D7.5 - AGENT INTEGRATION DEMONSTRATION

Grant Agreement nr	856879
Project acronym	PRESENT
Project start date (duration)	September 1st 2019 (36 months)
Document due:	August 31st 2022
Actual delivery date	August 31st 2022
Leader	Framestore
Reply to	Richard.Ollosson@framestore.com
Document status	Submission Version

Project funded by H2020 from the European Commission

Project ref. no.	856879
Project acronym	PRESENT
Project full title	**P**hotoreal **RE**altime **S**entient **ENT**ity
Document name	Agent Integration Demonstration
Security (distribution level)	Public
Contractual date of delivery	31/08/2022
Actual date of delivery	31/08/2022
Deliverable name	D7.5 Agent Integration Demonstration
Type	Demonstrator
Status & version	Submission Version
Number of pages	20
WP / Task responsible	FS
Other contributors	all partners
Author(s)	Theo Jones, Richard Ollosson, Adam Alsegard
EC Project Officer	Ms. Diana MJASCHKOVA-PASCUAL Diana.MJASCHKOVA-PASCUAL@ec.europa.eu
Abstract	Key aspects of the realtime agent integration architecture and functionality along with demonstrations of the adaptation of that architecture to meet a wide variety of use case needs.
Keywords	digital human, real-time, Unreal Engine
Sent to peer reviewer	Yes
Peer review completed	Yes
Circulated to partners	No
Read by partners	No
Mgt. Board approval	No

Document History

Version and date	Reason for Change
1.0 28-06-2022	First Draft created by Theo Jones
1.1 25-08-2022	Version reviewed by Roberto De Prisco
1.2 31-08-2022	Version for submission post internal peer review

TABLE OF CONTENTS

1. EXECUTIVE SUMMARY

This deliverable presents the key architecture and protocols of the integrated agent, both mid-resolution and high-resolution. These systems have been developed within and alongside Unreal Engine. Unreal Engine is the most widespread game engine in use today for high fidelity characters and therefore gives the maximum possible access, flexibility and visual quality for the final demonstrations and ongoing development.

An overall architectural design of the agent integration within the Reference Implementation is first presented in section 4.1, followed by a description of the different partner components in section 4.2. Section 4.3 then makes a comparison of the agents and demonstrates the framework that allows for the high-resolution agent, created by Framestore as part of the WP3, to be driven directly from the animation data targeted to the mid-resolution agent, created by Cubic Motion as part of their contribution to WP3. This showcases the capability of the final implementation to be tailored to the specific computational system resources available to the user.

Thanks to Cubic Motion's position as part of Epic games the mid-resolution agent has been built based upon the Epic MetaHuman character framework, which became publicly available in April 2021 and is now widely adopted, therefore giving maximum utility to the final uses of the agent. The wide variety of gender and ethnicities available on the MetaHuman architecture also addresses a key diversity weakness identified in the project.

The demonstrations and associated agent capabilities presented here are centred on the specific use cases selected as part of the Present project final deliverable:

- Adam Use Case
    - Demonstrates integration of the emotional recognition work carried out by University of Augsburg as part of WP4, the high-resolution virtual agent developed by Framestore as part of WP3 and the emotional modelling work implemented by Cubic Motion as part of WP3.
- Sports Broadcast
    - Demonstrates integration of InfinitySet and the virtual studio carried out by Brainstorm as part of WP7 and the dialogue, animation and gaze management carried out by Cubic Motion as part of WP3.
- Registration Authority Officer
    - Demonstrates integration of the security protocols implemented by InfoCert in WP5, the high-resolution virtual agent developed by Framestore as part of WP3 and the decision logic carried out by UPF as part of WP6 .
- Delirious Departures
    - Demonstrates integration of motion-matching capabilities developed by Cubic Motion as part of WP3, 1-n agent-to-agent interaction implemented by Inria as part of WP4 and the experimental production work carried out by CREW as part of WP8.

Although the demonstrations presented here relate specifically to the PRESENT use cases the Reference Implementation architecture is designed in such a way as to be easily extensible to satisfy use cases not directly covered.

2. BACKGROUND

This deliverable belongs to the task WP7T2, Agent Integration, led by the University of Augsburg, and the Adam Use Case demonstration is included to cover this functionality. However, given Framestore's role in integrating the two agent resolutions and the multiple partner components, a wider demonstration of the various agent functionalities and the integration architecture is included. This deliverable builds on work presented in multiple partner component demonstration deliverables across WP3, WP4, WP5 and WP6. In particular it should be viewed alongside D5.5, Protocols and API's Implementation, where details can be found of mechanisms for the adaptation of the reference implementation for new use cases, and D3.4, where the procedure for the addition of novel mid-resolution agents via the MetaHuman framework is provided.

