script del Vendor. En la següent secció s'explica l'estructura que s'utilitzarà per al funcionament de l'agent i la comunicació amb DialogFlow; ara ens centrarem en la part de `Vendor.cs`, que necessita d'una altra funció molt important per al funcionament com a agent conversacional. Vendor ha de rebre les accions que l'agent de DialogFlow li remet, per a poder realitzar certes accions que tinguin coherència d'acord amb el que diuen les respostes que envia DialogFlow.

Per il·lustrar-ho; l'usuari pot demanar de veure els barrets de la botiga, DialogFlow rep aquesta petició i contesta: 'aquí tens els barrets...', juntament amb una 'action' anomenada '*ShowHats*', aleshores, el personatge a Unity ha d'obrir la botiga[7], no només mostrar la resposta. Per tant el canvi més important és el fet que el venedor té una funció, anomenada **SetAction(AIResponse r)** , que es crida sistemàticament a cada resposta de DialogFlow que capturem des del '*Api.AI Module*' del Vendor, i que destria quina acció realitza el nostre NPC. La tria es fa mitjançant un conjunt de condicionals amb els 'tags' concrets i coneguts, ja que s'han introduït prèviament com a nom de les 'actions' en l'agent de DialogFlow (recordar la definció de 'action' a la secció 1.1), i que s'enumeraran juntament amb els intents a la secció 5.2.2 .

Una altra funció que s'afegeix a *Vendor.cs* es **GetItemID()** que s'encarrega de, quan l'usuari sol·licita veure un objecte en concret, convertir el paràmetre que DialogFlow adjunta amb la resposta, en un enter per a després cridar al script de la botiga i demanar que només mostri l'objecte en concret (allà es realitza la conversió de ID a Objecte).

Així doncs apareix l'últim dels canvis, al script *Shop.cs* encarregat de mostrar la botiga i la categoria que es seleccioni. S'afegeix una nova versió de la funció **RenderCategory()**, en la qual hi ha un paràmetre extra, anteriorment es seleccionava la categoria, en aquesta nova funció es selecciona la categoria i es passa una ID d'objecte, si la ID és zero, és el funcionament normal de la funció, mostrar la categoria íntegrament.

De manera menys important, parlant de l'impacte en la funcionalitat, queden per mencionar les funcions del venedor per a activar més d'una animació que no estava sent utilitzada. Aquestes funcions són molt similars entre elles, però estan orientades a ser utilitzades en diferents moments. L'agent disposa d'una llista d'animacions predefinides, per tant podem triar la que es vulgui, en el nostre cas, per simplificar només es tenen en compte una per mostrar alegria i una altra per parlar per defecte. També mencionar que és on es fa la crida per pronunciar de veu la resposta i alhora mostrar-la per text a través d'una bombolla de diàleg, que permet determinar el temps que es mostra la bombolla amb la resposta de l'agent.

5.2 Disseny dels Agents de DialogFlow

Els dos agents comparteixen el fet d'estar dissenyats per a mantenir diàlegs no-lineals, de forma que, en la majoria dels casos, l'usuari pot canviar el tema de conversació sense restriccions, i l'agent s'hi adapta. Les particularitats dels agents no-lineals han estat detallades a la secció 1.1.

[7]El personatge Vendor, al joc Francslands, disposa d'un menú botiga, amb objectes per vendre, que originalment s'obria en clicar al personatge. Ara aquesta acció es durà a terme a través de l'agent conversacional.

5.2.1 Funcionament de DialogFlow

En aquesta secció s'introdueixen, de forma general, els agents de DialogFlow, tenint en compte les definicions dels conceptes relacionats de la secció 1.1. La creació d'un nou agent de DialogFlow és senzilla, des de la plataforma de Google es poden crear en uns pocs clics, després s'han de començar a definir els intents necessaris acord amb el disseny de l'agent que vulguem crear, tant les frases model que dirà l'usuari per activar l'intent, com les possibles respostes que l'agent donarà. Juntament amb els intents, s'han de definir el conjunt d'entitats les quals el nostre agent ha de tractar. Finalment s'apliquen els contexts per a dirigir el flux de la conversa i la informació que es guarda, mitjançant els contexts, però també declarant els paràmetres necessaris als intents amb context.

DialogFlow ens respon en un format JSON, el qual el programa que l'utilitzi, en el nostre cas Unity, ha de desglossar i capturar els valors que es necessiten. A la Figura 19 es mostra alguns dels camps de més importància dels que ens retorna DialogFlow.

```
"result": {
  "source": "agent",
  "resolvedQuery": "user's original query to your agent",
  "speech": "Text defined in Dialogflow's console for the intent that was matched",
  "action": "Matched Dialogflow intent action name",
  "actionIncomplete": false,
  "parameters": {
    "param": "param value"
  },
  "contexts": [
    {
      "name": "incoming context name",
      "parameters": {},
      "lifespan": 0
    }
  ],
```

Figura 19: Format dels camps del JSON que ens envia com a resposta DialogFlow.

Utilitzarem el camp *speech* com a la frase per a reproduir de veu. El camp *action* conté el 'tag' que s'utilitzarà perquè el personatge dins de Unity faci alguna funció. De *parametres* es poden extreure els paràmetres que existeixen en l'àmbit d'aquest intent, però per consistència utilitzarem els paràmetres continguts dins de *contexts*, ja que són els que es transmeten entre intents, i dels que es vol extreure la informació.

5.2.2 Agent Vendor

L'agent Vendor disposa de diverses funcions que es poden agrupar en dos blocs, les relacionades a la botiga de Fracsland i les relacionades amb interaccions socials diverses. Per a comprendre aquesta secció és important d'haver entès els conceptes relacionats amb DialogFlow, així com recordar que un intent és un 'mapeig' entre el que l'usuari diu i l'acció que ha de dur a terme el software.

A continuació s'enumeren tots els intents que han estat definits per a l'agent Vendor, juntament amb el 'tag' definit en cadascun dels seus camps 'action'.

1. ComeCloser.
   Action: Social.Closer
2. Social.Greetings.
   Action: Social.Greetings
3. Social.Greetings (w Purchase).
   Action: Social.Greetings
4. Shop.ShowAllHats.
   Action: Shop.Show.Hats
5. Shop.ShowAllAxes.
   Action: Shop.Show.Axes
6. Shop.ShowAllSkins.
   Action: Shop.Show.Skins
7. Shop.ShowOneHat.
   Action: Shop.Buy.Hat
8. Shop.ShowOneAxe.
   Action: Shop.Buy.Axe
9. Shop.ShowOneSkin.
   Action: Shop.Buy.Skin
10. Shop.NoMoney.
    Action: Shop.NoMoney
11. Shop.ConfirmPurchase.
    Action: Shop.Purchased
12. Social.Jokes.get.
    Action: Social.Jokes.Get
13. Social.Jokes.get.more.
    Action: Social.Jokes.Get.More
14. Social.Jokes.Bad.
    Action: Social.Jokes.feedback.bad
15. Social.Jokes.Good.
    Action: Social.Jokes.feedback.good
16. Social.Bye
    Action: Social.Bye.NoPurchase
17. Social.Bye (w Purchase)
    Action: Social.Bye.Purchased

A continuació s'expliquen cadascun d'ells. El primer dels *intents* a tractar és el de la salutació proactiva [8]. És un *intent* que s'activa mitjançant un '*trigger*', quan el jugador s'apropa, Unity, de manera transparent a l'usuari, enviarà un missatge/comanda[9] concret a DialogFlow, activant el *intent* 'ComeCloser' (nº 1), que saludarà a l'usuari i li dirà que s'apropi.

El diàleg per les funcionalitats de la botiga s'ha dissenyat de la següent forma: Es disposa de 3 *intents* (nº 4, 5 i 6) per a mostrar, respectivament, les 3 categories disponibles d'objectes (barrets, destrals i roba), disponibles des de la majoria de punts de la conversació. Aquests *intents* no requereixen cap context ni en generen cap de nou.

[8] Acció proactiva: Acció que s'anticipa a problemes o necessitats futures sense l'acció directa del jugador.

[9] Es tracta d'un missatge com qualsevol que l'usuari pogués enviar, la particularitat és que va comprès entre '[ ]' per evitar que l'usuari pogués activar-lo accidentalment. L'intent 'ComeCloser' únicament accepta una sola frase d'entrada. És per aquest motiu que s'anomena comanda, ja que només s'accionarà aquest intent a través d'uns caràcters concrets.

Hi han 3 *intents* més (nº 7, 8 i 9), un per a cada una de les categories, per a mostrar un sol objecte en concret, indicat per al jugador. Aquest intent li requereix a l'usuari d'indicar l'objecte en concret que desitja, a part del tipus de moneda que utilitzarà pel pagament. Ambdós paràmetres poden estar continguts en una mateixa frase, com es mostra en la Figura 20, on es mostren les frases d'entrenament per a un dels intents comentats, 'ShowOneHat' (nº 7). L'agent de DialogFlow, a mesura que va entrenant-se, aprèn a combinar les diferents frases i models d'entitats que es poden esperar de l'usuari per a un determinat *intent*.

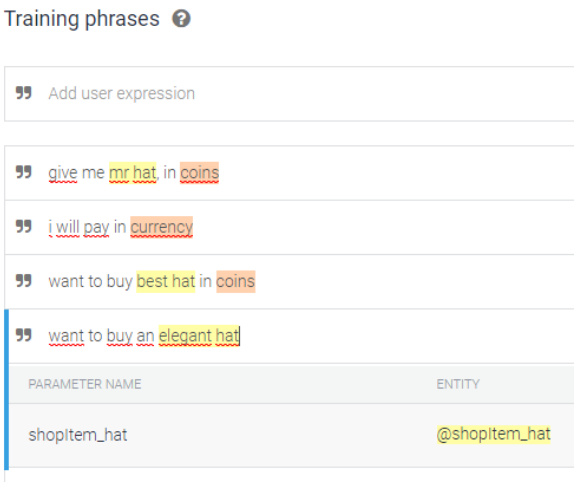

Figura 20: Frases d'entrada del intent 'ShowOneHat'.

A la Figura 21 es mostra com indicar a DialogFlow, que els paràmetres són d'obligada resposta. De no trobar-se algun d'aquests '**paràmetres requerits**', DialogFlow ens permet definir uns missatges, que s'enviaràn de forma automàtica, demanant la informació que falta, com es mostra a la Figura 22.

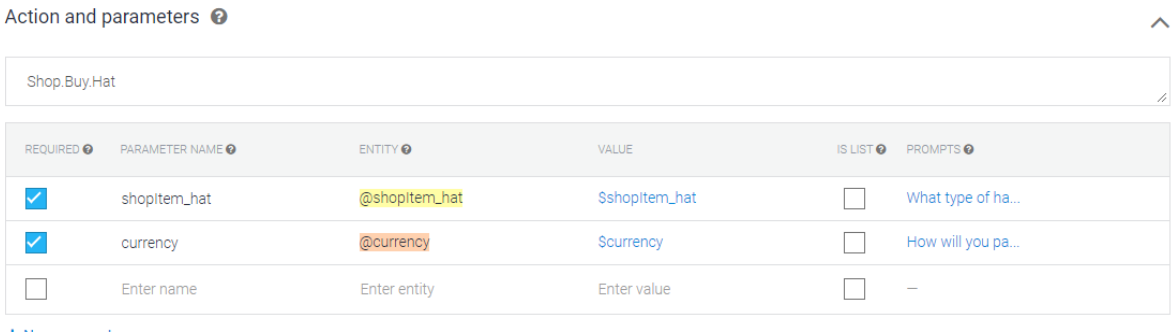

Figura 21: Paràmetres marcats com a 'Requerits' per a finalitzar el *intent*.

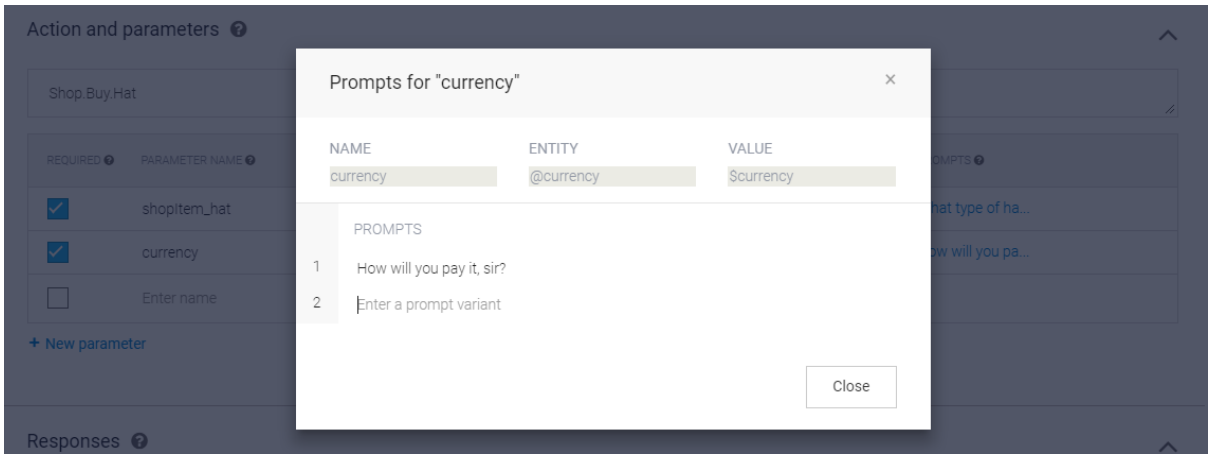

Figura 22: Menú on es determinen les frases que es mostren l'usuari si no introdueix algun paràmetre requerit.

