Avaliação de soluções de streaming para aplicar ao Avatar Recepcionista

Thiago D. S. Lacerda, Artemis Moroni, Josué Ramos

t244712@dac.unicamp.br, {artemis.moroni, josue.ramos}@cti.gov.br

Núcleo de Robótica e Visão Computacional– NRVC
CTI/MCTI Renato Archer – Campinas/SP

Abstract. *This project aims to propose a solution to the render streaming problem within the project of developing the receptionist avatar on Unity and Unreal Engine platforms. For that, alternatives were evaluated using TCP, WebRTC protocols and STUN and TURN servers, with the Unity Render Streaming package and Unreal Engine Pixel Streaming Plugin to implement the communication between client computer and server.*

Resumo. *Este projeto tem como objetivo propor soluções para o problema de renderização remota dentro do projeto do desenvolvimento do avatar recepcionista avaliando a viabilidade de uso das plataformas Unity e Unreal Engine. Para tal, foram analisadas alternativas usando protocolos TCP, WebRTC e servidores STUN e TURN, o pacote Unity Render Streaming e Unreal Engine Pixel Streaming para implementação da comunicação entre computador cliente e servidor.*

1. Introdução

Com o avanço das pesquisas em robótica, a sociedade integra-se cada vez mais com robôs e assistentes robóticos cotidianamente. Essas tecnologias, quando oferecidas para uso pessoal, promovem maior convivência entre as partes, idealmente auxiliando os usuários em tarefas manuais e mentais. A multidisciplinaridade dentro da área de Interação Humano Robô [1] é, portanto, necessária e envolve desde psicologia e sociologia a engenharia mecânica e ciências da computação. Com isso em mente foi desenvolvido a Avatar Recepcionista Ana, que incorpora aspectos de um SIR (Robô Socialmente Interativo) [2][3].

Entretanto, a interação humano-robô atualmente é possível somente se o usuário tiver acesso ao avatar embarcado, dificultando escalabilidade e locomoção do avatar, entre outras desvantagens. Transferir alguns serviços do avatar para um servidor, como síntese de voz, renderização e animação, diminuem os requisitos do sistema embarcado, facilitando assim a interação entre usuário e avatar.

2. Objetivos

Inspirado no robô recepcionista da Carnegie Mellon University (CMU), o objetivo deste projeto é propor topologias de rede para que o avatar possa ser executado remotamente em um servidor em plataforma Linux, partes integradas a plataformas de desenvolvimento de jogos multiplataforma, como Unity [4] e Unreal Engine [5]. O avatar deve manter suas atuais funcionalidades: síntese de voz, movimentação e

animações e processamento de strings em áudio; manter conexão com o *cérebro,* interpretador de dados desenvolvido pelo NRVC/CTI [6][7]; ser executado em ambiente Linux e transportar o produto de suas funcionalidades usando o protocolo de rede TCP IP [8].

3. Pesquisa

O diagrama da figura 1 apresenta o atual funcionamento geral e relações entre os subsistemas do avatar.

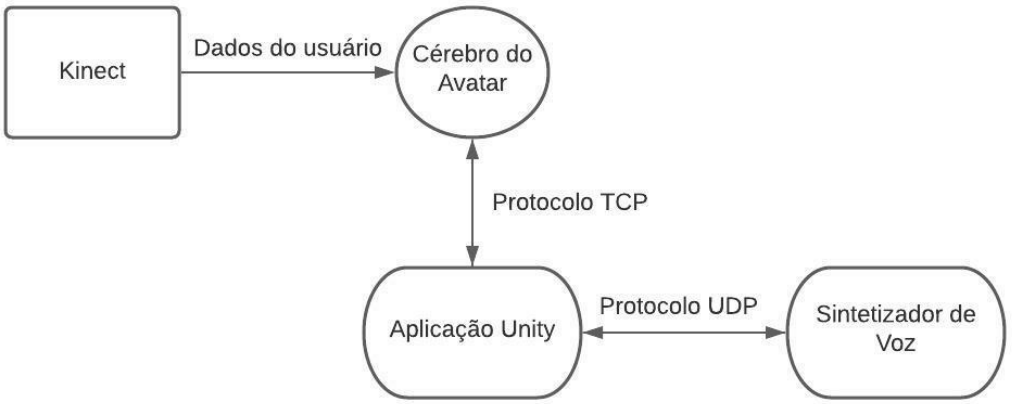

Figura 1: Topologia geral das aplicações do Avatar [3]