3. INTRODUCTION

Framestore was tasked with developing a common project architecture that could be utilised by all project partners to both integrate their individual components and configure those components, along with others developed by other partners on the project, to meet the needs of a wide variety of use cases.

This deliverable details the reference implementation architecture developed to meet the above requirements, while utilising a number of the project's use cases to demonstrate the success of this architecture in adapting to meet the wide variety of use case demands.

3.1 Main Objectives

- To provide a common project framework that allows for the easy integration of multiple, varying external components covering functionality such as dialog management, emotional modelling, locomotion and security.
- To develop a project architecture that allows for the easy adaptation to novel use cases and the configuration of the functional components to meet bespoke requirements.
- To integrate multiple resolutions of virtual agents to allow for the adaptation of the use cases to different computational budgets and visual fidelity requirements.

3.2 Terminology

- **BML:** Behaviour Markup Language[1] is a XML description language used to control verbal and nonverbal behaviour of (humanoid) embodied conversational agents (ECAs). The standard defines the form and use of BML blocks, mechanisms for synchronisation, the basic rules for feedback about the processing of BML messages, plus a number of generic basic behaviours.

- **Epic**: Refers to Epic Games Inc., the company that develops Unreal Engine and its native features.

[1] https://projects.cs.ru.is/projects/behavior-markup-language/wiki (accessed 18/08/2022)

- **FIRA**: Fast Immersive Rigging and Animation, refers to the machine learning rig evaluation component developed by Framestore as part of *D6.2 Fast Renderer Demonstration.*
- **InfinitySet**: Application developed by Brainstorm to control virtual studio integrations[2].
- **MetaHuman**: Refers to digital human avatars created in Epic's MetaHuman Creator tool[3]. Its introduction has made it much simpler for users without intrinsic knowledge in advanced Digital Content Creation applications to create realistic digital humans of great diversity.
- **Reference Implementation:** This refers to the Unreal Engine project responsible for the integration of the various PRESENT partner components and their configuration to satisfy the requirements of individual use cases.
- **UE**: Unreal Engine, the game engine used for the reference implementation. Developed by Epic Games. Unless specified otherwise 4.27 is the version that is referred to[4].
- **UMANS**: Unified Microscopic Agent Navigation Simulator[5] is a crowd-simulation framework developed by Inria that can reproduce many different algorithms for local navigation behaviour (such as collision avoidance) in crowds.

4. AGENT INTEGRATION

This chapter gives a brief overview of the architecture of the Reference Implementation in Unreal Engine (UE) and then describes how the different virtual agents have been integrated into that project. This includes both the medium-resolution (mid-res) agent, based on Epic's MetaHuman and further developed by Cubic Motion, and the high-resolution (high-res) agent created by Framestore. This chapter also describes the differences between the agents, how the APIs relate to each other and which agent is better suited for each use case.

4.1 Reference Implementation Architecture

The UE reference implementation project was added to create a common platform for all use cases with an architecture that could be utilised by all partners. This section will give a brief overview of that architecture. For the full documentation of the reference implementation and how to use it please see *D5.5 Protocols and APIs Implementation*.

The reference implementation is structured in a modular way to make it easy to reuse different "actors" and components in the different use case "maps". A "map" or "level" in UE is a new world with spawned objects that it owns. One map can be loaded into or unloaded from another map, but it will always keep ownership of its objects. An "actor" in UE is any object that can be spawned in a world. Not all actors have to be visible. The PRESENT UE project has a collection of actors that are common for each use case,

[2] https://www.brainstorm3d.com/products/infinityset/ (accessed 18/08/2022)
[3] https://metahuman.unrealengine.com/ (accessed 18/08/2022)
[4] https://docs.unrealengine.com/4.27/en-US/WhatsNew/Builds/ReleaseNotes/4_27/ (accessed 18/08/2022)
[5] https://project.inria.fr/crowdscience/project/ocsr/umans/ (accessed 18/08/2022)

some actors that are swapped depending on the use case (e.g. mid-res agent vs high-res agent) and a few data objects or logic implementations that are specific to a single case.

To enable parallel development one base map was created for each use case, initially only containing the common actors. As the project evolved some maps added actors or sublevels (for example with a standard lighting setup or a collection of target actors) while others removed a few. Each map enables the partner plugins needed for its use case and loads the common actors with the correct configuration. The objects and logic related to the common actors have been collected and organised separately so that the same actors can be reused in a different use case, but with different configuration. Figure 1 demonstrated how some of these objects have been organised in the project.