Aquests últims *intents* mencionats (nº 7, 8 i 9) tenen un context de sortida, anomenat 'objectSelected', on es guarden els paràmetres introduïts per l'usuari. Aquest context de sortida és el context d'entrada (i per tant, condició obligatòria per entrar) a dos *intents* anomenats 'Shop.ConfirmPurchase' (nº 11) i 'Shop.NoMoney' (nº 10).

Es tracten de dos *intents* amb una peculiaritat particular, no estan pensats per a ser accionats per l'usuari, sinó per la lògica de Unity; això és pel fet que DialogFlow no pot sostenir càlculs ni comprovacions sobre funcions específiques del codi de les diferents aplicacions que l'utilitzen.

Per tant, quan des de Unity, mitjançant la informació en el camp '*action*' de cada intent (el format i la informació rellevant de la resposta de DialogFlow s'ha tractat a la secció 5.2.1.), detectem que l'usuari ha demanat per un objecte i ha indicat amb quin tipus de moneda ho vol pagar, calculem si té prou diners (del tipus que ha elegit) per a l'objecte que ha elegit, recuperat del context, que està inserit dins la resposta de DialogFlow; i enviem a l'agent un '**[yes]**' o '**[no]**' en funció de si es pot comprar o no.

El missatge '[yes]', juntament amb el context 'objectSelected', accionarà l'intent per a completar la compra. Mentre que '[no]', també amb el context que indica que s'ha passat per l'intent de seleccionar un objecte, acciona l'intent que informa al jugador de què no té prou diners.

A continuació, a la Figura 23 podem veure un exemple de diàleg per a comprar el barret elegant.

Usuari:	Hola!
Vendor:	Benvingut a la botiga!
Usuari:	Mostrem els barrets que tens disponibles.
Vendor:	Aquí tens els barrets..
Usuari:	Vull el barret elegant!
Vendor:	Bona elecció! Vols pagar-lo en monedes o en diamants?
Usuari:	En monedes.
Vendor:	Perfecte, el barret elegant en monedes.
Vendor:	Anem a comprovar que tinguis prous diners...
Unity:	[yes]
Vendor:	Tot correcte, gaudeix del teu nou barret elegant!

Figura 23: Exemple de diàleg per a comprar el barret elegant.

Existeixen dos *intents* de salutació (nº 3) i d'acomiadament (nº 17), amb el context 'objectSelected' d'entrada, d'aquesta forma, podem extreure els paràmetres dels contexts i saludar o acomiadar a l'usuari amb un comentari personalitzat depenent de l'objecte que hagi adquirit, com es mostra a la Figura 24 i com s'ha explicat a la secció 1.1, a la subsecció de contexts.

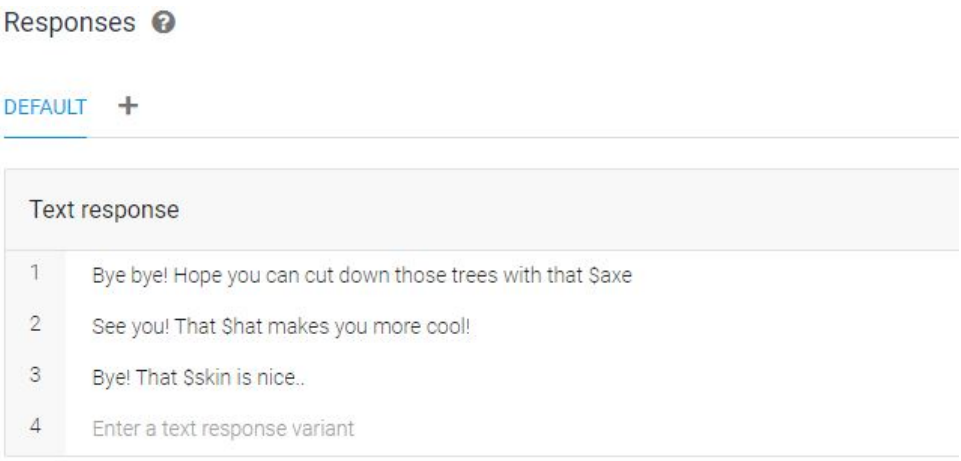

Figura 24: Respostes que contenen paràmetres extrets d'un context.

Fins aquí l'explicació del flux del diàleg dels *intents* relacionats amb la botiga, per als de caràcter social, no hi ha cap ordre, l'usuari pot saludar, acomiadar-se, preguntar qualsevol cosa o demanar algun acudit a l'agent en qualsevol moment.

Dins el cas dels acudits (nº 12), es crea un context el qual provoca que l'usuari entri en un sub-*intent* (nº 13), des d'on s'espera rebre o feedback positiu (nº 15), o negatiu sobre la broma (nº 14), o una altra petició de broma addicional. A la Figura 25 podem veure un exemple del flux del diàleg en el cas de les bromes.

Usuari:	Hey!
Vendor:	Benvingut a la botiga!
Usuari:	Expliquem alguna broma.
Vendor:	La meva mare era tant sobreprotectora que només ens deixava jugar al 'pedra, paper'.
Usuari:	Que ocurrent!
Vendor:	Gràcies! En vols sentir una altra?
Usuari:	Si.
Vendor:	El meu cap m'ha explicat tants acudits dolents avui que no recordo ni com era el meu riu
Usuari:	Aquesta ja es més dolenta...
Vendor:	Disculpa.. Segur que en tinc de millors, només pregunta.

Figura 25: Exemple de diàleg amb una broma de l'agent.

A continuació, a la Figura 26 es pot veure en global, com és el flux de diàleg possible amb l'agent Vendor. Com es pot comprovar, totes les branques del diàleg convergeixen en un punt, que representa la unió entre totes les funcionalitats de l'agent i com el flux del diàleg pot passar de un tema a un altre.

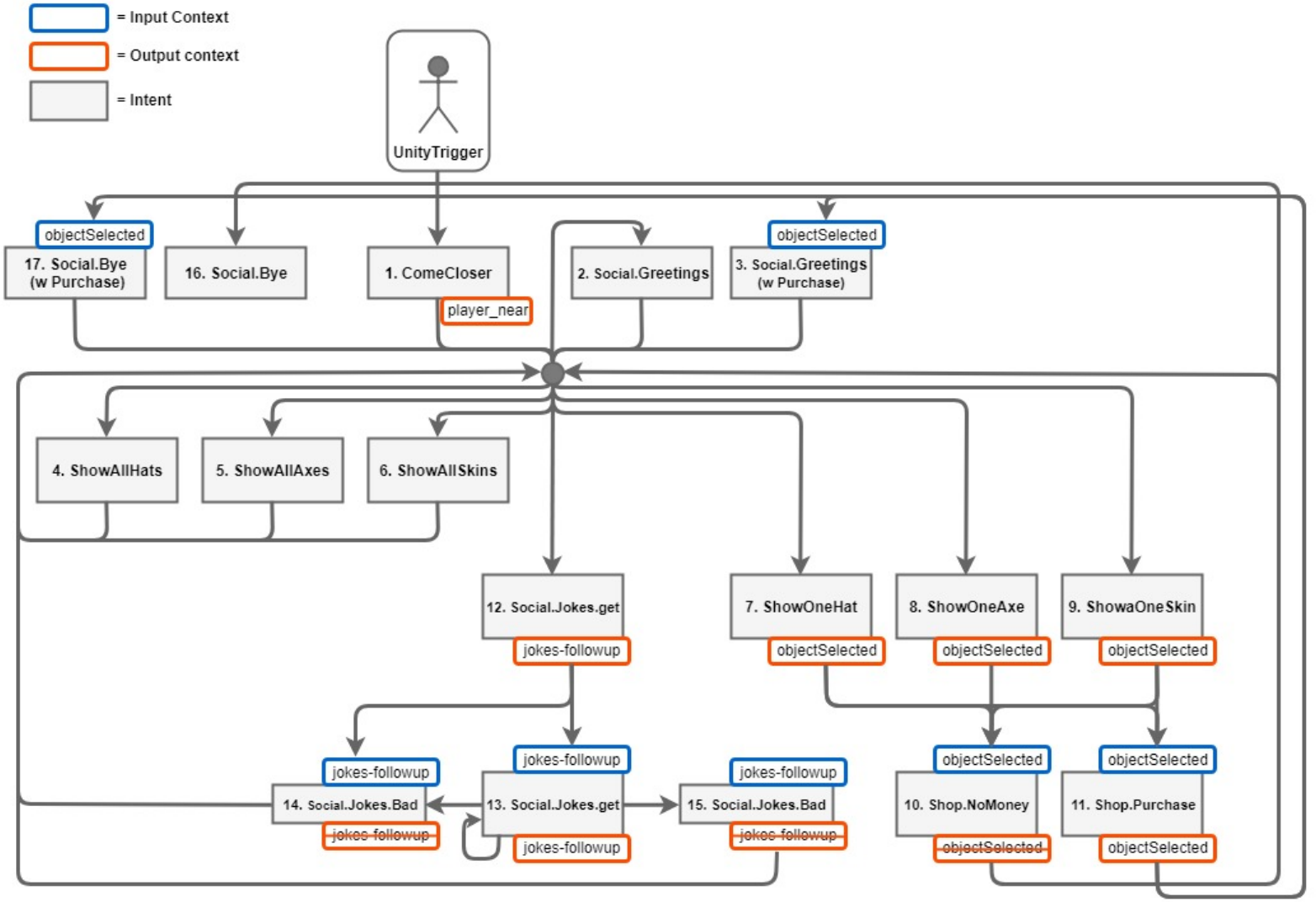

Figura 26: Diagrama del flux de diàleg entre els diversos intents del agent Vendor.

5.2.3 Agent Assistant

La principal funcionalitat de l'agent Assistent és la de donar informació l'usuari sobre el mapa de Fracsland on es troba. A part d'aquesta funcionalitat particular, també disposa, de la mateixa forma que l'agent Vendor, d'un conjunt de funcions socials, les quals requereixen molt poc esforç de disseny, ja que són, per el general, *intents* que funcionen de manera independent, és a dir, se'ls pot cridar en qualsevol moment del diàleg i no requereixen de l'ús de contexts.

En aquesta secció ens centrarem principalment en els *intents* més específics de la funcionalitat del agent Assistant, i com aquest utilitza comandes transparents al usuari per accionar de forma automàtica certs intents. La principal funcionalitat de l'agent és la de transmetre informació en moments que l'usuari no sàpiga que ha de fer. Una de les formes mes directes de fer-ho és dotar a l'agent de consells i 'tips' per a cada zona del joc, que són: Town, Farm, Beach i Forest.

Per a dur a terme aquesta tasca de obtenir ajudes particulars a cada escenari possible, s'han dissenyat un seguit d'*intents* específics per a cada mapa del joc. En qualsevol moment que l'usuari digui al bot frases com '*no se que haig de fer*' o '*quina es la missió?*', l'agent contesta amb la missió associada al mapa on es troben, en el cas de tenir una missió activa, sinó remet al jugador a Lord Barus, el personatge que assigna les missions.

Addicionalment, com s'ha comentat, s'han dissenyat un seguit d'*intents* per a ser accionats de forma proactiva sense que el jugador ho accion-hi directament. Quan el jugador entra en una nova zona, el codi de Unity envia una comanda, concreta i única, a DialogFlow per activar un *intent* específic per a la zona, en que d'una forma poc intrusiva, l'agent fa un petit comentari sobre la zona, del estil: 'No m'agraden els llops..'.

A la Figura 27 es poden veure alguns dels *intents* del agent, es destaquen els següents:

- En vermell (`_CreateMissionContext` i `_ClearMissionContext`):

  Es tracta de dos *intents*, utilitzats de forma automàtica per la lògica del joc, creen un context si el jugador té alguna missió activa, quan la supera, s'elimina. D'aquesta forma podem discernir entre la situació en què el jugador demana informació tenint alguna missió o no. Com s'ha comentat anteriorment, si no hi ha missió activa, l'agent farà algun comentari sobre la zona on es troba amb el jugador, però després recomanarà anar a veure a Lord Barus per obtindre'n alguna.

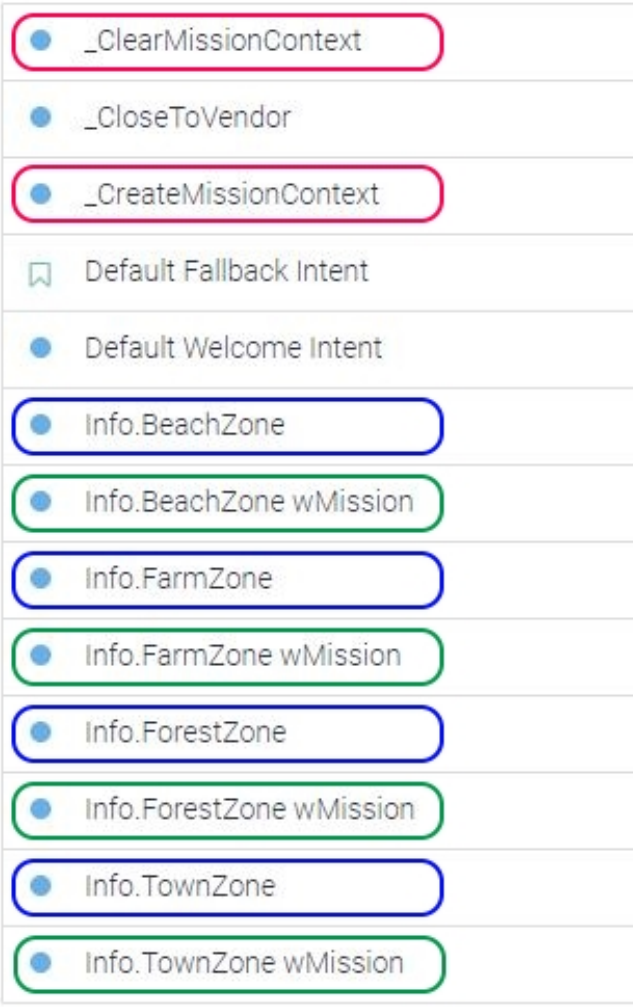

Figura 27: Alguns dels intents més significatius de l'agent Assistant.

