

Documentação de um Produto de Software

Projeto AViS – Alloy Virtual Space

LES – Laboratório de Engenharia de Software

Prof. Daniella Franceschinelli Arruda

Pedro Bernardo de SOUSA 0030481711006

Weuller Júnior Souza Bessa 0030481621040

Vítor Andrade Marques da Silva 0030481511040

Sorocaba

2019.2

Sumário

[1. DOCUMENTAÇÃO DE VISÃO DO PROBLEMA: 4](#_Toc19039824)

[1.1 Compreensão do Problema 4](#_Toc19039825)

[1.2 Proposta de Solução de Software 5](#_Toc19039826)

[1.3 Visão Geral dos Pré-Requisitos 6](#_Toc19039827)

[1.3.1 Primeira fase (a implementar até meados 11/2019) 6](#_Toc19039828)

[1.3.2 Segunda fase (implementação em 2020) 7](#_Toc19039829)

[1.4 Glossário 8](#_Toc19039830)

[2. REQUISITOS DO SISTEMA 9](#_Toc19039831)

[2.1. Requisitos Funcionais 9](#_Toc19039832)

[2.1.1 Diagrama de Caso de Uso 9](#_Toc19039833)

[2.1.2 Diagrama de Classes 10](#_Toc19039834)

[2.1.3 Descrição dos Casos de Uso 11](#_Toc19039835)

[2.2. Requisitos Não-Funcionais 18](#_Toc19039836)

[2.2.1 Tempo de resposta 18](#_Toc19039837)

[2.2.2 Uso de memória 18](#_Toc19039838)

[2.2.3 Uso de espaço em disco 18](#_Toc19039839)

[2.2.4 Uso de recursos de processamento no servidor 19](#_Toc19039840)

[2.3. Diagrama de Atividades 19](#_Toc19039841)

[2.4. Protótipo 20](#_Toc19039842)

[2.5. Cronograma 21](#_Toc19039843)

# DOCUMENTAÇÃO DE VISÃO DO PROBLEMA:

## Compreensão do Problema

Na sociedade atual, por diversos fatores como falta de tempo, necessidade de deslocamentos excessivos, disponibilidade de ofertas de cursos e custo elevado no curso desejado, muitos estudantes têm optado pela modalidade de ensino à distância. Nessa realidade, estes alunos encontram diversas plataformas para oferecer a capacitação desejada, porém, essas plataformas não são capazes de oferecer muitas das experiências vivenciadas em uma sala de aula presencial, onde, em um ambiente compartilhado entre alunos e professores, as trocas ocorrem de forma espontânea e natural.

Ainda que em uma sala de aula de ensino presencial a imersão seja mais significativa, há alguns pontos a serem observados, o ritmo individual de um aluno com maior desenvoltura pode ser afetado negativamente pelo ritmo dos demais, além disso, ocorrem interações paralelas nem sempre em uma direção produtiva.

Os sistemas de ensino à distância disponíveis atualmente, não propiciam aos usuários uma experiência de imersão rica o suficiente em um ambiente virtual com os estímulos adequados para o pleno desenvolvimento do aprendizado.

Uma estimulação sensorial mais abrangente tende a oferecer resultados mais significativos no processo de aprendizado. Assim, além da possibilidade da escolha do melhor momento para o aprendizado, em uma localização geográfica favorável ao aluno, usando de seus recursos, para o pleno aproveitamento do sistema EAD, a plataforma deve ser capaz de oferecer uma simulação o mais próxima o possível do que seriam as interações produtivas que ocorreriam em ambientes reais.

Em uma plataforma bem planejada, há ainda a possibilidade de os alunos terem seus respectivos rendimentos de aprendizado e progresso, sendo ditados pelo seu próprio empenho e capacidade. Obviamente que a disciplina de cada indivíduo é um fator determinante. Porém, no sistema EAD, ao contrário do que ocorre no sistema de aulas presenciais, alunos que possam atuar de forma mais independente junto ao material disponível não terão seu ritmo de aprendizado ditados pelas características dos demais.

Essa autonomia buscada a respeito de outros fatores já citados, podem também ser o diferencial para um desenvolvimento mais autodidata. Tudo isso depende da qualidade das experiências que possam ser ofertadas pela plataforma de ensino à distância.

É notável, portanto, uma demanda por ferramentas EAD que supram melhor a necessidade de contato visual, boa qualidade de som, noção de espaço, som direcional etc. Essa necessidade não atendida é claramente observada e documentada a respeito de FLE (Francês Língua Estrangeira) pela escola de francês Pantoufle (<https://pantoufle.online>), que oferece cursos de francês para adultos exclusivamente a distância. Professores e alunos da escola apontam para as rudimentares ferramentas usadas hoje como a principal desvantagem da modalidade EAD na área de ensino de língua, face à modalidade presencial.