No computador cliente, o Oak-D/Realsense estará conectado para coleta de dados dos usuários, que serão enviados ao servidor, onde é executado o *cérebro*, a aplicação Unity ou Unreal Engine e o sintetizador de voz. Dessa forma as facilidades já implementadas no avatar mantêm-se intactas quanto a sua implementação.

3.1 - WebRTC

WebRTC [9] é uma API open source que viabiliza a transmissão de áudio e vídeo em tempo real via navegadores, permite estabelecer conexões peer-to-peer entre 2 ou mais navegadores que poderão então trocar dados sem a necessidade de uma aplicação nativa ou servidor de terceiros.

Para estabelecer a conexão, o primeiro par instancia um objeto Offer SDP [9] (Session Description Protocol), um meta-dado que descreve o formato do conteúdo da conexão, e o envia a um servidor intermediário; o segundo par o procura e cria um objeto Answer SDP para enviar ao servidor e ser lido pelo primeiro. Esse processo, conhecido como signalling, permite que os participantes estabeleçam conexão sem que o servidor acesse o conteúdo da transmissão.

3.1.1. Servidor STUN

Entretanto, para contornar firewalls e mudanças de endereços de IP (NAT), é utilizado o padrão ICE server [10] (Interactive Connectivity Establishment). Ambos os pares geram uma lista de ICE Candidates, que contém endereços de IP e Port. A WebRTC implementa esse processo via requisições ao servidor STUN [11], um

protocolo para descobrir endereços públicos e determinar/mapear restrições em redes que podem impedir conexões P2P. A figura 2 apresenta um exemplo de conexão P2P usando o servidor STUN.

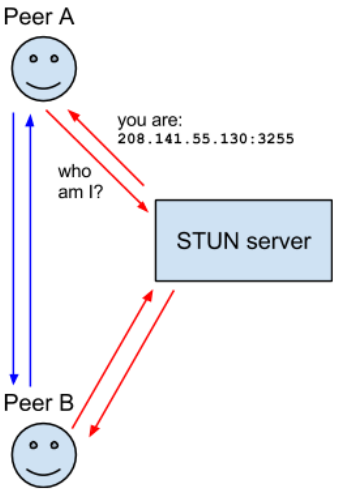

Figura 2: Comunicação peer-to-peer usando servidor STUN

3.1.2. Servidor TURN

Alternativamente, pode-se utilizar um servidor TURN [11] (Traversal Using Relays around NAT), que retransmite todas as informações que a ele chegarem. Dentro da Unity Render Streaming [12], podemos configurar o envio dos dados da Unity para o computador cliente usando um servidor TURN, que atende à restrição da comunicação ser realizada usando o protocolo de rede TCP. A implementação desse servidor é feita utilizando o projeto open source Coturn [13], e seu comportamento é de retransmissão de dados recebidos (Server Relay). A figura 3 apresenta um exemplo de conexão P2P usando o servidor TURN.