- En blau (`Info.BeachZone`, `Info.FarmZone`, `Info.ForestZone` i `Info.TownZone`):

  Trobem els intents dirigits pel cas de no tenir el context de missions actiu, les respostes són comentaris sobre les estètiques del mapa o sobre el que està visualment a l'escena, juntament amb la recomanació d'anar a visitar a Lord Barus per obtenir alguna missió.

- En verd (`Info.BeachZone wMission`, `Info.FarmZone wMission`, `Info.ForestZone wMission` i `Info.TownZone wMission`):

  Finalment en verd estan els intents que tenen com a condició d'entrada el context de les missions, per tant només són accessibles quan el jugador disposa d'alguna missió activa. Les respostes són pistes o comentaris sobre les missions específiques per a cada zona i com resoldre-les.

Per a que les mateixes frases, per part de l'usuari, activin els consells, sigui quina sigui la zona del joc on es trobi el jugador, s'ha utilitzat la següent tècnica: els *intents* són activats per pràcticament les mateixes 'Training Phrases' com es pot veure a la Figura 28, amb la particularitat que porten un 'tag' (com per exemple [Town] o [Farm]) que s'afegeix automàticament des de la lògica de Unity depenent de l'escena on es trobi el jugador al moment.

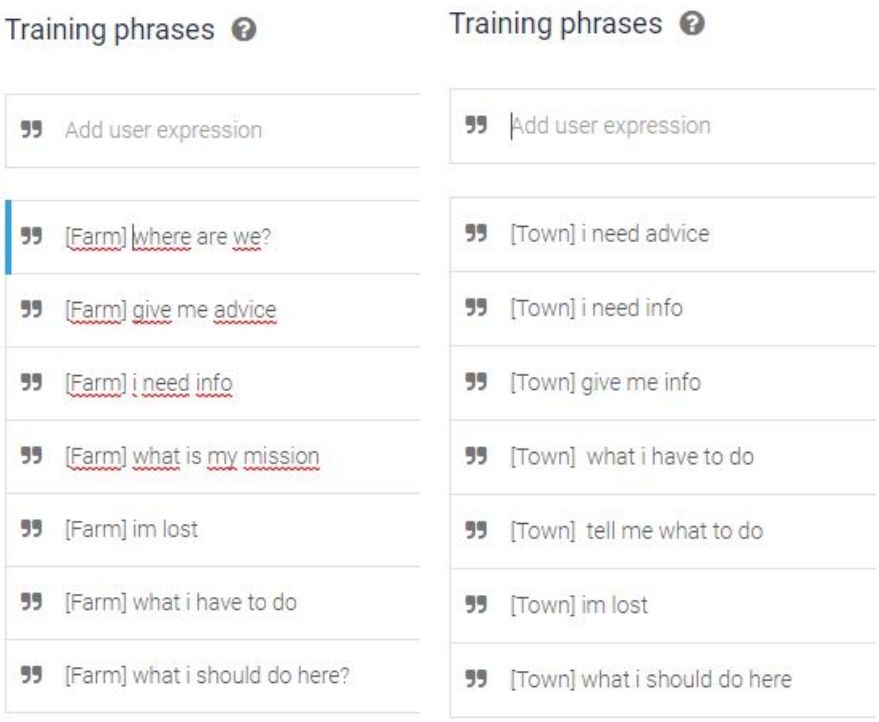

Figura 28: Els intents de cada zona son activats per les mateixes frases de l'usuari, la lógica del joc afegeix un 'tag'.

A continuació, a la Figura 29, trobem dos exemples del que comentàvem anteriorment. El primer d'ells tracta el cas de l'usuari demanant ajuda sense cap missió activa, l'agent el dirigeix a Lord Barus. En el segon cas hi ha una missió activa (perquè li ha donat Lord Barus), però l'usuari segueix a la zona de Town[10], aleshores l'agent l'indica que segueixi la direcció que marca la fletxa verda[11].

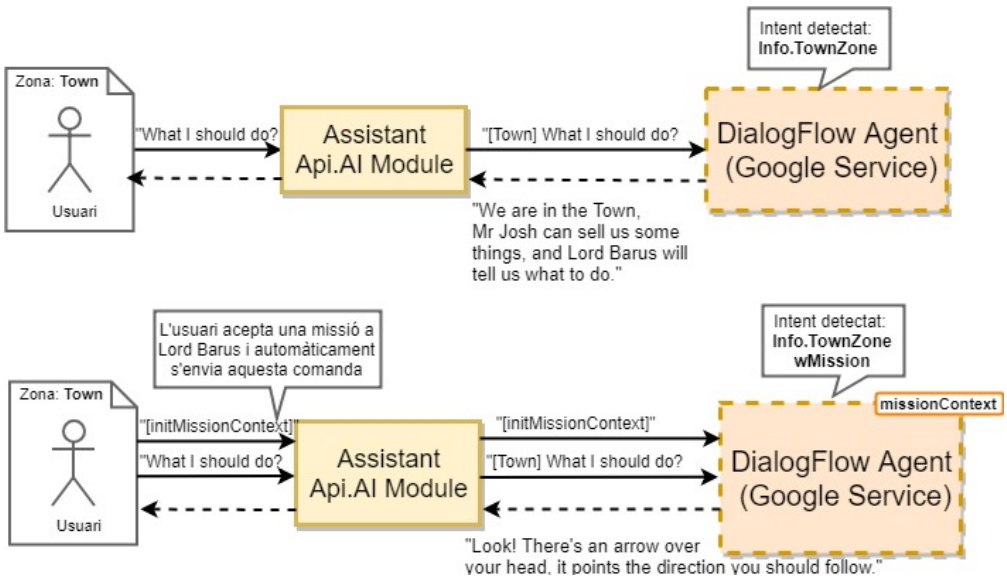

Figura 29: Exemple i procés de comunicació entre el jugador i l'agent.

[10]Town és el escenari principal, on no s'hi desenvolupa cap missió.

[11]La fletxa verda és un element del joc Fracsland situada a la part superior de la pantalla, que indica la direcció en la que es troba el punt, on ha d'anar el jugador, de la missió actual. En la versió VR es troba sobre el cap del jugador.

6 Implementació

En aquesta secció es tracta la implementació del projecte, passant per les diverses etapes, amb la finalitat d'aconseguir la màxima modularitat possible en el codi. Al llarg de la implementació d'aquest projecte s'han desenvolupat tres versions consecutives que han anat incorporant funcionalitats de manera incremental. La primera és una interfície 2D per una comunicació directa (tant per veu com per text) amb l'agent de DialogFlow, incorpora alguns elements per a poder inspeccionar les respostes rebudes. A la segona versió es parteix del projecte Fracsland i en una escena a part es copia el personatge Vendor, en el que se li incorporen les funcionalitats d'agent conversacional, mantenint la interfície en 2D per a comunicar-se amb ell. La tercera consisteix a afegir, a la segona versió, a l'agent Assistant i implementar el seu comportament. Finalment, la quarta i última versió del projecte consisteix a integrar, a la versió VR de Fracsland, els dos agents conversacionals.

6.1 Versió 1. Comunicació amb el agent, interfície 2D

La primera versió del projecte és una escena de Unity amb una interfície 2D amb diversos botons per a poder interaccionar amb l'agent de DialogFlow i veure les seves respostes. La informació de la resposta que es mostra en pantalla és el paràmetre 'speech', destinat a contenir la resposta dirigida estrictament a l'usuari (text, en llenguatge natural), la 'action' o acció de el 'intent' actual, així com un desglossament dels paràmetres del context en cas d'haver-hi. Consta també d'una senzilla màquina d'estats en el codi, i de forma visual a la pantalla, que emula els casos que al futur venedor haurem de tenir en compte. A l'esquerra de la pantalla hi ha dos camps de text del sistema de reconeixement de veu, un va mostrant les hipòtesis internes del sistema en el procés de detectar la frase pronunciada i l'altre el resultat final de la detecció. A la dreta de la pantalla es troba un quadre de text que manté un registre de la conversació que s'està duent a terme entre l'usuari i l'agent.

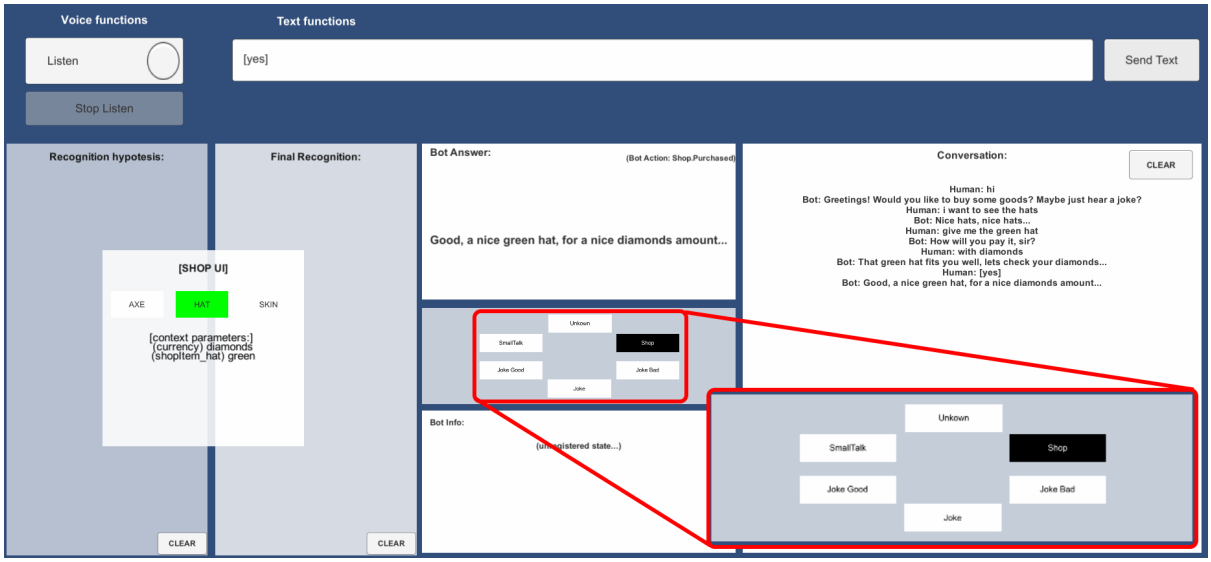

Figura 30: Distribució de la interfície per comunicar amb el agent de DialogFlow

A la Figura 30, es pot apreciar un cas pràctic d'ús, l'usuari ha demanat de comprar un barret de color verd, es pot observar com l'estat actual és el referent a 'Shop' i s'ha

activat un menú per simular, de forma molt senzilla, el comportament de la botiga, on està marcada la categoria de 'Hats' i mostra a paràmetres del context '(hat) green', demostrant que es capturen de forma correcta els valors introduïts per l'usuari. L'acomiadament conté una referència a l'objecte comprat.

Els elements que intervenen en aquesta versió són els representats anteriorment de forma conceptual a la Figura 14 i a la Figura 18 (com a diagrama de classes), a excepció del component referent al script de l'Agent (Vendor en aquest cas), ja que en aquesta escena no disposem de cap bot a la pantalla, únicament la UI de Unity per enviar i respondre.

Com es pot veure a la Figura 31, **DictationScript** necessita una referència a ApiAiModule, ja que des d'allí es comunica amb DialogFlow.

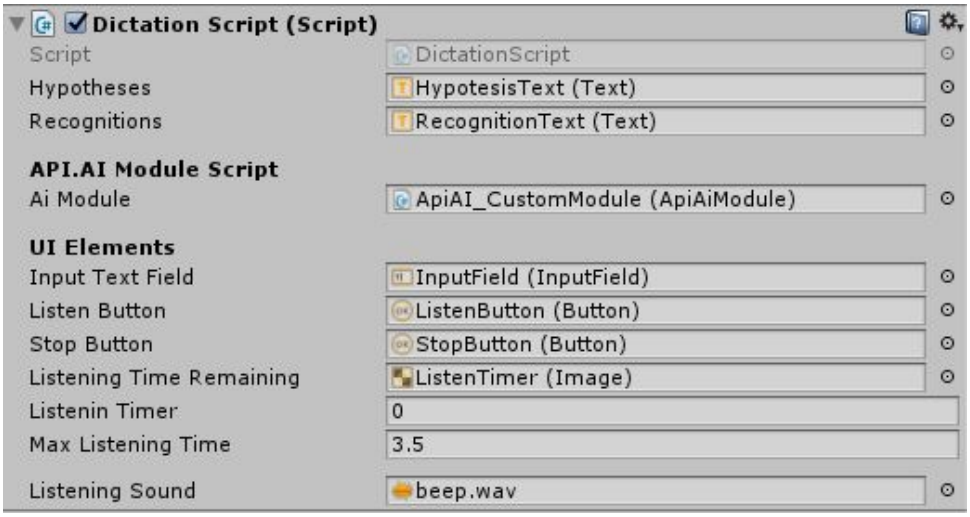

Figura 31: Paràmetres de DictationScript a l'escena.