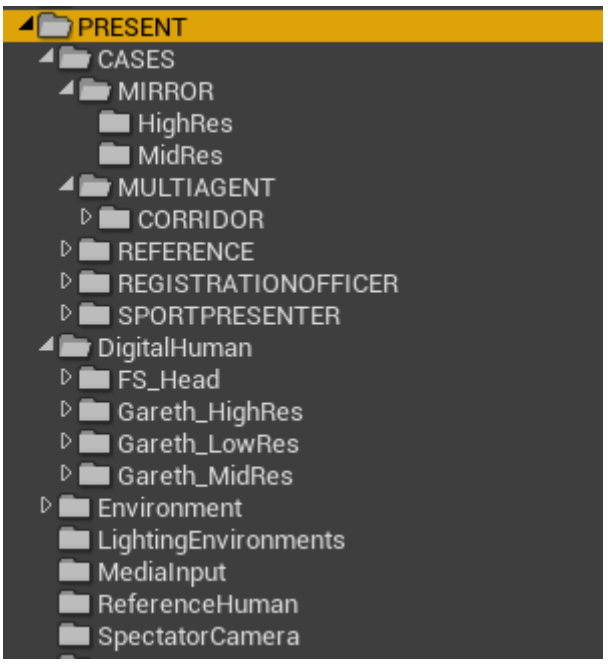

Figure 1: The content folder structure in the reference implementation project

4.1.1 Common actors

The two main common actors are the *Digital Human* and *Digital Human Processing*. The Digital Human is the virtual agent and can be either of medium-resolution or high-resolution and can be spawned either as a single actor or as multiple actors. It is built to mostly be a "dumb" actor which means that it should not have much internal logic but instead have an external "controller" which takes all the input needed, processes it with logic depending on the use case and then produces instructions for how the virtual agent(s) should act. That controller is the Digital Human Processing actor. It has direct access to the Digital Human and all other actors in the map and can direct how they should behave. This is where the partner components (described in section 4.2) are loaded and connected which then determine the logic for the specific use case. An example of how a Digital Human Processing blueprint can look is shown in Figure 2. The processing actor is invisible by default but it is possible to display a representation for each of the enabled components which reports debug information about the data sent and received.

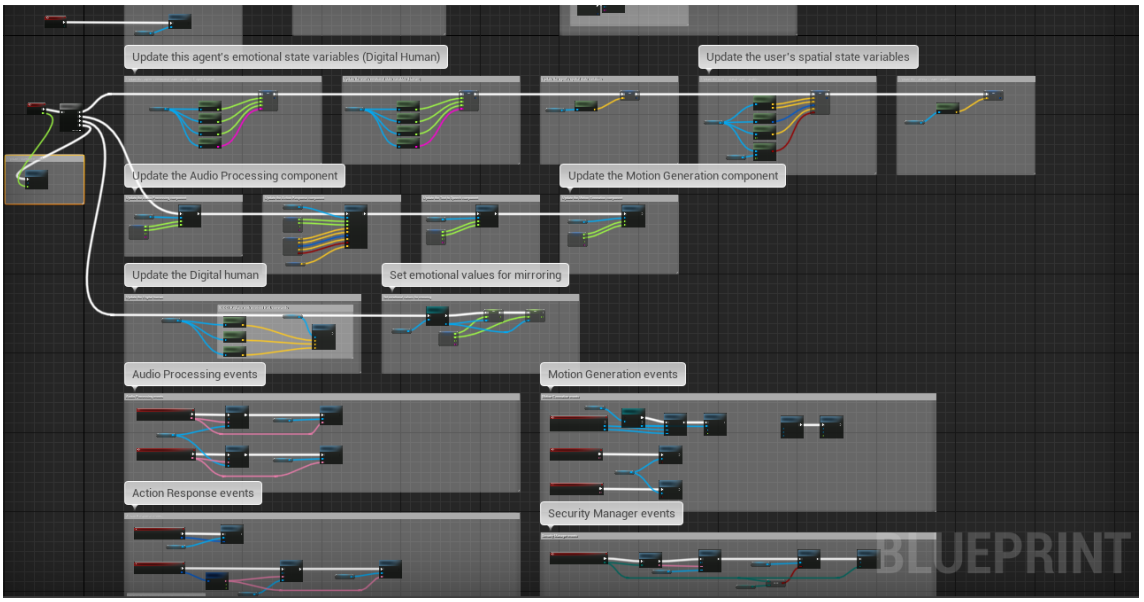

Figure 2: A subset of the "Event Graph" blueprint in a Digital Human Processing actor

Other examples of common actors are:

- The *Input Controller* which handles user input from keyboard, mouse, gamepad or any other device.
- The S*pectator Camera* which sees the scene from a third person point of view. Is controlled by the Input Controller. It is possible to add multiple spectator cameras and switch between them during runtime.
- The *Reference Human* which is a basic representation of the user in the world. It has functions that can be called by the Input Controller, or used by the Digital Human Processing in a particular use case.

4.2 Partner components

The main logic is implemented in different components that are included in the Digital Human Processing actor and connected differently depending on the use case. The processing actor can either pull data from the components with certain intervals, for example on every frame, or it can react on events when new data has been produced by the components.