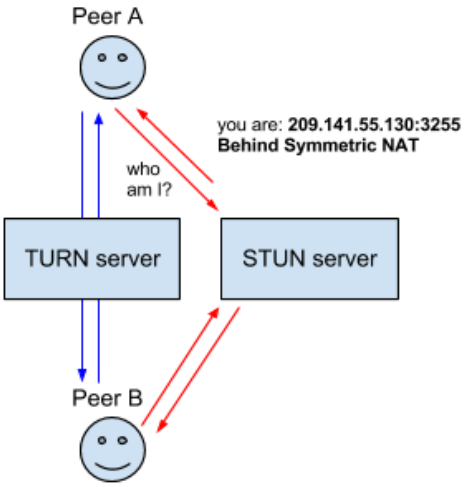

Figura 3: Comunicação peer-to-peer usando servidor TURN

3.2 - Unity: Comunicação servidor-cliente

Para realizar a comunicação servidor-cliente, foi escolhida a extensão Unity Render Streaming. Esse pacote, que ainda se encontra em preview até a escrita deste artigo, possibilita o envio de vídeo e áudio para o servidor cliente usando o protocolo de comunicação WebRTC, implementado como pacote Unity.

3.2.1. Unity Render Streaming

Esse pacote nos possibilita configurar uma transmissão peer-to-peer [14] (P2P), suportando transmissão de vídeo e áudio renderizados na Unity, e também fornece ao usuário a opção de interagir com a aplicação com o envio de mensagens simples como o aperto de botões ou interação com a UI.

A conexão da rede é estabelecida entre o Unity (servidor) e o Navegador (cliente), o envio dos dados é realizado via protocolo UDP [15] por padrão, mas existe a possibilidade de configurar um servidor TURN e configurar a comunicação para o uso do protocolo TCP. Os componentes da estrutura do sistema e a compatibilidade com navegadores podem ser vistos nas figuras 4 e 5, respectivamente. Os problemas de compatibilidade devem-se ao uso do pacote WebRTC [16] (preview), dependência da Render Streaming.

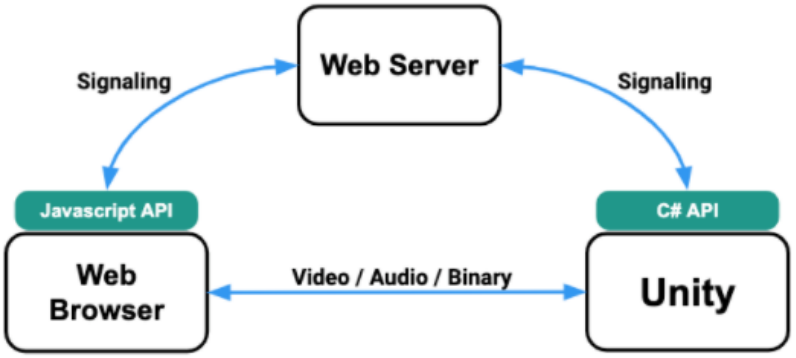

Figura 4: Topologia da rede usada pela extensão Unity Render Streaming

Browser	Windows	Mac	iOS	Android
Google Chrome	✅	✅		✅
Safari		✅	✅	
Firefox	✅			
Microsoft edge (Chromium based)	✅			

Figura 5: Browsers suportados atualmente pelo Unity Render Streaming

3.2.2. Requisitos de Hardware e Software

A implementação da WebRTC na Unity requer mínimos de hardware e software, visto que o funcionamento da API depende do Codec SDK 9.1 da Nvidia. A tabela de compatibilidade é visível na figura 6.

Platform	Graphics API	Hardware Encoder	Software Encoder
Windows x64	DirectX11	✅ (Require NVIDIA Graphics card)	✅
Windows x64	DirectX12	✅ (Require NVIDIA Graphics card)	✅
Windows x64	OpenGL Core		
Windows x64	Vulkan	✅ (Require NVIDIA Graphics card)	✅
Linux x64	OpenGL Core	✅ (Require NVIDIA Graphics card)	✅
Linux x64	Vulkan	✅ (Require NVIDIA Graphics card)	✅
MacOS	Metal	✅	✅
iOS	Metal	✅	✅
Android	Vulkan	✅	✅
Android	OpenGL ES	✅	✅

Figura 6: Suporte a Encoder via hardware e software

3.2.3. Comunicação cliente-servidor

Até o momento, não há suporte ao envio de áudio e vídeo do navegador à Unity via Unity Render Streaming, logo a etapa de transmissão de dados do usuário terá de ser desenvolvida separadamente. Para atingir esse objetivo, o envio dos dados do usuário, captados pelo Oak-D/Realsense vinculado ao computador cliente, pode ser feito usando o protocolo TCP entre o computador cliente e o *cérebro*, executado no computador remoto. Alternativamente é possível também modificar e adicionar funcionalidades à biblioteca Unity e implementar essa comunicação via servidor TURN, dessa forma a completa comunicação entre cliente e servidor seria feita utilizando WebRTC.