Les altres referències que es poden observar a la imatge són secundàries i no afecten la funcionalitat final. Per exemple, les referències als botons no són per a res més que per a fer-los interactius o enfosquir-los en el moment que no són actuables (no es pot pressionar Stop si no està actualment escoltant). Les referències als camps de text d'hipòtesis i reconeixements estan pel fet que el script escriu en ells el procés que va efectuant.

Així doncs, com es pot observar a la Figura 32, la crida a activar l'escolta del script de reconeixement de veu es duu a terme a través del event 'OnClick' de la classe 'Button' de Unity, disponible des de l'Inspector, a l'editor [12].

[12]Unity disposa, per a les classes referents a la UI, d'un sistema d'assignació de funcions a esdeveniments o 'events' predefinits de forma molt senzilla. El programador només ha de fer un 'drag & drop' de l'objecte que conté el script on volem cridar a la funció determinada per al 'event' desitjat.

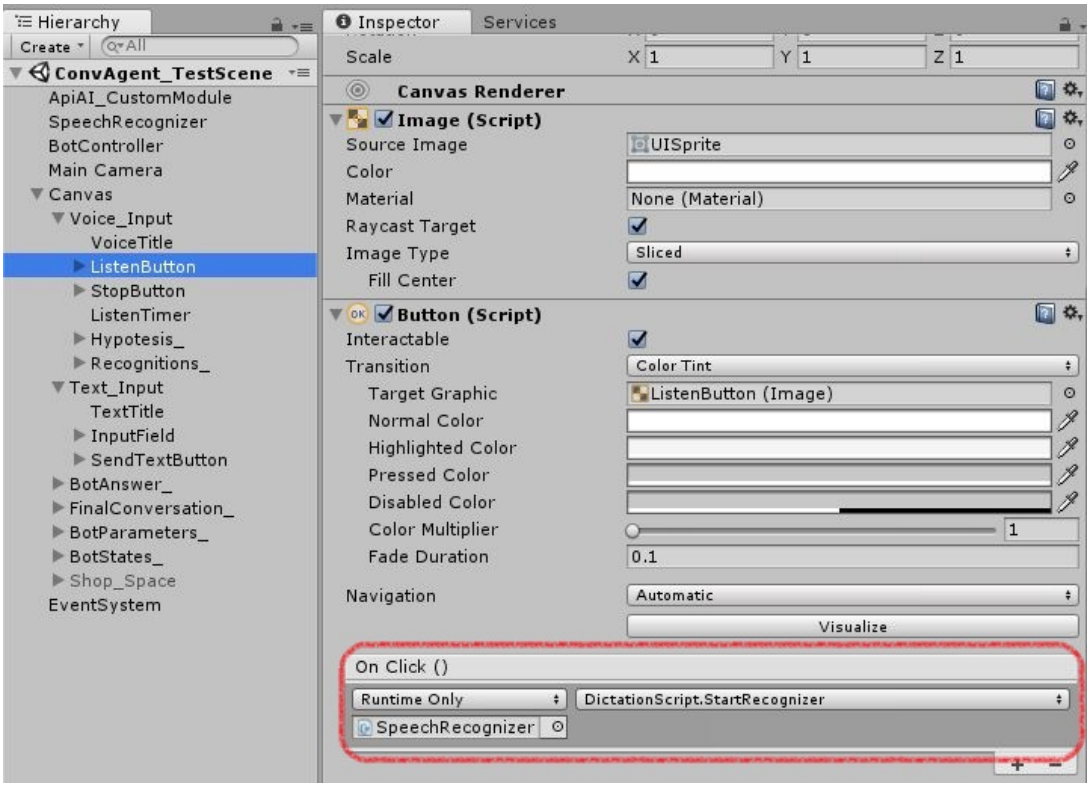

Figura 32: S'utilitza el 'event' de Unity per accionar el reconeixedor de veu.

6.2 Versió 2. Agent implementat al 'vendor' de Fracsland

Per a extendre la versió anterior, incloent-hi al personatge Vendor de Fracsland, es va decidir de crear un 'UnityPackage' [13] amb totes les eines emprades per a importar-lo al projecte de Fracsland en un sol pas.

Per evitar complicacions, es va decidir crear una escena a part, dins el projecte de Fracsland, i traslladar (importar) al NPC venedor a aquesta nova escena, on es va crear una nova versió del script '`Vendor.cs`', copiat de l'original, però amb el nom canviat a '`Vendor_DFlow.cs`' per evitar incompatibilitats amb la reutilització del projecte.

Per a aquesta implementació es van reutilitzar certs elements de la versió amb la interfície 2D com es pot veure a la Figura 33.

[13] UnityPackage: Format per a exportar/importar 'assets' i qualsevol tipus de fitxer entre projectes de Unity en un sol fitxer.

Figura 33: Distribució de la interfície integrada amb el NPC Vendor

En aquest script es van programar les accions que el personatge havia de dur a terme depenent de la 'action' en la resposta de l'agent. Per tant, es va haver de copiar tot el referent a la botiga del joc i fer la versió pròpia dels seus scripts, un per la botiga en general, anomenat `Shop.cs` i un per als ítems de la botiga, anomenat `ShopItem.cs`, seguint la mateixa nomenclatura que en el cas del Vendor, afegint '`_DFlow`' al final del nom, per a poder tenir la funcionalitat de la botiga disponible i programar la selecció de categoria i el fet de mostrar algun element en concret.

A la Figura 34 es pot veure el resultat d'una petició per part de l'usuari demanant de veure els barrets disponibles; després l'usuari demana pel barret que vol en concret així com amb el tipus de moneda que vol pagar-lo. A la Figura 35 es pot apreciar el pas final de la compra, on l'agent pregunta per el tipus de moneda amb la que farà el pagament.

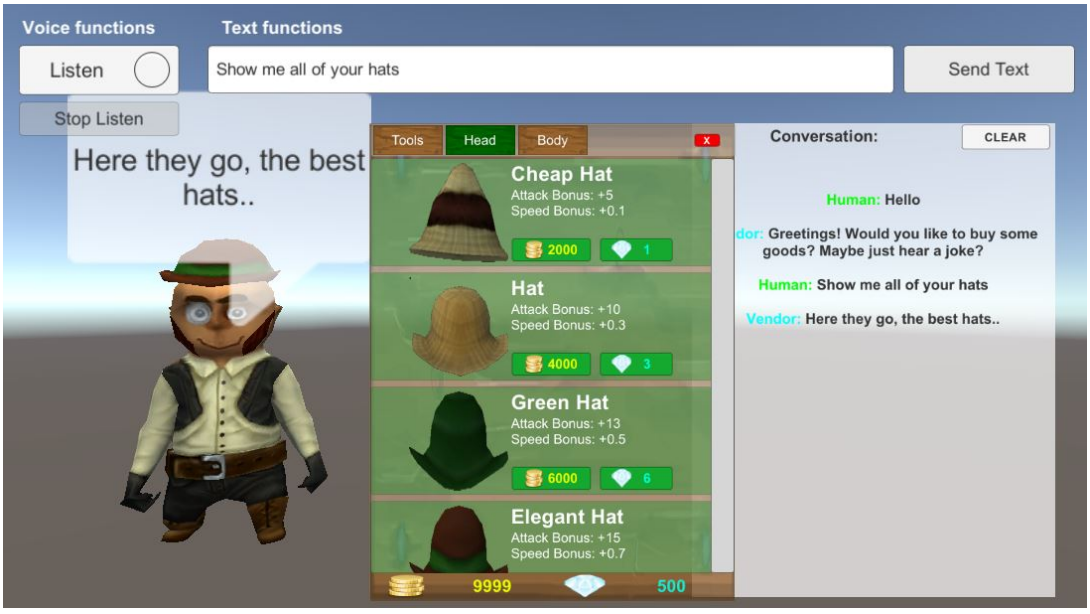

Figura 34: El venedor mostrant tota la categoria de barrets per petició de l'usuari.

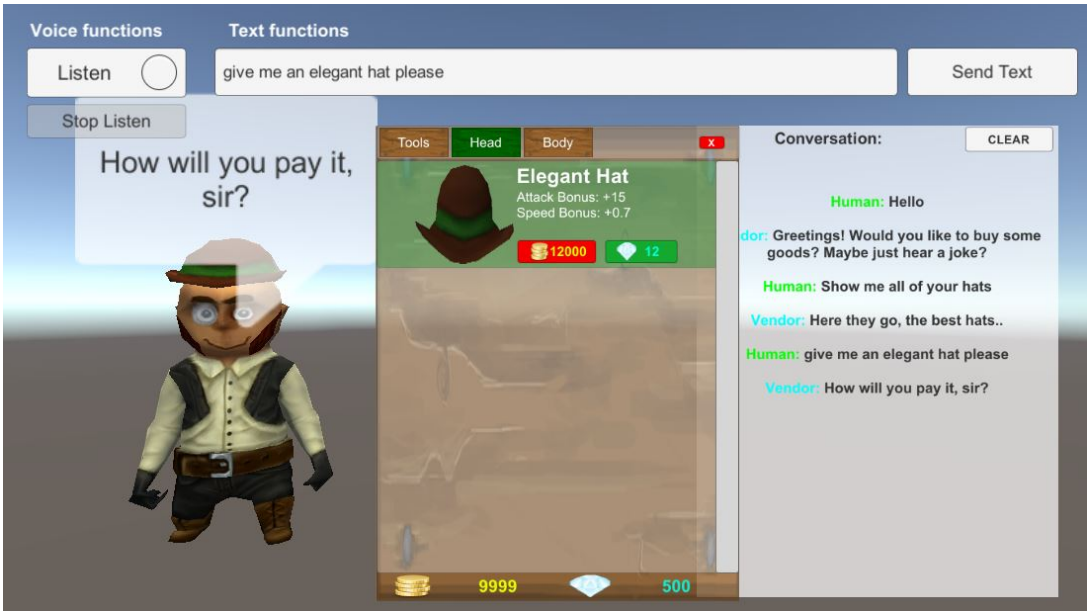

Figura 35: El venedor mostrant un únic objecte, sol·licitat per l'usuari.

Un cop vist el procés de compra i el resultat, ara s'explicarà el procés que es duu a terme dins de la funció `SetAction(AIResponse)` del venedor per discernir entre les accions que el personatge durà a terme. Aquesta funció rep per paràmetre una estructura anomenada `AIResponse` que conté la informació estructurada del JSON resposta de DialogFlow, a la Figura 36 es pot veure els camps que ens interessen d'aquesta estructura, com per exemple el camp 'action' el qual conté el tag que s'utilitza per a determinar que ha de fer l'agent dins el joc.

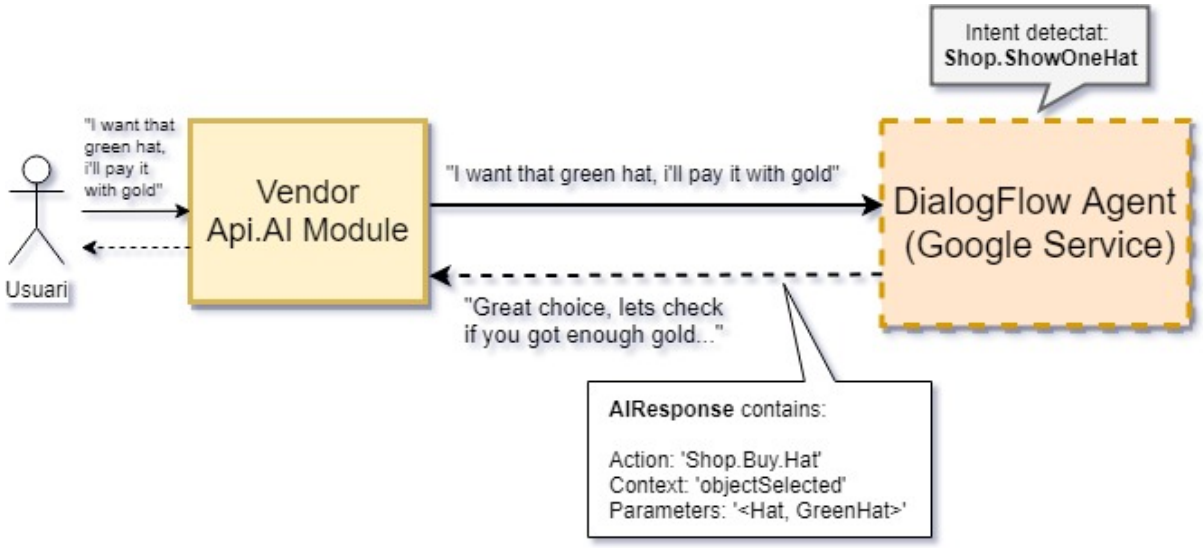

Figura 36: Els camps de AIResponse que s'utilitzen al projecte.

Els tags s'utilitzen de forma additiva, per il·lustrar-ho, un exemple seria "Shop.Show.Hats" o "Social.Jokes.BadJoke". El primer a considerar és que hi ha dos possibles inicis per als tags, 'Shop' relacionat amb la compra i 'Social' per a les altres funcions, per tant el primer punt que s'ha de tenir en compte és si es tracta d'una acció relacionada amb les funcionalitats de la botiga o forma part de les funcionalitats socials, que en aquest últim

cas no hauria de fer gaire més que executar una animació particular per als casos que es pugui detectar que són positius o negatius. A la Figura 37 es pot veure, en un diagrama de flux, els casos que es consideren des de `SetAction(AIResponse)`.