To facilitate partners working on integrating their functionality in parallel the components were created as separate UE plugins. Partners could then either implement their logic directly in the UE plugin or communicate with their API through a pre-built DLL or a web service. The diagram in Figure 3 shows the different components and the initial suggestion for how the data would flow and how components would communicate with each other. A more detailed description of all components follows.

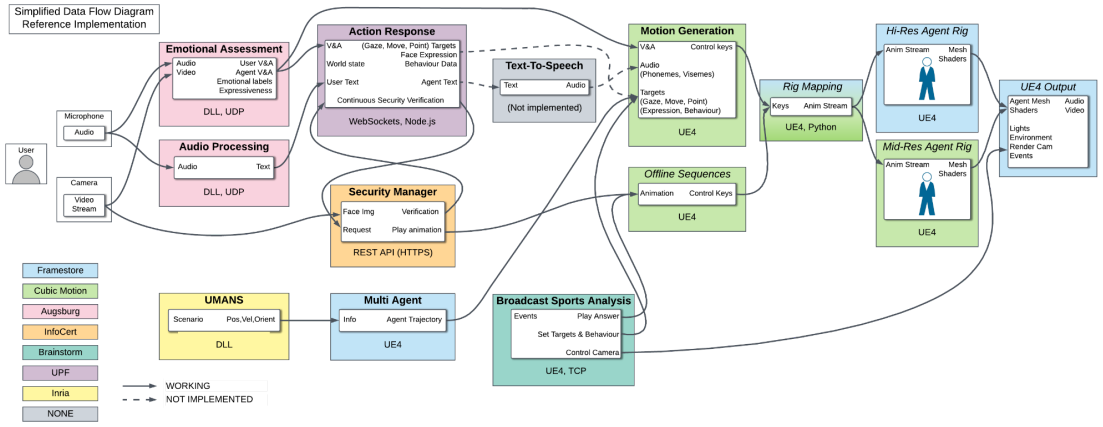

Figure 3: A simplified data flow diagram of the partner components

4.2.1 Emotional Assessment

The purpose of this component is to determine the current emotional state of the user and simulate the emotional state of the agent by using video and audio streams of the user. That functionality was implemented by the University of Augsburg as a separate application (called *SSI Pipeline*) which is explained in depth in *D4.5 Agent Social Interpretation Enabling.*

The SSI Pipeline communicates with the reference implementation project via UDP sockets. This is suitable as the output from the application is primarily a stream of valence and arousal values for both the user and the agent and it does not matter too much if a single message is lost. An expressiveness factor can also be received and from the received values an emotional label can be calculated. New emotional values are pulled by the Digital Human Processing on every frame.

4.2.2 Audio Processing

Its purpose is to convert the user's speech audio to text. This functionality was also included by the University of Augsburg in the SSI Pipeline and uses the same UDP connection. Every time a sentence has been parsed it is sent to the Digital Human Processing which broadcasts it to all listener components.

The use cases selected for the final PRESENT deliverable had no direct need for this component and it is therefore not showcased in the results. If it would be needed in the future one potential improvement could be to use a TCP connection instead to make sure no sentences were lost or received out-of-order.

4.2.3 Action Response

The function of this component is to take the emotional values of the user and agent along with other information about the world and targets defined by the user and then produce facial and body behaviour for the agent. For example, take a location target to point to, another to look at and a third to move to, along with information about where the user is and which emotional state the agent is in. It also takes the user speech text and produces a text response for the agent.

Although no single use case utilises all aspects of this component, the different functionalities of this component are used by separate use cases. For example, the *Behaviour Planner* developed by UPF produces behaviour triggers and scripted text responses for the agent in the *Virtual Clerk* demonstration (see *D8.5 Prototype Evaluation Results* for a detailed description). An initial connection from the reference implementation project to the Behaviour Planner was set up with WebSockets[6] and later changed to NodeJS[7]. However, as the Virtual Clerk use case already demonstrated the full functionality of the Behaviour Planner in a WebGL[8] application with a lower resolution agent it was decided that the reference implementation would focus on showcasing the part of the script that interacted with the Security Module developed by InfoCert. This would be integrated with the high-res agent as a proof-of-concept and is demonstrated in the Registration Authority Officer use case (see section 5.3).

Another example of similar functionality is the possibility to send BML commands to trigger certain behaviours in the agents. This functionality was moved to the Motion Generation component and is explained further in section 4.2.5.

Generating a text response for the agent was not part of any of the partner use cases in WP7 and has therefore not been demonstrated here.

4.2.4 Text-to-Speech

The purpose of this component is to convert a generated agent response from text to audio while also producing the visemes and phonemes to be used by the agents when pronouncing the audio. As with the generation of text responses this was not needed for any of the partner use cases in WP7 and has therefore not been demonstrated here.