3.3 - Unreal Engine: Comunicação servidor-cliente

Para realizar a comunicação servidor-cliente, foi escolhida a extensão Pixel Streaming Plugin [17]. Esse pacote, que está disponível em versão beta apenas para Unreal Engine 4 ou versão superior, possibilita o envio de vídeo e áudio para o servidor cliente usando o protocolo de comunicação WebRTC, implementado internamente pelo motor gráfico.

3.3.1. Unreal Engine Pixel Streaming

Esse pacote nos possibilita configurar uma transmissão peer-to-peer (P2P), suportando transmissão de vídeo e áudio renderizados na Unreal Engine, e também fornece ao usuário a opção de interagir com a aplicação com o envio de mensagens simples como o aperto de botões ou interação com a UI, que pode ser customizada com o uso de tecnologias como Javascript [18] e HTML [19].

A conexão da rede é estabelecida entre a Unreal Engine (servidor) e o Navegador (cliente) usando dois componentes: o Pixel Streaming Plugin e o Signalling and Web

Server. O primeiro é executado internamente na Unreal Engine e é responsável pela codificação/encoding dos frames comprimidos e, juntamente com o áudio da cena, forma um pacote a ser enviado aos navegadores via conexão P2P. O segundo é responsável pelo gerenciamento e estabelecimento das conexões entre o Pixel Streaming Plugin e os navegadores, além de fornecer as aplicações Javascript e HTML aos navegadores para visualização da mídia enviada pelo servidor.

Assim como o pacote unity Render Streaming, a Pixel Streaming realiza o envio dos dados via protocolo UDP mas existe a possibilidade enviar os dados via protocolo TCP ao configurar servidores TURN. Para redes locais não é necessário o uso de serviços STUN e TURN. Caso a conexão cliente-servidor envolva passagem por algum NAT, o uso dos serviços STUN e TURN é necessário. A Figura 7 mostra uma versão simplificada de uma conexão que faz uso do pacote Pixel Streaming.

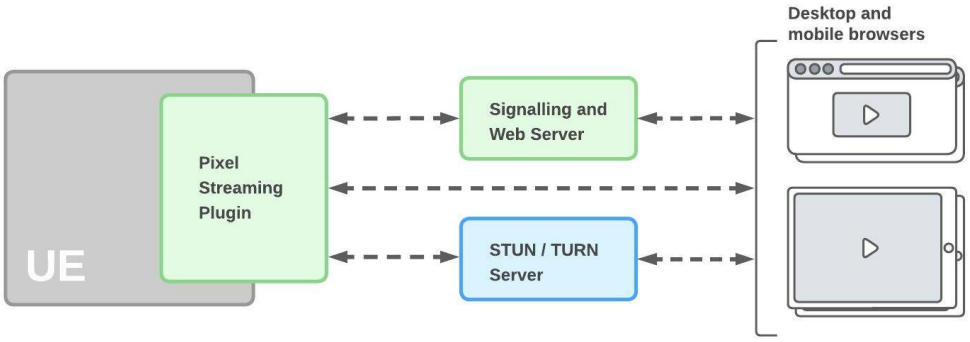

Figura 7: Topologia de comunicação entre cliente e servidor - Pixel Streaming

Ademais, o pacote conta com Selective Forwarding Unit (SFU) [20], feature em fase experimental até a escrita deste artigo. A SFU é um servidor intermediário que deve receber os pacotes de mídia da Unreal Engine e entregá-las aos pares conectados de forma que, considerando a qualidade da rede disponível para uso da aplicação, a resolução de imagem recebida pelo cliente pode variar entre 4k e 720p de modo a reduzir a latência. A Figura 8 mostra a arquitetura de rede entre esses componentes.