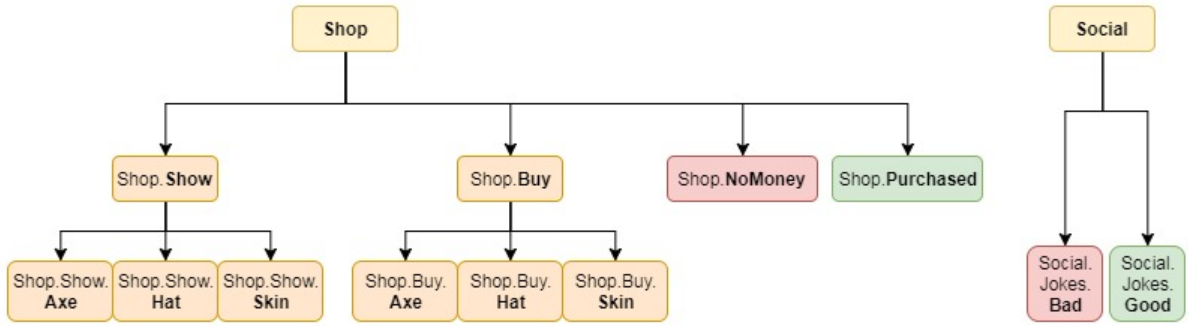

Figura 37: Diagrama que mostra els possibles tags i l'ordre en que l'agent de Unity els llegeix per destriar l'acció a fer.

Un cop vist el procedir, a continuació es desglossen els possibles tags i com l'agent gestiona cadascun d'ells.

- Conté el tag '**Shop**':

  Si es tracta d'alguna funció de la botiga, el següent pas es veure si es tracta d'una petició per a veure els objectes disponibles o es tracta ja d'una compra:

  - Conté '**Show**':

    Show s'utilitza per a mostrar les categories (decisió de disseny, veure 5.2.2), per tant disposem de tres tags més a analitzar:

    * **Axe**.
    * **Hat**.
    * **Skin**.

    A l'estar dins el cas de mostrar, ho considerem com mostrar una categoria en la seva totalitat, per tant discernim entre els tres possibles casos i simplement es crida a la funció de `Shop.cs` anomenada '**RenderCategory(**category, 0**)**', sent 0 el ID del objecte a mostrar (explicat a la secció 5.1.3), ja que per com està feta la funció, si el ID és 0, es mostra la categoria al complet.

  - Conté '**Buy**':

    Al contenir Buy, per decisió de disseny es considera l'estat de precompra, és a dir, l'usuari ha demanat per un objecte en concret, encara que sigui només per a veure'l, el venedor li oferirà comprar-lo en arribar a aquest punt. Consta, de nou, d'estar seguit dels tres possibles següents tags:

    * **Axe**.
    * **Hat**.
    * **Skin**.

    Degut a que només rebrem el tag Buy si es tracta de una petició concreta per un sol objecte, acte seguit de trobar a quina categoria pertany (ens ho diu el propi tag, que segueix a Buy, un exemple seria 'Shop.Buy.Hat'), hem d'accedir

al context que ens transmet la resposta i extreure'n l'objecte que ens demana l'usuari i que DialogFlow ha identificat.

Està bloquejada l'opció, des de la plataforma de DialogFlow, d'obtenir una llista de paràmetres, així que podem assegurar que només n'hi haurà un. Els paràmetres venen en format diccionari, en parelles de valors de clau-valor; això és a causa de la naturalesa de les entitats de DialogFlow [explicat a 1.1 Conceptes Relacionats] en què hi ha com a clau l'entitat i com a valor la sub-categoria de l'entitat (un exemple seria `<Key:Hat, Value:Green>`). Tot i així, s'efectua un bucle *for* per tots els paràmetres que puguin haver-hi, es descarten els valors buits i es crida a la funció '**GetItemID(key, value)** que ens retorna una ID per identificar l'ítem demanat en concret. Aquesta mateixa ID només té sentit per al script Shop.cs, que des de la seva funció **RenderCategory(categoria, ID)** s'utilitza per a només renderitzar l'ítem desitjat a la llista de la botiga, juntament amb la categoria d'aquest, seleccionada gràcies a la informació continguda al tag.

– Conté '**Purchased**' o '**NoMoney**': (Introduït a la secció 5.2.2)

Un cop l'usuari ha realitzat el procés de selecció d'ítem i ha especificat amb quin tipus de moneda pagarà, la lògica de Unity s'encarrega de determinar si es pot pagar o el preu és més alt que els diners del jugador, en aquest moment envia una petició (oculta a l'usuari) amb el format '[**yes**]', indicant que es pot efectuar la compra. La resposta a aquest intent conté el tag 'Shop.Purchased', és en aquest punt que el venedor, de Fracsland, sap que ha realitzat la venda i ho celebra amb una animació particular.

Si per al contrari rep un '[**no**]', la resposta de l'agent contindrà el tag 'Shop.NoMoney', en aquest cas el venedor executa una animació de decepció i informa a l'usuari de què no disposa de prou diners.

Ambdós casos, al finalitzar, retornen a la botiga, a la última categoria que havia obert l'usuari. A la Figura 38 podem veure per un costat, com en el registre de la conversació mostra el procés de la compra, així com el objecte finalment comprat. Remarcar que a la resposta de l'agent es fan referència a les opcions introduïdes per l'usuari.

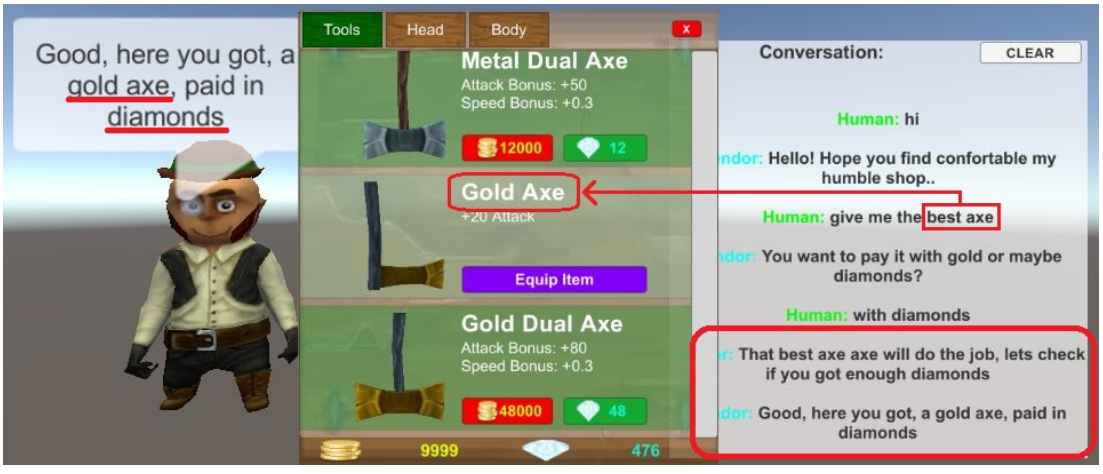

Figura 38: Estat final de la compra, un cop detectat automàticament des de la lògica de Unity, si l'usuari pot pagar l'objecte

- No conté el tag '**Shop**':

  Si per al contrari, no conté el tag 'Shop', es tracta d'una acció social. Es detecta si conté la paraula 'Bad' o 'Good' per a executar respectivament animacions d'alegria o tristor.

Per introduir un element proactiu, es va decidir d'afegir un Collider de tipus *trigger*[14], a l'escena on l'agent es situés, per a detectar quan el jugador està a prop del venedor (comentat a la secció 5.2.2), en activar-se, envia un missatge, ocult a l'usuari, a DialogFlow, que contesta saludant; d'aquesta és l'agent qui ha iniciat el diàleg. Per a aquesta funcionalitat es va crear el script '*VendorTrigger.cs*', on hi han les funcions predefinides de Unity per a la detecció d'entrades i sortides d'objectes del *trigger* en qüestió, en detectar-se alguna d'aquestes situacions, comprova per 'tag' (tag propi de Unity, no confondre amb els mencionats de DialogFlow[15]) si es tracta del jugador o no. Si es tracta del jugador, en entrar al trigger, l'agent el saludarà, de la mateixa forma en sortir, l'agent s'acomiadarà del jugador.

En última instància d'aquest pas, es va traslladar al nou venedor amb les funcions d'agent conversacional, a l'escena de FracslandVR anomenada TownVR, que és on estava el venedor original, substituint-lo i afegint la funcionalitat per a funcionar amb l'input de la plataforma OculusVR[13] i posicionar el *trigger* de la salutació proactiva.

Es va decidir que la mecànica seria la de mantenir pressionat un botó, en detectar-se el 'hold' per part de les funcions del SDK de Oculus, activar l'escolta de veu, mitjançant la funció, ja comentada, '**StartRecognizer()**' i en el moment de detectar que l'usuari deixa anar el botó, es crida a '**StopRecognizer()**' aturant d'aquesta forma l'escolta d'àudio i el reconeixement de veu, enviant mitjançant 'ApiAiModule', el text resultant al servei de Google.

6.3 Versió 3. Agent assistent afegit

En aquesta secció s'explica el desenvolupament del projecte en què l'agent assistent va ser implementat. Al disposar del codi fet per l'agent Vendor, es va partir de la seva estructura per a programar a l'agent assistent.

El model 3D de l'agent es tracta d'un mussol que es trobava als recursos utilitzats per al projecte de Fracsland, per tant no hem requerit de cap nou model o recurs extern. A més, el model porta una sèrie d'animacions predefinides.

La diferència principal radica en el fet que l'assistent no disposa de cap funcionalitat de botiga, i té la particularitat de seguir físicament al jugador allà on va. Alhora, el bot ha d'estar disponible per a rebre alguna pregunta de l'usuari, tramesa amb la veu en accionar el botó definit. Per decisions de disseny es va decidir que l'assistent se situaria en algun punt fix definit, per exemple, a l'espatlla o al cap del jugador, en el moment de contestar a l'usuari. Per al contrari, si no té res a dir, es posa a seguir a l'usuari, traçant una ruta al voltant seu. Es disposa de dos punts de posicionament per l'agent i de dues possibles rutes a seguir, pensant en la futura implementació d'aquest a la versió VR de

[14] A Unity es permet la creació d'un tipus de Collider anomenat 'trigger', que té la particularitat de que no afecta a les físiques, els objectes el poden travessar, d'aquesta manera executa les funcions pertinents (OnTriggerEnter i OnTriggerExit) per a obtenir els objectes que hi interactuen.

[15] Fins ara hem parlat de 'tags' per a explicar com s'utilitza el camp 'action' de DialogFlow. El 'tag' referit però, no és de DialogFlow, sinó que es refereix al tag que tots els GameObjects de Unity tenen. En concret el jugador té el tag de 'Player'.

Figura 39: Agent Assistant en mode de càmera de tercera persona.

Fracsland, on hi ha dos modes de càmera, amb punt de vista a primera i tercera persona, per a la primera persona l'agent es situa enfront de la càmera mentre que per a la tercera persona es situa sobre el jugador. Un exemple d'aquesta funcionalitat es pot apreciar a la Figura 39.

Per a complir aquests propòsits es va adoptar un model de màquina d'estats, i el moviment de l'agent, juntament amb les animacions, es porta a terme en funció de la variable '*currentState*', visible des de l'Inspector de Unity com es pot apreciar a la Figura 40.

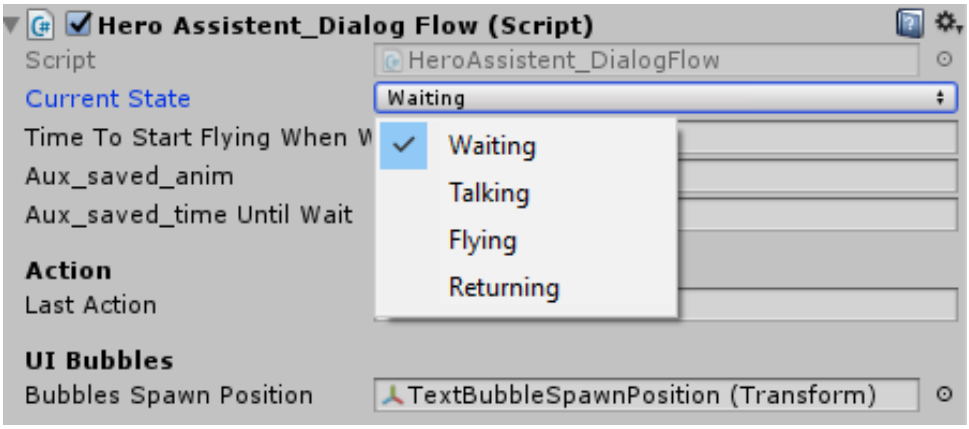

Figura 40: Variable pública *currentState* visible des de l'Inspector.