4.2.5 Motion Generation

The purpose of this component is to control the motion of the agents. Cubic Motion has introduced several different ways to interact with this component. It can now play animation assets and sequences directly or stream control keys every frame, both for the mid-res and high-res agent. It can also trigger certain behaviours by BML commands, for example setting a target to move to or shaking/nodding of the head. This is also where the motion-matching functionality is integrated as the different behaviours and animations are blended together and then matched to the subsequent animation. Cubic Motion also added an *Emotional Modelling* functionality that created a more realistic emotional space for the agent (see *D6.4 Interactive Facial Animation Demonstration*).

The default implementation of each feature is targeted to the mid-res agent and implemented directly into UE in a few different plugins. A couple of the features have also been retargeted to the high-res agent by Framestore. Specifically playing animation clips and sequences directly, streaming control keys and using certain BML commands to trigger behaviours.

[6] https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (accessed 18/08/2022)
[7] https://nodejs.org/en/ (accessed 18/08/2022)
[8] https://www.khronos.org/webgl/ (accessed 18/08/2022)

4.2.6 Security Manager

The responsibility of this component is to communicate with the *Security Module* developed by InfoCert (see *D5.6 Trust and security infrastructure software components v2*) via a REST (HTTPS) API. It can start a new authorization process by taking a photo of the user's face and send it to the Security Module. The user then needs to verify their identity by scanning a QR code displayed in the reference implementation with the mobile app developed by InfoCert. If something goes wrong along the way or if the authentication fails it will try again until successful or the configuration tells it to stop.

Once the initial authentication has been successful the user can interact with the rest of the project. A new verification check will be performed continuously, on a configurable interval, to make sure that the user is still authorised. This will once again take a photo of the user's face and match it with the database but no QR code needs to be scanned once the initial authentication has been completed.

The proposed protocol by InfoCert was for both the reference implementation and the security server to send REST requests to each other. However, as the reference implementation can be run inside secure firewalls this proved to be more difficult than initially expected. The protocol was therefore changed so that only the reference implementation sends requests. Instead of having the server send back the results when ready, the UE project checks for new results periodically until a successful or failed authentication is produced.

4.2.7 Broadcast Sports Analysis

This component was implemented by Brainstorm with the purpose of communicating with their *InfinitySet* application (see D7.*4 Virtual Studio Integration Report*). It relays commands and events both ways and can for example trigger specific animations via the Motion Generation component and receive an event when the animation has finished playing. It communicates to InfinitySet via a TCP connection and has a Web Remote Control[9] (HTTP) interface set up as well.

4.2.8 Multi Agent

This was the last component to be added and its purpose is to work as an interface to the UMANS library developed by Inria, especially to make use of the *Interaction Fields* functionality (see *D4.7 Reactive Agent and Touch Enabling*). It can start a prepared scenario with multiple agents or generate a completely new scenario with specified configuration from the user. This component is responsible for everything regarding the agents on the UE side, for example spawning, updating and removing them.

UMANS is integrated as a pre-built DLL and its output is mainly the position, velocity and orientation of all the simulated agents. The Multi Agent component can then calculate and visualise the trajectory for all the agents. Then they are animated by using motion-matching via the Motion Generation component.

[9] https://docs.unrealengine.com/4.27/en-US/ProductionPipelines/ScriptingAndAutomation/WebControl/ (accessed 18/08/2022)

The mid-res agent is the default one used but other actors such as the UE4 Mannequin or any StaticMesh or SkeletalMesh actor could also be simulated. The motion-matching has been trained with the mannequin and should work for other MetaHumans as well. The high-res agent would not work well with this component as it has been optimised for rendering of a single agent and most normal workstations are not powerful enough to render several high-res agents simultaneously in real-time.

4.3 Virtual Agents

From the start of the project it was decided that multiple versions of the agent were needed, primarily as the high-resolution agent developed by Framestore would be too computationally heavy for several of the use cases. Therefore, Cubic Motion developed a medium-resolution version of the scanned actor as well. Thanks to Cubic Motions position as part of Epic Games the mid resolution agent was able to be targeted to the MetaHuman character framework ahead of its official release giving the project an advantage in terms of flexibility and diversity.

The mid-res agent uses MetaHuman joints, controls, clothes and an UE HairStrands rendered groom. The first version of it was integrated into the reference implementation project in v3.2.0 and shared with partners in M19. The high-res agent on the other hand uses an internal rig and drives both face, body and clothes with the internal machine learning FIRA component (see *D6.2 - Fast Renderer Demonstration*). However, it still uses the UE HairStrands plugin to render the groom. It was integrated in v4.0.0 and shared with partners in M31. Screenshots with static agents are showcased in Figure 4.