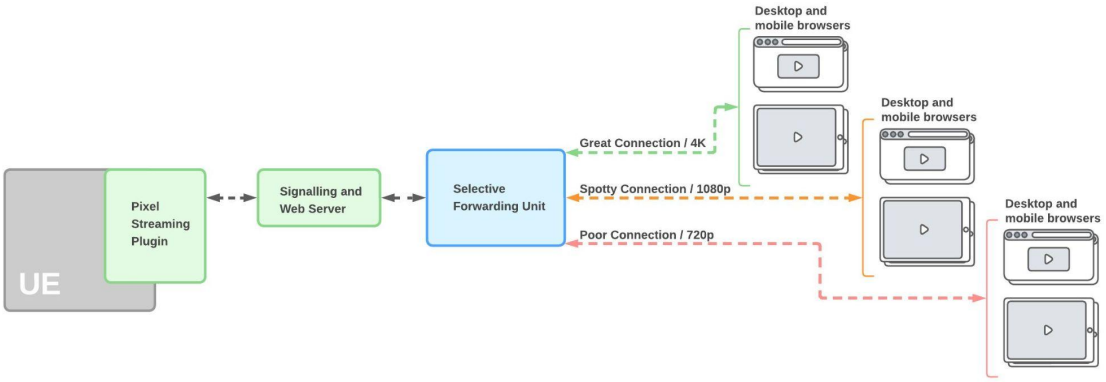

Figura 8: Topologia de Selective Forwarding Unit

3.3.2. Requisitos de Hardware e Software

Os navegadores suportados são Google Chrome (desktop e mobile), Microsoft Edge (desktop), Mozilla Firefox (desktop e mobile) e Apple Safari (desktop e mobile) [21]. O Pixel Streaming Plugin é suportado em plataformas Windows (recomendado na versão 10) e Linux [22] (recomendado nas versões 18.04/20.04). Requisitos de GPU podem ser encontrados nas páginas da AMD [23] e NVIDIA [24] indicadas na seção de referências. Os encoders suportados podem ser conferidos na Figura 9.

Encoder	Acceleration	Encoding Speeds at 1080p/4K	Benefits	Quality at low bitrate	CPU/GPU usage per peer.
H.264	GPU (Nvidia or AMD)	~4.6ms/~15.5ms	Fast encoding/decoding speed. Widely supported across many devices at hardware level.	Blocky	GPU only encodes one session regardless of peer count.
VP8	CPU	~10.5/~25ms	Produces a better image quality at a lower bitrate.	Average	CPU performance scales linearly with peers.
VP9	CPU	~15ms/~50ms	Has highest image quality at lowest bitrates, compared to other encoders.	Good	CPU performance scales linearly with peers.

Figura 9: Encoders/Codificadores compatíveis com Pixel Streaming

3.3.3. Comunicação cliente-servidor

Até o momento, assim como a Unity Render Streaming, não há suporte ao envio de áudio e vídeo do navegador à Unreal Engine via Pixel Streaming, logo a etapa de transmissão de dados do usuário terá de ser desenvolvida separadamente. Para atingir esse objetivo, o envio dos dados do usuário, captados pelo Oak-D/Realsense vinculado ao computador cliente, pode ser feito usando o protocolo TCP entre o computador cliente e o *cérebro*, executado no computador remoto. Alternativamente é possível também modificar e adicionar funcionalidades ao plugin Pixel Streaming e implementar essa comunicação via servidor TURN, dessa forma a completa comunicação entre cliente e servidor seria feita utilizando a estrutura WebRTC.