A la Figura 41 es pot veure representada la màquina d'estats d'aquest agent i a continuació s'explica el funcionament de cada un d'aquests estats:

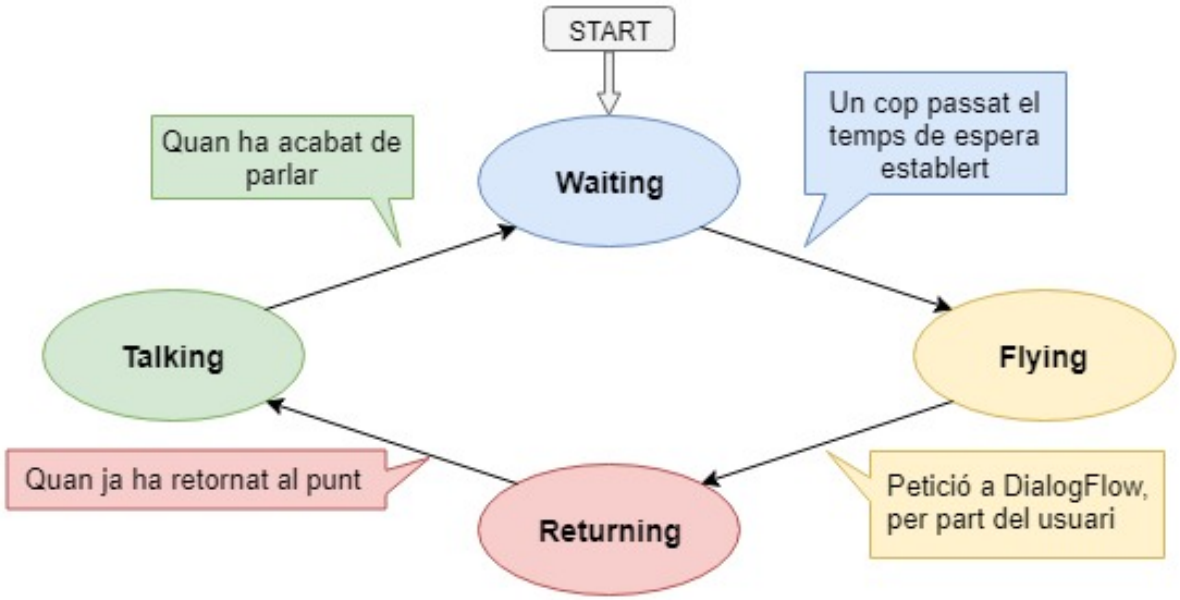

Figura 41: Diagrama que mostra la màquina d'estats de l'animació i moviment de l'agent així com les condicions de transició.

- **Waiting**:

  Quan l'agent està en estat d'espera, es manté en el punt definit. El punt ve definit per les variables públiques 'firstCam_place' i 'thirdCam_place', depenent si l'agent es troba en mode primera o tercera persona. Funcionalitat pensada amb previsió de futur, com ja s'ha comentat, per a ser afegit amb les mecàniques de FracslandVR, on hi ha mode de càmera de primera i de tercera persona.

  Existeix la variable '*timeToStartFlyingWhenWaiting*' al script de l'agent, en l'estat d'espera s'utilitza aquesta variable per a determinar quan ha passat el temps d'espera indicat per a l'agent, un cop passat el temps definit, l'agent passa automàticament a l'estat de 'Flying'.

- **Talking**:

  Aquest estat és seleccionat des de la funció '**Talk()**', funció la qual es crida cada vegada que es vol pronunciar una resposta de l'agent.

  La funció d'aquest estat és la de mantenir a l'agent al punt d'espera definit (comentat a l'estat anterior), mentre executa la pronunciació de la frase resposta i la mostra per escrit en una bombolla de text.

- **Flying**:

  A l'estat de volant s'executa l'animació de vol pertinent. El script de l'agent disposa d'una llista de punts que simbolitzen una ruta per on l'agent es mou quan no està parlant [16]. Quan entra en estat de 'Flying', se li assigna a l'agent un punt al qual ha de desplaçar-se, un cop hi arriba, se li assigna el següent, i així successivament, en forma de bucle. Per a sortir d'aquest estat, l'usuari ha de parlar amb l'agent.

[16]Els punts s'han d'assignar des de l'Inspector de Unity, ja que l'efecte buscat es que segueixi al jugador, els punts de la ruta es troben dins l'objecte 'Hero' de Fracsland, el jugador. D'aquesta forma segueix al jugador traçant una ruta al voltant seu.

- **Returning**:

  L'estat de retorn o 'Returning' està pensat perquè quan el jugador li envia una petició o frase a l'agent, aquest pari de volar, retorni al punt definit per parlar, i aleshores parli.

  Per tant és només un estat passatger entre Flying i Talking.

La utilitat de tenir els estats en el bot és el fet de poder ampliar-li la funcionalitat en un futur fàcilment, o millorar el feedback dels elements del projecte acord amb el que el bot es troba fent en un determinat moment. A la Figura 42 podem veure com actua el bot en els diferents estats.

Figura 42: D'esquerra a dreta. Agent Assistant en estat d'espera, volant, retornant i parlant.

De la mateixa manera que l'agent Vendor, l'assistent necessita la seva funció **SetAction(AIResponse)**[17] per a poder actuar en coordinació amb les frases que pronuncia com a resposta a l'usuari.

En aquest cas, al no tenir la funcionalitat de la botiga, aquesta funció és molt més senzilla. L'agent ha estat dissenyat des d'un inici per a disposar d'una sèrie de *intents* que s'activen únicament per 'comandes' que s'envien, de forma transparent per a l'usuari, des de Unity; per tant hem de detectar quan es contesta des de DialogFlow amb una resposta que no va dirigida a l'usuari. Recordar que la llista d'intents possibles per a aquest agent es troba a la secció 5.2.3.

Així doncs, els casos que causen accions diferents en l'agent assistent són:

Si es detecta el tag 'Command' dins la 'action' de la resposta de DialogFlow, s'ignora, ja que l'agent no ha de parlar amb el jugador (es tracten de missatges de l'estil "*Mission Context created*"destinats a ser mostrats en mode Debug). Si es detecta el tag 'Bad', de la mateixa forma, l'agent executa una animació en particular com a feedback, mentre fa la crida per mostrar la resposta.

Tots els altres casos són tractats de la mateixa forma, però es podria afegir tantes animacions per a casos particulars com el desenvolupador desitgés. En tots els casos, a excepció de detectar-se el tag 'Command', l'agent ha de mostrar (i pronunciar de veu) la resposta que està rebent de DialogFlow, en el cas de l'assistent, això es fa de la següent manera:

Per a que l'agent parli la resposta es crida a la funció **ReturnMainPositionToTalk(speech, animation, timeAnim)**, on els paràmetres són, per ordre, el text per mostrar (i pronunciar), l'animació que s'executarà juntament amb la pronunciació del

[17]Els continguts de AIResponse han estat comentats a la secció 6.2, i es poden veure a la Figura 36 els camps d'interès.

text i el temps d'aquesta. Aquesta funció guarda els valors introduïts i estableix en l'agent l'estat de 'Returning', que bloqueja la parla de la resposta fins que l'agent està en posició.

Quan l'agent ha arribat al punt designat per a parlar, executa l'animació de frenar el vol, se situa, i crida a la funció **Talk()**, que utilitzarà els valors guardats anteriorment per a executar l'animació en concret, mostrar el text de resposta en una bombolla de diàleg i alhora utilitzar el component 'WindowsVoice' per a pronunciar-lo de veu.

A la secció 5.2.3, referent al disseny del agent Assistant, s'ha vist un exemple de com seria la comunicació i l'ús dels tags, concretament a la Figura 29

6.4 Versió final. Vendor i Assistant al joc FracslandVR

Per a la implementació en FracslandVR, el primer pas és el d'afegir tots els scripts amb les funcionalitats dels agents necessàries a l'escena. És a dir, `DictationScript`, els Moduls dels dos agents de Google (`ApiAiModule_Vendor` i `ApiAiModule_Assistant`) i el script `WindowsVoice`.

Un cop tenim els components que ens donen les funcionalitats bàsiques de veu-text, text-veu i de comunicació amb DialogFlow, només falta afegir als personatges agents.

Ja que tenim a DictationScript a l'espera d'un input per a començar a reconèixer la veu de l'usuari, i no es parlarà en cap situació amb els dos agents alhora, s'ha introduït en aquest script un *enum* amb els bots disponibles, en el nostre cas 'Vendor' i 'Assist', i una variable 'currentAgent'.

Als scripts propis de cada agent (no als *AiModule*, sinó als que porten la lògica dels personatges) se'ls hi ha afegit un mètode anomenat '*InputUpdate()*' que conté les comprovacions pertinents per veure si l'usuari està pressionant el botó definit al controlador del Oculus.

Com que DictationScript no té comunicació directa amb els scripts dels agents (veure secció 5.1.2), les funcions es criden a través de la referencia al script de cada agent continguda dins dels *AiModule*'s de cada un dels agents.

Aleshores el funcionament és el següent, DictationScript, al seu mètode *Update()* comprova quin agent és actiu, i al que li pertoqui li invoca el seu mètode *InputUpdate()* a través del ApiAiModule del agent en qüestió, com es mostra a la Figura 43.

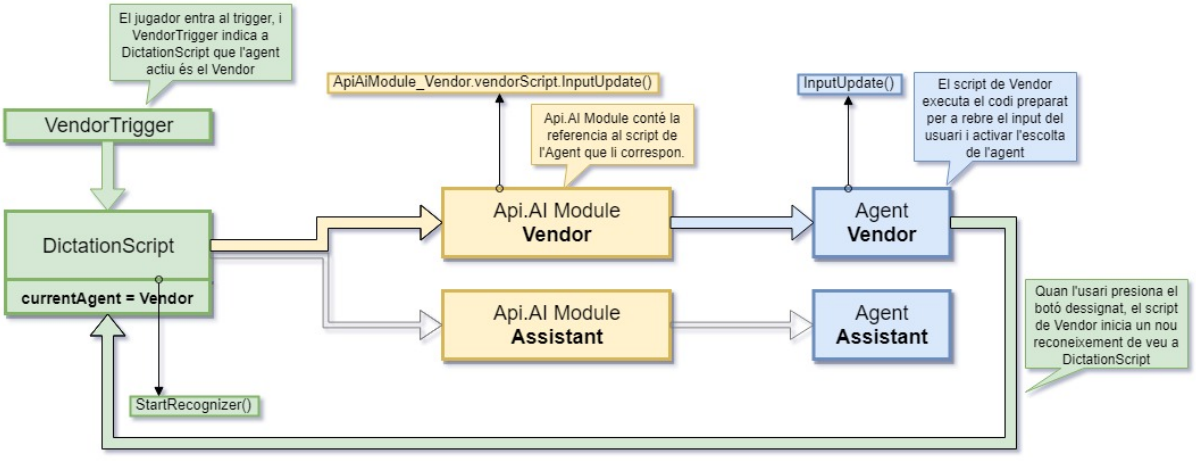

Figura 43: DictationScript selecciona quin agent està actiu, i activa el seu mètode per a detectar l'input de l'usuari. Exemple d'activació de Vendor.

La selecció de l'agent actiu a DictationScript es duu a terme des del script '`VendorTrigger.cs`', on se li ha afegit una referència a DictationScript, ja que ambdós scripts estan a objectes que no muten a l'escena. Es canvia l'agent actiu de la següent forma, si el jugador està accionant el *trigger*, que es mostra a la Figura 44, l'agent actiu serà el Vendor, en el cas contrari, l'Assistent.

Figura 44: Àrea coberta pel trigger de Vendor

Ja que podia ser desitjable, es va implementar una versió del text que portava el registre de la conversació a les versions anteriors, a la versió VR. El quadre de text mostra el diàleg entre el jugador i els agents, mostrant etiquetes de colors per a facilitar a l'usuari la comprensió. Es va decidir que aniria situat al dors de la mà dreta, amb un botó per mostrar-lo o desactivar-lo quan l'usuari ho desitgi, a part de comptar amb un botó 'Clear' per a netejar el quadre amb la conversa, com es mostra a la Figura 45. El text amb els diàlegs es guarda entre escenes utilitzant un script original de Fracsland que no s'elimina al pas entre escenes, i es recupera d'allí en carregar una escena nova.

Figura 45: Log de la conversa entre l'usuari i els dos agents (a l'esquerra) i botó per a amagar/mostrar aquest log (a la dreta).

Un altre punt important a tractar és el fet que el jugador s'instància per codi una sola vegada, per l'altra banda, l'agent Assistant es troba present d'inici a totes les escenes, i necessita accés al jugador, ja que l'agent ha de seguir una sèrie de punts que tracen una ruta per aconseguir l'efecte de què es mogui al voltant del jugador.

Com que el jugador es mou lliurement, es va decidir aprofitar la possibilitat de crear jerarquies de *GameObjects* de Unity[18] i fer la llista de punts 'fills' del jugador com es pot veure a la Figura 46. Es disposa de dos llistes de punts, una per al mode de càmera en primera persona, i l'altre per al mode de tercera persona.

Figura 46: A la esquerra, els punts que l'Assistant utilitzarà com a ruta, a la dreta, la jerarquia dins el jugador on es poden veure els punts de la ruta

El problema aleshores és que l'agent Assistant necessita una referència a aquests punts, per tant es va crear un script anomenat '`AssistBotInitializerHelper.cs`' (Figura 47), situat a l'objecte del jugador, amb totes les referències fixades [19], que a l'iniciar-se l'escena, espera uns mil·lisegons, per a donar temps al jugador d'instanciar-se al complet, busca a l'agent (per 'tag' de Unity) i li assigna les referències que necessita.

[18] A Unity els objectes s'organitzen per jerarquia, es poden assignar objectes 'fills' d'altres, amb això s'aconsegueix que aquests objectes es moguin amb l'objecte 'pare' sense codi extra.

[19] A Unity es poden guardar objectes en format 'prefab'. Un prefab és un GameObject, juntament amb tota la jerarquia d'objectes fills que disposi, així com tots els components que el formen. Si algun dels components es tracta d'un script, es guarda també els valors de les variables en el moment de crear-se el prefab.