Figure 4: Visual comparison between the mid-res agent (left) and high-res agent with basic lighting/settings (middle) and high-res agent with final custom lighting/settings taken from FMX demo (right)

4.3.1 Mid-Resolution to High Resolution Agent Retarget System

The Motion Generation component was developed by Cubic Motion with the primary focus to drive the animation of the mid-res agent. As that is based on a MetaHuman rig other MetaHuman agents should work automatically as well. However, to make use of the same functionality for the high-res agent a retargeting system for both body and face was needed to be implemented.

A pipeline to retarget offline animations outside of UE has been developed by Framestore. However, in some cases, as for the Adam use case (see section 5.1) the animation data is streamed per frame and not available ahead of time. For those cases a retargeting system has been added into the reference implementation as well which uses animation blueprints[10] for both the face and body. The body mapping is fairly straight-forward as it only retargets MetaHuman joints to high-res rig joints. The MetaHuman face on the other hand is animated with ControlRig[11] controls whereas the high-res face is controlled by custom floating point attributes and the mapping is not 1-to-1 to get the same expression. There are also a few cases where something is controlled by a joint in one rig but by a control or an attribute on the other which adds to the complexity of the remapping. A complex custom remapping was therefore developed by Framestore to allow the flexibility of retargeting use cases to either the mid-res or high-res agent as appropriate. Figure 5 shows a part of the remapping blueprint needed for the face.

The animated values for the high-res agent are then fed into the FIRA machine learning component developed by Framestore (see *D6.2 - Fast Renderer Demonstration*) which drives the rig. FIRA's purpose is to optimise the animation evaluation of rigs with high complexity. It has been trained on the high-res agent and can drive the body, face and clothes in real-time.

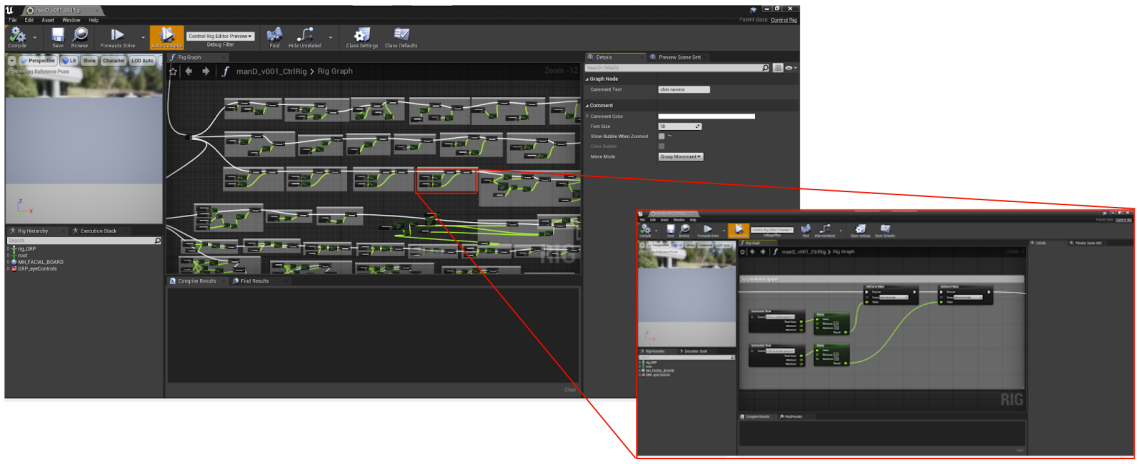

Figure 5: A subset of the facial retargeting animation blueprint in UE

[10] https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/SkeletalMeshAnimation/AnimBlueprints/ (accessed 18/08/2022)
[11] https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/SkeletalMeshAnimation/ControlRig/ (accessed 18/08/2022)

Because of the differences between the agents every new animation or feature that is produced for the mid-res agent needs to be retargeted before it can be used with the high-res agent. Offline animations can now be processed ahead of time and the mapping from MetaHuman controls to the high-res agent attributes enables streaming during runtime. However, the emotional modelling uses a space that is created by a set of emotional expressions that needs to be retargeted and the motion-matching is using a base set of animations and a blending directly on the joints in an animation blueprint which all need to be retargeted before those features can be used for the high-res agent.

A demonstration of the retarget system in action can be found in the accompanying video:

HighResAgentRetarget_Demo.mp4

This video first shows the performance capture animation collected by Cubic Motion and targeted to the mid-res agent. It then goes on to show the same animation data retargeted to an offline rendered asset as well as the high-res agent in Engine. Two more clips follow to showcase the quality of the high-res agent and its ability to be rendered at realtime framerates in the Unreal Editor UI.

4.4 Unreal Engine 5

During the development of the project Epic released a new major version of Unreal Engine[12]. By the time it was available to all partners the bulk of the reference implementation project had already been implemented in version 4.27 and for some partners, such as Brainstorm, an upgrade to UE5 was not feasible in the timescope of the PRESENT project. Therefore, it was decided that the reference implementation should stay on 4.27, even if that meant that new functionality in UE5 could not be used. This could have improved certain features, especially related to the motion-matching.