4. Resultados

O uso do pacote Unity Render Streaming e/ou do Unreal Engine Pixel Streaming e do protocolo WebRTC podem auxiliar na implementação remota do Avatar. A comunicação entre cliente e servidor pode ser feita expandindo funcionalidades da Unity Render Streaming e/ou da Unreal Engine Pixel Streaming para suportar o envio de áudio e vídeo, ou esses dados pré-processados, do cliente para a máquina servidora. Alternativamente, o protocolo TCP pode ser usado para enviar os dados do cliente ao servidor e os pacotes dos motores gráficos seriam usados para enviar os dados do servidor ao cliente. A figura 10 apresenta uma possível topologia para solucionar o problema da renderização remota do Avatar.

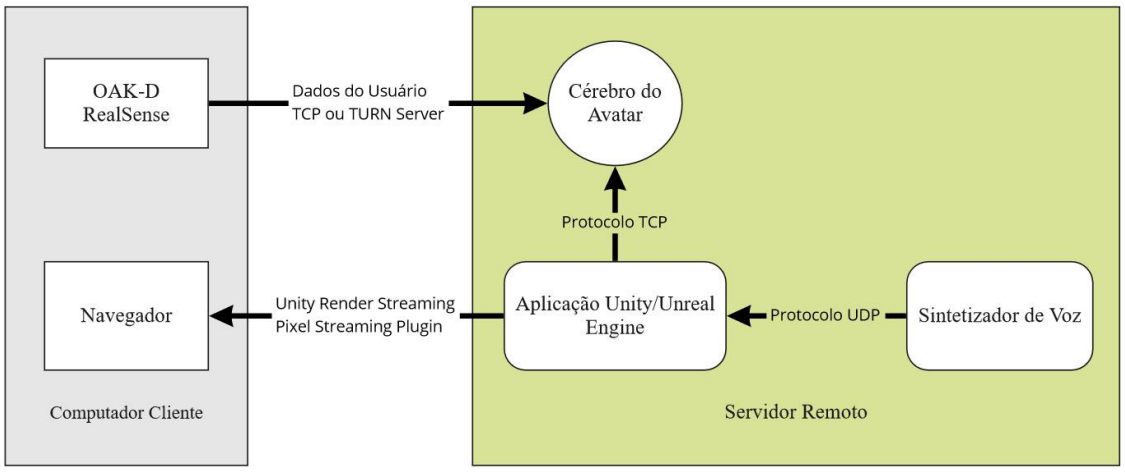

Figura 10: Topologia geral das aplicações remotas do Avatar Recepcionista

5. Conclusão

O pacote Unity Render Streaming e o Unreal Engine Pixel Streaming apresentam alternativas promissoras para a implementação do Avatar Recepcionista em um servidor remoto, o envio dos dados renderizados via streaming usando o protocolo WebRTC viabiliza que o computador cliente tenha apenas as configurações necessárias para suportar tanto a conexão com o Oak-D/Realsense quanto o envio dos dados coletados ao servidor, reduzindo custos de instalação do sistema embarcado atual.

6. Próximos Passos

O pré-processamento dos dados deve ser realizado na máquina que gerar menor latência total para a aplicação, pois quanto menor o tempo de resposta do Avatar melhor será a experiência do usuário. As opções consideradas são: processar dados coletados pelo Oak-D/Realsense no computador cliente e enviá-los ao servidor ou enviar os dados coletados pelo Oak-D/Realsense ao servidor e processá-los no computador servidor.

Além disso, é possível considerar o envio da renderização do Avatar ao computador cliente via servidores STUN, visto que o envio de dados é mais rápido quando comparado a servidores TURN e a latência de transmissão seria reduzida. Essa implementação depende de 3 fatores. O primeiro é se as razões para a restrição do uso do protocolo TCP não apresentam conflito com o uso do servidor STUN. O segundo é se a experiência do usuário é pouco impactada com a eventual perda de pacotes na transmissão servidor-cliente. O último ponto é em especial relevante pois a WebRTC foi construída para facilitar a implementação de serviços de comunicação em tempo real por chamadas de vídeo entre usuários, logo, esses experimentariam potenciais perdas de dados na transmissão do Avatar, semelhante a flutuações da estabilidade de conexão em serviços de comunicação por vídeo, como Skype ou Google Meets. O terceiro é, no caso da comunicação cliente-servidor também ser implementada com esse servidor, se os pacotes perdidos na transmissão levam o *cérebro* a cometer erros críticos na interpretação do áudio recebido e na geração da resposta do Avatar.