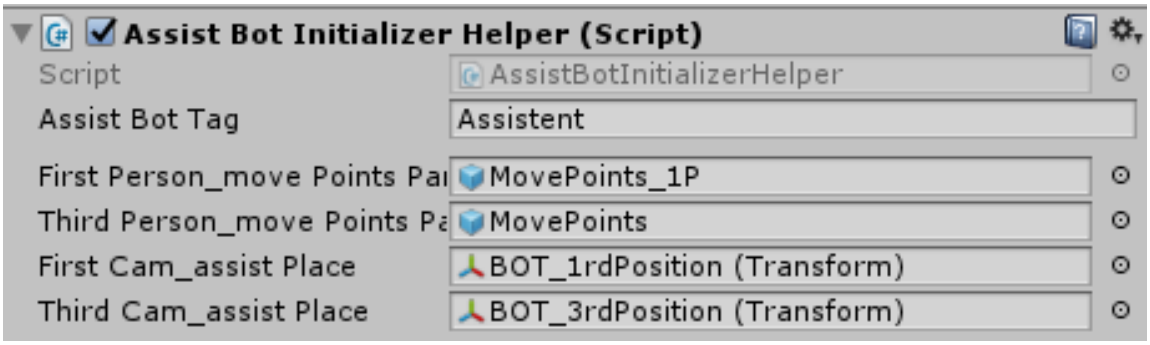

Figura 47: Script AssistBotInitializerHelper amb les referències assignades.

7 Conclusions

Tal com es va esmentar a la secció d'objectius, l'objectiu d'aquest projecte és el d'aprendre el funcionament dels agents conversacionals (concretament, de DialogFlow), que comprèn saber crear-los i saber utilitzar-los, així com la seva integració al joc Fracsland, fet amb el motor gràfic Unity, i a Fracsland VR, creat en paral·lel a aquest projecte, partint del joc original Fracsland.

A continuació es comenten els objectius plantejats i el seu nivell d'assoliment.

1. S'han investigat i après els serveis de DialogFlow satisfactòriament, així com els recursos que ofereix la plataforma, gràcies a la documentació oficial de DialogFlow.

2. S'han creat agents conversacionals utilitzant els coneixements adquirits. Concretament s'han desenvolupat dos agents (vendor i Assistant) amb dos rols diferents dins el joc de Fracsland.

3. S'han trobat els recursos necessaris per a poder utilitzar la plataforma de DialogFlow des de l'editor Unity. En les diferents fases del projecte s'han creat tres escenes de Unity per a comunicar-se amb el agent, una amb una interfície 2D bàsica, i les altres dos per a comunicar-se amb els dos agents creats (vendor i Assistant).

4. S'han trobat solucions efectives per al reconeixement de veu des d'un projecte de Unity, utilitzant els recursos que ofereix el sistema de veu de Windows.

5. S'ha trobat una solució per a que la resposta de l'agent sigui pronunciada de veu, un cop més, a través de les eines de Windows.

6. S'ha integrat un agent conversacional al personatge Vendor del joc original Fracsland. S'ha dissenyat de nou el funcionament de la botiga (element ja present al joc) per a utilitzar-la de veu, fent servir els paràmetres obtinguts de DialogFlow per a gestionar-ho.

7. S'ha creat un nou personatge al joc i s'ha integrat un agent conversacional en ell, l'hem anomenat Assistant. S'ha dissenyat la seva funcionalitat d'assistent al jugador i el seu comportament dins el joc.

8. Finalment, s'han integrat satisfactòriament els dos agents, en els seus personatges respectivament, dins el projecte de Fracsland VR, amb el resultat de tenir un agent conversacional a qui poder comprar objectes i un altre que aconsella al jugador, ambdós es comuniquen amb l'usuari per veu.

Així doncs es pot afirmar que s'han assolit els objectius proposats. La creació d'agents conversacionals ja no és un misteri, i el seu ús dins de l'editor de Unity és factible, com s'ha pogut comprovar. El desenvolupament s'ha dut a terme, en la mesura del possible, de forma modular pensant en la reutilització en un futur, ja sigui per a crear de zero un projecte amb agents conversacionals (utilitzant la primera part del projecte) o ja sigui per extendre més enllà les funcionalitats del joc Fracsland.

Com es comentava, tot i haver assolit els objectius plantejats, el projecte és extensible, i àmpliament. A la següent secció es tractaran els possibles punts a millorar o ampliar en funcionalitat.

7.1 Treballs futurs i idees d'expansió

En aquesta secció es tracten algunes possibles extensions i treballs futurs a realitzar sobre el projecte. No es tenen en compte les possibles extensions per al joc de Fracsland, ja que poden ser múltiples i no ha estat el focus principal del projecte.

1. **Lord Barus amb un agent de DialogFlow**

   Una idea per a una possible extensió que involucrés tan agents conversacionals, com a algun personatge a Fracsland, seria la d'afegir a Lord Barus la possibilitat de comunicar-se amb l'usuari de veu i implementar les seves funcionalitats a través de la tecnologia d'un agent conversacional.

2. **Creació d'un component Agent Controller**

   Ja que les competències entre els dos agents realitzats en aquest projecte eren molt distants, no ha calgut gestionar de forma complexa el fet de tenir diversos agents dins d'una mateixa escena. La idea de millora seria el fet de crear un script AgentController que tingués control sobre l'agent actiu en el moment, per determinar cap a on dirigir l'input de l'usuari. Amb aquest component es podrien fer moltes més ampliacions, com per exemple estudiar la possibilitat de fer que dos agents parlessin entre ells en algun moment del joc.

3. **Suport per a diferents idiomes**

   La possibilitat de poder utilitzar als agents conversacionals creats en diferents idiomes, no només en anglès. Aquesta funcionalitat s'hauria de dur a terme des de la plataforma de DialogFlow, des d'on es deixa seleccionar idiomes addicionals, però requereix escriure un a un tots els *intents* existents.

4. **Funcions pro-actives a l'agent Assistant**

   Afegir funcionalitats proactives a l'assistent del jugador, tals com: Dir algun comentari concret en diferents moments del joc, o avisar quan el jugador està sent atacat per algun enemic, ja que en algunes escenes hi ha animals que ataquen, i concretament a la versió VR del joc no es pot assegurar que l'usuari tingui al jugador dins la pantalla en algun moment, ja que la càmera va lligada al moviment del cap del usuari, i no hi ha cap restricció per mirar en una direcció on no està el personatge jugador, per tant, un avís podria ser molt útil.

5. **Millorar animacions de l'agent Assistant**

   L'agent conté un component de Unity AnimatorController per a poder gestionar màquines d'estats d'animacions diverses. Actualment l'agent utilitza una llista de clips d'animacions i executa la necessària en el moment que es demana. La idea de millora seria el fet de gestionar les animacions des de l'Animator, aconseguint així un efecte més polit en les transicions entre diverses animacions o moviments.

8 Referències

[1] Muriel Ordoñez, Cristian: *Fracsland, joc seriós per aprendre fraccions*, 7-8, Barcelona, Espanya: Universitat de Barcelona, 2014.

[2] Google: *DialogFlow Documentation*, recuperat de https://dialogflow.com/docs/ (Juny 2018)

[3] SearchCIO: *IBM Watson CTO: A range of conversational technologies can coexist.*, recuperat de https://searchcio.techtarget.com/feature/IBM-Watson-CTO-A-range-of-conversational-technologies-can-coexist (Maig 2018)

[4] Wikipedia: *Microsoft Cortana*, recuperat de https://es.wikipedia.org/wiki/Microsoft_Cortana (Juny 2018)

[5] Wikipedia: *Siri*, recuperat de https://es.wikipedia.org/wiki/Siri (Juny 2018)

[6] Wikipedia: *Amazon Alexa*, recuperat de https://es.wikipedia.org/wiki/Amazon_Alexa (Juny 2018)

[7] Wikipedia: *Asistente de Google*, recuperat de https://es.wikipedia.org/wiki/Asistente_de_Google (Juny 2018)

[8] MeriStation: *Podrás hablar con tu Espectro de Destiny 2 gracias a Alexa*, recuperat de http://meristation.as.com/noticias/podras-hablar-con-tu-espectro-de-destiny-2-gracias-a-alexa/2245904 (Juny 2018)

[9] FayerWayer: *Sam es el asistente virtual para videojuegos de Ubisoft*, recuperat de https://www.fayerwayer.com/2018/01/sam-ubisoft-asistente-virtual/ (Juny 2018)

[10] The Verge: *In these games starring AI bots, chatting is the new shooting*, recuperat de https://www.theverge.com/2016/9/14/12905356/komrad-evento-chatbot-games-preview-conversation-ai-personality (Juny 2018)

[11] GitHub: *Unity library for Dialogflow*, Autor: xVir (usuari de GitHub de Danil Skachkov, membre del equip de DialogFlow) recuperat de https://github.com/dialogflow/dialogflow-unity-client (Març 2018)

[12] ChadWeisshaar: *Microsoft Windows Text-to-Speech for Unity*, Autor: Chad Weisshaar recuperat de http://www.chadweisshaar.com/blog/2015/07/02/microsoft-speech-for-unity/ (Abril 2018)

[13] Prieto, Astor: *Estudi i Disseny d'Interaccions d'un Joc Seriós en Realitat Virtual*, Barcelona, Espanya: Universitat de Barcelona, 2018.

Apèndix A: Manual tècnic

En aquest apèndix es tractarà tot el necessari a saber per a un desenvolupador que vulgui utilitzar aquest projecte o extendre'l.

A.1 Instal·lació: Requeriments mínims i passos a seguir

A continuació es detallen els requeriments mínims i les eines o recursos necessaris per a poder executar aquest projecte. Per el que fa als requeriments de hardware mínims per a poder utilitzar aquest projecte tenim, per una banda els originals de Fracsland [1] (si volem executar els agents a la versió original) o els necessaris per a executar Fracsland VR [13] (en cas de voler utilitzar la versió final en VR), i per l'altra banda, els que s'afegeixen per al ús dels agents conversacionals, els quals són:

- Connexió a internet. Els agents necessiten accés a la xarxa per comunicar-se amb els serveis de DialogFlow.
- Micròfon. Si es desitja comunicar-se de veu amb els agents, serà necessari disposar de un micròfon.

Referent als recursos i programari que necessitarem tenir:

- **Editor Unity**. Aquest projecte ha estat desenvolupat amb la versió 2017.3.1f1, per tant no es recomanable utilitzar-lo o editar-lo en una versió anterior a aquesta. Per a documentació sobre com instal·lar el editor Unity veure:

  `https://docs.unity3d.com/Manual/InstallingUnity.html`

- **Plataforma DialogFlow**. Per a poder utilitzar un agent conversacional de DialogFlow necessitarem un compte a la plataforma i com a mínim un agent creat. Per a aquest projecte s'ha creat un compte amb tots els agents que s'utilitzen. Les credencials per entrar-hi són:

  - **Nom**: `conv.agent.tfg@gmail.com`
  - **Password**: `dialogflow1`

  L'enllaç a la plataforma és el següent: `https://console.dialogflow.com/`

- **DictationScript**. El component que s'encarrega de convertir la veu a text. És present als fitxers del projecte. Es pot obtenir de:

  `https://docs.unity3d.com/ScriptReference/Windows.Speech.DictationRecognizer.html`

- **API de DialogFlow per a Unity**. Conté tots els scripts necessaris per a poder implementar un agent de DialogFlow a Unity. És present als fitxers del projecte. Es pot obtenir de:

  `https://github.com/dialogflow/dialogflow-unity-client`

- **WindowsVoice Plugin**. És el plugin que permet l'enviament de cadenes de text al sistema de veu de Windows per a que reprodueixi de forma parlada el text enviat. És també present als fitxers del projecte. Es pot obtenir de:

  `http://www.chadweisshaar.com/blog/2015/07/02/microsoft-speech-for-unity/`

A.2. Manual del desenvolupador

Abans de res, cal mencionar que han un seguit d'elements comuns a totes les versions, com ja s'ha explicat al llarg de la memòria.

1. ApiAiModule:

   Per al funcionament de qualsevol agent dins de l'entorn Unity és necessari tenir un `ApiAiModule` per a cada un dels agents en ús, aquest script comunica amb DialogFlow, per tant es al script on introduirem el **token** de connexió que DialogFlow ens dona, a la secció de configuració del agent, dins la seva web, com podem veure a la Figura 48.

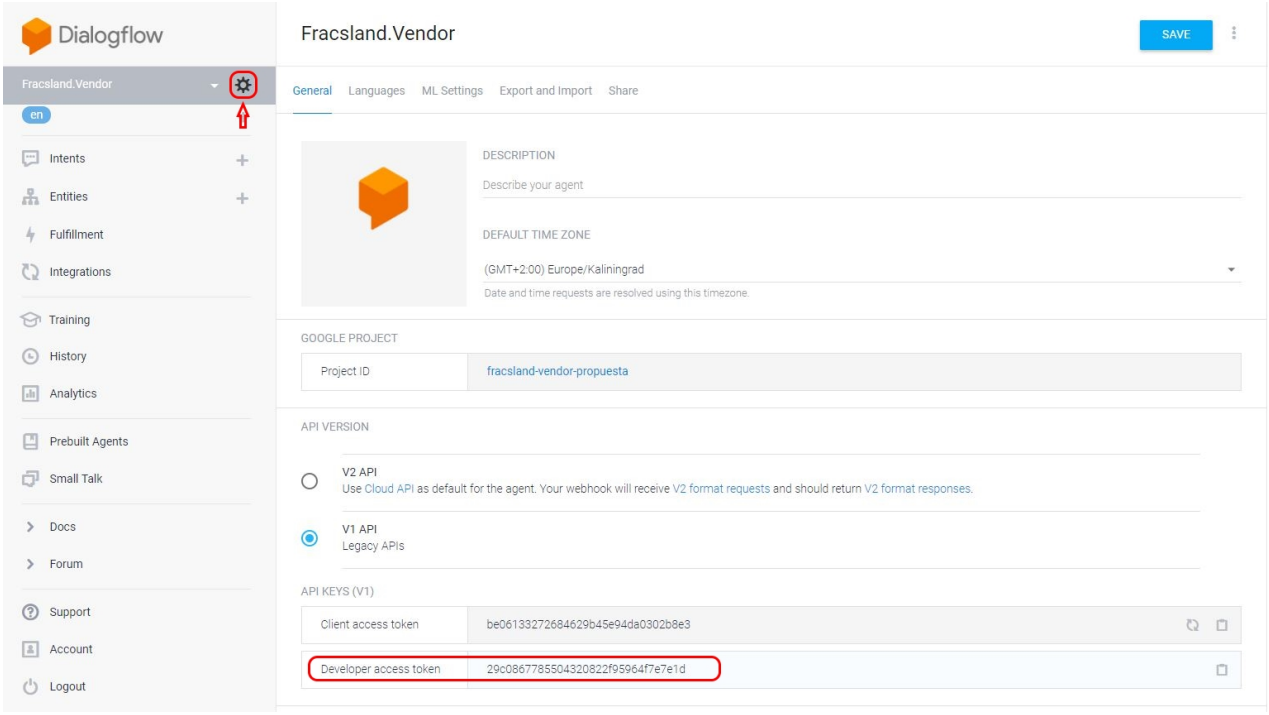

Figura 48: Localització del token de connexió, al pàgina de configuració de DialogFlow

   L'ús d'aquest script és molt senzill, disposa d'una funció SendText(string) per a enviar una petició a DialogFlow, la mateixa funció captura la resposta i la envia al script del agent dins del joc. Per tant és necessari modificar ApiAiModule per a adaptar-lo al script que el desenvolupador hagi realitzat per al seu agent. En el cas d'aquest projecte, els dos ApiAiModule (respectius a cada agent) contenen la referència al script del agent que tenen associat.