However, there is a working build of the reference implementation in UE5 with most of the base functionality. Figure 6 shows the map used for testing all components being run in UE5. The biggest exclusions in this build are the Broadcast Sports Presenter component, with its integration to InfinitySet, and some of the FIRA data needed for rendering the high-res agent. This build is not used in any of the presented use cases as the UE5 release came too late in the schedule. However, it clearly demonstrates that a future utilisation of UE5 is entirely possible.

[12] https://www.unrealengine.com/en-US/unreal-engine-5 (accessed 18/08/2022)

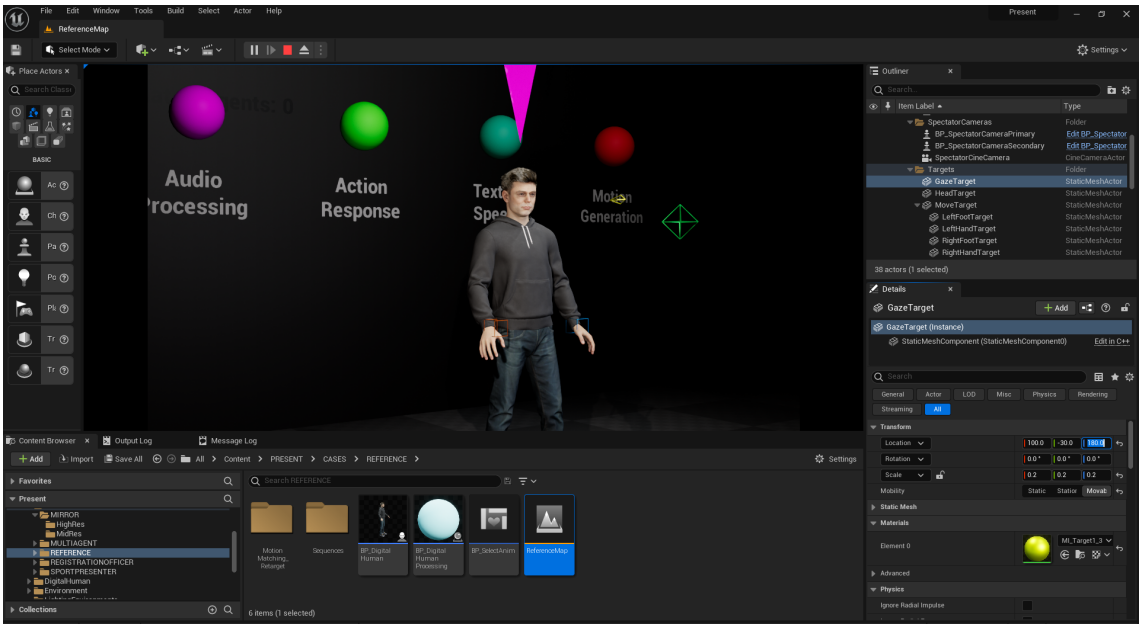

Figure 6: The "Reference Map" test case running in UE5 with the mid-res agent

5. RESULTS

This chapter presents each demonstrated use case, with a description of all the different components that are used for that specific implementation.

5.1 Adam Use Case - Non-Verbal Interaction

The purpose of the Adam use case is to create an agent that the user can interact with non-verbally. The agent uses the Emotional Assessment component to determine the emotional state of the user from voice and facial expression and then generates an emotional state for the agent. That state is given to the Motion Generation component which produces an appropriate facial expression and maps it to the current agent. This can be used for both the mid-res and high-res agents. For the mid-res agent it is also possible to use the emotional modelling developed by Cubic Motion for a more sophisticated emotional space. A couple of visual comparisons between the agent version can be seen in the Appendix.

The emotional values can also be simulated in other ways, for example completely randomised, mimicking the user or generated with a control like a gamepad. This can be used in a demonstration with a separate admin user who controls the emotional state of the agent while the other user interacts. This is briefly demonstrated by CREW in *Adam_UseCaseDemo.mp4*, followed by the normal state where the emotional values are generated and finally the agent is mirroring the emotion of the user. The video shows the version of the Adam use case with the high-res agent.

Partner components used: Emotional Assessment & Motion Generation
Compatible agents: Mid-Res & High-Res

Demo Videos: ***Adam_UseCaseDemo.mp4***
EmotionalModelling_Demo.mp4
GazeManagment_Demo.mp4

5.2 Sports Broadcast Use Case - Dialogue and Motion Management

This use case demonstrates the integration to InfinitySet and the virtual studio developed by Brainstorm along with Cubic Motion's functionality for playing scripted animation clips, emotional modelling and setting targets for where the agent should look. An in-depth evaluation of this use case has been carried out in *D7.4 Virtual Studio Integration Report.*

In the demonstration video (*SportsBroadcast_UseCaseDemo.mp4)* InfinitySet is in control of the scene. It sets the camera, environment, lighting, graphics as well as adding other actors like the table and then streams the mid-res agent from the reference implementation project into the scene. InfinitySet can then trigger animations for the agent and react to events, e.g. when an animation finishes playing or the agent finishes talking. It can also set location targets for what the agent should look at or point to. That behaviour is then blended with the underlying animation. When a scripted animation finishes the agent blends into an idling animation which runs until the next scripted animation is triggered.