O modelo 3D do Avatar recepcionista foi desenvolvido manualmente [3] com o uso de tecnologias como Fuse 3D, Mixano, dentre outras. Uma nova atualização do modelo 3D seria laborioso para quem o fizesse, entretanto, a Unreal Engine 5 conta com uma ferramenta MetaHuman [25], um framework gratuito da própria Unreal Engine que permite geração e customização de aspectos físicos de alta fidelidade de avatares humanos que já contam com esqueletos/rigs e meshes para animações corporais e faciais. A geração do avatar é feita em nuvem e é necessário uso de plataformas Windows ou MAC para tal, contudo, uma vez gerado o modelo, este pode ser exportado para um projeto Unreal Engine em ambiente Linux. Idealmente o modelo gerado atende aos mesmos padrões de qualidade que o desenvolvimento do Avatar atual, feito em ambiente Unity [26][27].

Além disso, como o sistema da Unreal Engine permite a importação de um mesh customizado para integrá-lo a face do avatar gerado [28], é possível avaliar a possibilidade de gerar as expressões faciais da avatar usando um sistema automatizado como o JALI [29], sistema que gera expressões faciais e sincronia labial com suporte multilingue para avatares digitais recebendo áudio e informações como tonalidade e emoção pretendida como parâmetros. Esse sistema foi desenvolvido em parceria com a Universidade de Toronto para o jogo Cyberpunk 2077 [30] da empresa polonesa CD Projekt Red**.** O desenvolvimento de um sistema semelhante especializado na língua portuguesa pode ter muito a acrescentar a qualidade de interação humano-robô que o Avatar Recepcionista promove.

7. Agradecimentos

Agradecemos ao CNPq, pela concessão de uma bolsa do Programa Institucional de Bolsas de Iniciação Científica, ao NRVC/CTI pela oportunidade de realização deste projeto, aos meus Orientadores Artemis Moroni e Josué Ramos.

8. Referências

[1] IEEE/AC sobre interação Humano Robô:

http://humanrobotinteraction.org/category/conference/

[2] Fong, T. W.; Nourbakhsh, I.; Dautenhahn, K. “A Survey of Socially Interactive Robots: Concepts, Design, and Applications”, 2003.

[3] Ito, G. S.; Artemis; Josué; “Unity: Avatar Recepcionista”.

[4] Unity Manual. Disponível em:

<https://docs.unity3d.com/Manual/UnityManual.html>. Acesso em: ago. 2022.

[5] Unreal Engine. Disponível em: <https://www.unrealengine.com/en-US>. Acesso em: Ago. 2022.

[6] Oliveira, G. “A Integração do Sistema de Reconhecimento Facial ao Arcabouço do Robô Recepcionista do CTI”. Disponível em:

<https://drive.google.com/file/d/0B1PZV7lFyXDZUzlFZ0pCSHpQcnZ0c1NQV0Fnc3ZJbC1peG5n/view>. Acesso em: jun. 2022.

[7] Ramos, J. et al. “Informações não-verbais na interação humano-robô aplicado a um robô recepcionista”, 2015.

[8] Introdução à Arquitetura TCP/IP da Internet. Disponível em: <https://www.gta.ufrj.br/grad/03_1/ip-security/paginas/introducao.html>. Acesso em ago. 2020.

[9] Introduction to WebRTC protocols. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API>. Acesso em mar. 2021.