2. Agent Script:

   Es necessari de tenir un script al personatge que capturi les respostes de DialogFlow per a poder dur a terme accions concretes a l'aplicació.

   En cas de tractar-se d'una situació en que no tenim un personatge físic que representi al agent de DialogFlow, no seria necessari disposar d'un Agent Script, ja que no hem de realitzar cap acció. Tot i així si les respostes de l'agent tenen repercussions a l'aplicació, el script que les gestionés faria el rol de Agent Script.

3. DictationScript:

   Altrament, si es desitja el ús de veu per a comunicar-se, es necessari el script `DictationScript`, des de ell s'assigna la referència a `ApiAiModule` per a que cridi a la funció que envia les peticions a DialogFlow. El seu ús recau en les funcions `StartRecognizer()` i `StopRecognizer()`, les quals el desenvolupador decideix com i on les crida. En aquest projecte la funció `StopRecognizer()` es crida automàticament després d'haver accionat `StartRecognizer()` un cop ha passat un temps definit per la variable `maxListeningTime`.

4. WindowsVoice:

   És un component completament independent, només hi ha un per escena, i qualsevol script pot enviar text a la seva funció 'speak(string)' per a que ho afegeixi a la cua de frases que es pronunciaran de veu.

El projecte consta de quatre versions, de les quals, la tercera es incremental de la segona, així doncs, considerem tres parts.

1. Interfície 2D per a comunicar amb DialogFlow.

   En aquesta versió del projecte es disposa únicament d'una escena amb la interfície 2D. És útil per a provar la comunicació amb un agent que estiguem desenvolupant en DialogFlow, ja sigui per text o per veu. Per posar-la en funcionament només es requereix de donar Play al Unity.

   Es disposa d'un UnityPackage, anomenat '`ConvAgent_AllInOne.unitypackage`', amb tots els components necessaris per fer funcionar un agent de DialogFlow, inclosa la escena d'aquesta versió del projecte. Per integrar-ho a qualsevol altre projecte únicament s'ha de fer doble click i Unity ho importa al projecte que estigui obert en el moment. La escena que conté la interfície 2D s'anomena `ConvAgent_TestScene` i es troba dins la carpeta '`_Scenes`' del directori del projecte.

   Si es vol aplicar aquesta escena a un altre agent de DialogFlow, tan sols s'ha de canviar el token de connexió. Per a fer això:

   (a) Ens dirigim al GameObject `ApiAI_CustomModule`.

   (b) Obrim el component `ApiAiModule.cs` amb el editor de codi que vulguem.

   (c) Busquem la variable `ACCESS_TOKEN` (línia 80) i hi introduïm el token del nou agent.

   S'ha de tenir en compte que aquesta escena comparteix scripts amb les posteriors, és degut això que alguns camps del script estan buits. Partint del UnityPackage que comentavem, amb els components d'aquesta escena per a ser importats a qualsevol projecte, tenim la seguretat de que es poden canviar tots els scripts que es desitgi sense afectar a cap altra part del projecte de Fracsland.

2. Agents Vendor i Assistant en dos escenes independents, dins del projecte original de Fracsland.

   En aquesta versió disposem de dos escenes molt semblants, amb la diferència que una conté al agent Vendor, i una versió bàsica de la botiga, i l'altre conté al agent

Assistant, juntament amb el personatge que controla el jugador a Fracsland. L'escena que conté l'agent Vendor s'anomena '`DialogFlow_Vendor`' mentre que l'escena que conté a l'agent Assistant es diu '`DialogFlow_HeroAssist`', ambdues dins la carpeta '`ConversationalAgent`'.

Per a utilitzar qualsevol de les dos escenes el procediment es senzill, únicament s'ha de clicar al Play del editor de Unity un cop més.

Ens hem d'assegurar que DictationScript té les referències correctes als botons de 'Listen' i 'Stop', ja que sinó donarà error. Addicionalment, també és indispensable que tingui accés al ApiAiModule del agent al que està enviant les peticions. El camp de 'currentBot' només s'utilitza en la versió VR, així que podem obviar-ho.

Si volguesim crear un agent de nou, comptant que ja tenim `DictationScript` i `WindowsVoice` a l'escena, els passos serien els següents:

(a) Crear un nou GameObject a Unity. Aquest objecte contindrà el ApiAiModule del nou agent.

(b) Copiar el script existent de ApiAiModule i modificar-lo segons es desitgi.

(c) Crear, a `DictationScript` una nova variable per a referenciar el nou ApiAiModule creat.

(d) (Opcional) Si es vol generar algun feedback o fer alguna acció amb les respostes de DialogFlow, necessitarem un script (durant el projecte se'l ha anomenat 'Agent Script') i fer-li una funció que rebi la resposta de ApiAiModule i en generi algun comportament. Des de aquest script es pot, per exemple, remetre les respostes rebudes cap a `WindowsVoice`. Aleshores a ApiAiModule hem de cridar a aquest nou mètode al rebre la resposta de DialogFlow.

(e) Omplir totes les referències que puguin faltar, tals com els botons a `DictationScript`. A la Figura 49 es pot veure com a exemple, els scripts `DictationScript` i `ApiAiModule` amb les referències pertinents.

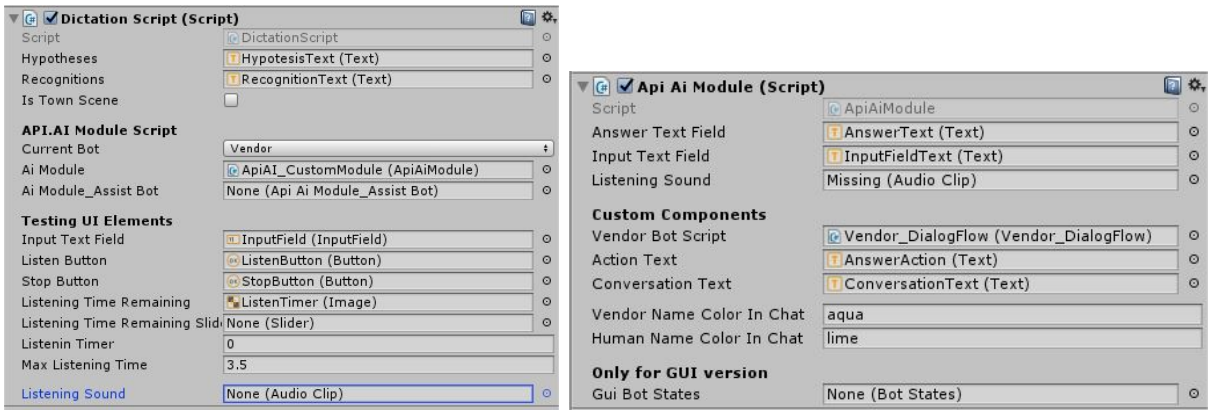

Figura 49: DictationScript i ApiAiModule amb les variables més importants

3. Agents Vendor i Assistant integrats a la versió joc Fracsland VR.

   En aquesta versió els agents son integrats al de Fracsland VR. El que hem de saber per a aquesta escena és que el script `VendorTrigger` és el encarregat de seleccionar el bot actiu en el cas de estar els dos agents a la mateixa escena (escena TownVR).

   El funcionament dels scripts, a part d'aquest detall, es exactament el mateix.

Apèndix B: Manual del jugador

En aquest apèndix s'explica com utilitzar als agents conversacionals creats en aquest projecte. L'ús dels agents en les escenes prèvies a la integració a Fracsland VR és senzill. A continuació es mostra a la Figura 50 la disposició dels elements que es mencionen a la escena.

Figura 50: Disposició dels elements de UI per comunicar amb el agent.

- Si volem comunicar de veu:
  1. Cliquem al botó 'Listen'.
  2. Veurem que el cercle del costat comença a reduir-se, es el temps restant per parlar.
  3. Podem clicar 'Stop' manualment o esperar a que passi el temps per aturar la grabació.
  4. De forma automàtica s'enviarà el resultat a DialogFlow i aquest contestarà.
  5. La resposta es mostrarà al quadre de text assignat, i es parlarà. A excepció de la primera versió del projecte (interfície 2D), en les altres tambe es mostrarà una bombolla de text amb la resposta.
- Si volem comunicar a través de text:
  1. Escrivim el que desitgem dir al bot al quadre de InputText.
  2. Cliquem al botó 'Send Text' al costat del input o polsem la tecla 'Enter' o 'Return' del teclat.
  3. La resposta es mostrarà al quadre de text assignat, i es parlarà. A excepció de la primera versió del projecte (interfície 2D), en les altres tambe es mostrarà una bombolla de text amb la resposta.

A continuació s'explica l'ús dels agents dins el joc de Fracsland VR. Al tractar-se d'una plataforma VR l'única via de comunicació amb els agents es a través de la veu. En aquets punts es dona per suposat que el jugador domina els controls de FracslandVR, explicats en el manual del jugador de la memòria del projecte [13].

- Per a comunicar amb el Vendor:

  1. Hem d'apropar el jugador al Vendor.
  2. Quan siguem suficientment a prop, el Vendor ens saludarà.
  3. Aleshores el usuari mantén pressionat el botó assignat a parlar amb els agents. (botó Y o B, depenent de si l'usuari és dretà o esquerrà, la distribució dels botons es pot apreciar a la Figura 51).
  4. L'usuari pronuncia el que vol dir-li.
  5. L'usuari rebrà la resposta, de text, a través d'una bombolla de diàleg, i de veu.

- Per a comunicar amb l'Assistant:

  L'única condició per comunicar-se amb l'Assistant és no estar a prop del Vendor.

  1. Allunyar-se del Vendor en cas d'estar a prop.
  2. Aleshores el usuari mantén pressionat el botó assignat a parlar amb els agents. (botó Y o B, depenent de si l'usuari és dretà o esquerrà, la distribució dels botons es pot apreciar a la Figura 51).
  3. L'usuari pronuncia el que vol dir-li.
  4. L'usuari rebrà la resposta, de text, a través d'una bombolla de diàleg, i de veu.

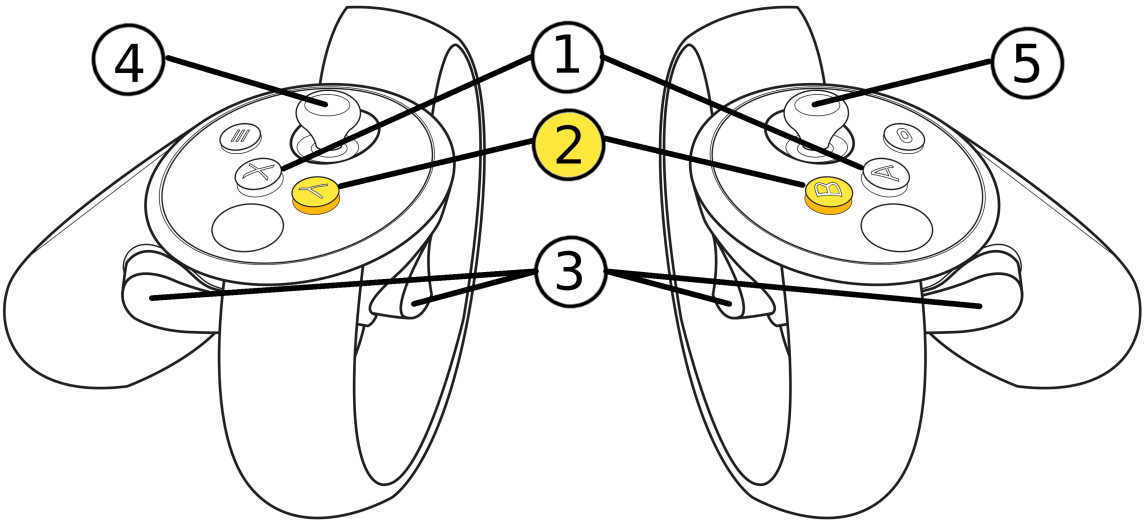

Figura 51: Distribució dels botons al comandament d'Oculus Touch.