Partner components used: Broadcast Sports Analysis & Motion Generation
Compatible agents: Mid-Res

Demo Video: ***SportsBroadcast_UseCaseDemo.mp4***

5.3 Registration Authority Officer Use Case - Security Management

This use case is a subset of the full Virtual Clerk script (see *D8.5 Prototype Evaluation Results*) and its purpose is to be a proof of concept that the Security Module developed by InfoCert can interact with the high-res agent as well.

The demonstration video (*RegistrationAuthorityOfficer_UseCaseDemo.mkv)* starts with a short introduction and a few scripted choices for the user to click through. Then a new authentication process is started by taking a picture of the user's face. The user then has to scan a QR code with the mobile application developed by InfoCert. The data is sent to InfoCert's server and checked against the ground truth image that the user took when creating the account. The video then demonstrates what happens if the authentication fails. After a retry the authentication is then successful and the user can interact with the full PRESENT project. A continuous check will be performed at certain intervals to make sure that the user is still authenticated. This interval is rather short in the video for demonstration purposes.

When the user has been authenticated the agent follows the user's camera with its gaze. The gaze target is also set to different locations and actors during the scripted animations. When the user can interact with the project again the different targets can be set to absolute or relative locations or parented to any actor in the world with the use of BML commands. BML can also be used to trigger certain animations or change the location of the camera.

Partner components used: Security Manager & Motion Generation
Compatible agents: High-Res

Demo Video: ***RegistrationAuthorityOfficer_UseCaseDemo.mkv***

5.4 Delirious Departures Use Case - Multiple Agents

The Delirious Departures use case was created by CREW and demonstrates that multiple agents can be simulated in complex social situations by the UMANS Interaction Fields and then visualised in Unreal. For their demonstration at SIGGRAPH 2022 (see *DeliriousDepartures_UseCaseDemo.mp4*) they used the UE4 Mannequin as the simulated actor but the same functionality works for the mid-res agent (as seen in *MultiMidResAgent_UseCaseDemo.mkv*). The reason CREW used the mannequin was purely performance related as the mid-res agent is inherently more complex to render, which did not suit the hardware used for this particular installation.

When running any use case with the Multi Agent component enabled it will control how to spawn, update and remove the simulated agents. The trajectory for all agents are updated on every frame and can be visualised and if the rendered agent type supports motion-matching it will be applied through the Motion Generation component.

Partner components used: Multi Agent (UMANS) & Motion Generation
Compatible agents: Mid-Res

Demo Videos: ***DeliriousDepartures_UseCaseDemo.mp4***
MultiMidResAgent_Demo.mkv

6. CONCLUSIONS

This deliverable has described the key architectural implementation of the common project framework underlying the PRESENT digital agent integration. A number of the project use cases, as well as bespoke demonstration videos, have been utilised to clearly show how this flexible architecture can be successfully adapted to satisfy the requirements of widely varying use cases.

As shown in the Results both the medium-resolution and high-resolution agents are fully integrated within the reference implementation project. The mid-res to high-res animation retargeting system, available for both offline processing and streamed animation data, provides a mechanism for easily adapting the agents visual fidelity to any computational budget. When this is combined with the openly available MetaHuman digital human architecture, which underpins the mid-res agent, a highly flexible and diverse range of outputs is possible.

The demonstration videos accompanying this deliverable clearly shows these agents to be capable of parallel execution of dialogue, socially sound non-verbal behaviour and input cues and of blending these outputs together. The agents are also shown to be capable of interacting with external services such as the security module and pre-built libraries such as the SSI Pipeline and UMANS' Interaction Fields to create the wanted behaviour for any particular use case.

Overall it has been demonstrated that the architectural project design presented here can be used to realise the scenarios as diverse as the single character, emotionally responsive interactions in the Adam use case through to the group dynamics and locomotion of Delirious Departures. This flexibility, combined with the adaptable retargeting system and diverse MetaHuman outputs, make for an adaptable, extensible solution to placing responsive digital agents into a wide variety of production environments.

7. APPENDIX

Appendix figure 1: Comparison between mid-res and high-res agents in the Adam use case but with fixed valence (0.7) and arousal (0.6) values. The mid-res agent is also utilising emotional modelling.

Appendix figure 2: Comparison between mid-res and high-res agents in the Adam use case when generating emotional values. Emotion strived for was "surprised/scared". The mid-res agent is also utilising emotional modelling.