[10] rfc5245 - IETF Tools. Disponível em: <https://datatracker.ietf.org/doc/html/rfc5245>. Acesso em jun. 2021.

[11] What is a STUN/TURN Server?. Disponível em: <https://blog.ivrpowers.com/post/technologies/what-is-stun-turn-server/>. Acesso em jun. 2021.

[12] Unity Render Streaming Manual. Disponível em:

<https://docs.unity3d.com/Packages/com.unity.renderstreaming@3.1/manual/index.html>. Acesso em jul. 2021.

[13] Coturn, Free open source implementation of TURN and STUN Server. Disponível em: <https://github.com/coturn/coturn/blob/master/README.md>. Acesso em: jun. 2021.

[14] Marciano, C.; Assis de Souza, F.; Baptista de Souza, R. “Redes Par-a-Par (Peer to Peer/P2P Networks) GTA - UFRJ”. Disponível em:

<https://www.gta.ufrj.br/ensino/eel878/redes1-2018-1/trabalhos-v1/p2p/arquitetura.html>. Acesso em jul. 2021.

[15] The User Datagram Protocol (UDP). Disponível em: <https://erg.abdn.ac.uk/users/gorry/course/inet-pages/udp.html>. Acesso em jun. 2021.

[16] Unity’s WebRTC Manual. Disponível em: <https://docs.unity3d.com/Packages/com.unity.webrtc@2.4/manual/index.html>. Acesso em jul. 2021.

[17] Unreal Engine Pixel Streaming Plugin. Disponível em: <https://docs.unrealengine.com/5.0/en-US/pixel-streaming-in-unreal-engine/>. Acesso em: Ago. 2022.

[18] JavaScript. Disponível em: <https://www.javascript.com/> Acesso em: Ago. 2022.

[19] HTML. Disponível em: <https://html.com/> Acesso em: Ago. 2022.

[20] Selective Forwarding Unit (SFU). Disponível em: <https://docs.unrealengine.com/5.0/en-US/hosting-and-networking-guide-for-pixel-streaming-in-unreal-engine/> Acesso em: Ago. 2022.

[21] Pixel Streaming Reference. Disponível em: <https://docs.unrealengine.com/5.0/en-US/unreal-engine-pixel-streaming-reference/> Acesso em: Ago. 2022.

[22] Linux Development Requirements. Disponível em: <https://docs.unrealengine.com/5.0/en-US/linux-development-requirements-for-unreal-engine/> Acesso em: Ago. 2022.

[23] Advanced Media Framework SDK. Disponível em: <https://gpuopen.com/advanced-media-framework/> Acesso em: Ago. 2022.

[24] NVIDIA Video Codec SDK. Disponível em: <https://developer.nvidia.com/nvidia-video-codec-sdk#NVENCFeatures> Acesso em: Ago. 2022.

[25] MetaHuman. Disponível em: <https://www.unrealengine.com/en-US/metahuman> Acesso em: Ago. 2022.

[26] Moroni, A.; Ramos, J.; Oliveira, G.; Ito, G. “Modelando a personagem de um recepcionista para brasileiros no ambiente Unity”.

[27] Ramos, J.; Azevedo, H.; Moroni, A.; Trovato, G.; Bernardes, R.; Magossi, S.;, Donizete, V. “Informações não-verbais na interação humano-robô aplicado a um robô recepcionista”.

[28] Custom Mesh to MetaHuman. Disponível em: <https://www.unrealengine.com/en-US/blog/new-release-brings-mesh-to-metahuman-to-unreal-engine-and-much-more> Acesso em: Ago. 2022.

[29] Edwards, P.; Landreth, C.; Fiume, E.; Singh, K. “JALI: An Animator-Centric Viseme Model for Expressive Lip Synchronization”. Disponível em: <http://www.dgp.toronto.edu/~elf/JALISIG16.pdf> Acesso em: Ago. 2022.

[30] Cyberpunk 2077. Disponível em: <https://www.cyberpunk.net/us/pt-br/> Acesso em: Ago. 2022.