## 1.2 Proposta de Solução de Software

A Alloy City Linguistics desenvolveu e mantém uma plataforma de ensino e aprendizado de francês baseada em tecnologias web. O objetivo do grupo AViS até o final do curso ADS dos participantes é desenvolver um cliente para essa plataforma existente. O cliente vai se chamar AViS - Alloy Virtual Space, e vai se apoiar na RESTful API disponibilizada pela Alloy para o projeto. O cliente AViS será capaz de simular um ambiente 3D onde professor e aluno poderão interagir de maneira mais próxima de interações humanas presenciais, isto é, com a impressão de compartilhar o lugar, e não apenas o momento.

Com o propósito de proporcionar uma experiência de aprendizado à distância mais abrangente, a plataforma do projeto AViS, busca fornecer aos usuários ferramentas em que os mais diversos aspectos da comunicação possam ser explorados durante a experiência de uso.

A comunicação humana natural se dá, não apenas através das palavras ditas e escritas. Aspectos da linguagem corporal e da entonação complementam este processo entre emissor e receptor. Assim, oferecer uma ferramenta capaz de abordar, da forma mais abrangente possível tais aspectos no processo de ensino à distância, proverá aos usuários, um significativo avanço na qualidade dos estímulos sensoriais, de forma que uma proporção maior das informações oferecidas seja absorvida.

Dois objetivos iniciais do projeto são, oferecer aos alunos um ambiente virtual em que a experiência de aprendizado aconteça como se aluno e professor partilhassem da mesma localização e oferecer aos professores a possibilidade de receber dos alunos um volume maior de informações para avaliá-los e assim poder aprimorar suas aulas e conteúdo.

Como possibilidades a serem implementadas a longo prazo, o projeto AViS, poderá se tornar uma plataforma unificada capaz de oferecer suporte ao ensino de diversos idiomas e de forma universal, compatível com todos os principais sistemas operacionais.

## 1.3 Visão Geral dos Pré-Requisitos

O projeto compreende duas grandes fases, que vão se estender pelos próximos dois semestres letivos (2019.2 e 2020.1). Dado o caráter exploratório e acadêmico do projeto, os requisitos a seguir podem mudar. Especialmente os da segunda fase.

### 1.3.1 Primeira fase (a implementar até meados 11/2019)

* NAT Punching: o sistema é capaz de enviar pacotes prioritários UDP pela rede, diretamente entre dois clientes. Em nome da performance desejada, certos dados (referentes a VoIP e FoIP, por exemplo) não podem ser enviados primeiro para um servidor remoto para, em seguida, serem retransmitidos para o destino, como se costuma fazer trivialmente. É preciso vencer a “barreira NAT” do usuário conectado, usando o servidor como mediador dos endereços e portas usadas para estabelecer uma conexão direta, entre os clientes de uma sala virtual.
* VoIP (Voice over IP): o sistema transmite diretamente entre clientes streams binários de áudio, com boa qualidade, e reproduz os stream recebidos em áudio binaural, de forma que o usuário note a direção de origem do som, que deve coincidir com a posição do avatar no espaço virtual. O volume de áudio deve variar de acordo com a distância entre os avatars.
* FoIP (Face over IP): o sistema deverá receber o feed da webcam do usuário para produzi um stream de vídeo contínuo, com o rosto do usuário. Esse stream binário deve ser transmitido via UDP a outro cliente. O stream recebido deve ser aplicado ao avatar correspondente, de forma que a expressão facial do usuário seja claramente perceptível a todos os usuários conectados a sala virtual.
* Sala virtual com sincronização da posição dos avatars entre os clientes

### 1.3.2 Segunda fase (implementação em 2020)

Uma vez desenvolvidas as tecnologias chave do projeto, a equipo pode se voltar às funcionalidades mais triviais da solução. A maioria delas já existe em uma aplicação web que se encontra já em produção. Será desafiador e trabalhoso re-implementar uma boa parte da aplicação em C++ e Unreal Engine 4, mas em compensação, a API no backend está pronta, é relativamente madura, e vai exigir pouca adaptação. Segue a lista dessas funcionalidades, de um ponto de vista abrangente/alto.

* Autenticação Stateless baseada em Jason Web Token
* Acesso ao material didático
* Acesso ao calendário de aulas
* Acesso a pacotes de aulas à venda
* Configuração do perfil do usuário
* Acesso a diferentes salas de aula
* Build para diversas plataformas (Windows, Linux, MacOS, Android, iOS, web, Oculus Store, Steam VR)
* Ambientes virtuais diversificados
* Auto-updates
* Acesso a perfil público de outros usuários
* HUD

## 

## 1.4 Glossário

**Student**: usuário com nível de acesso 0;

**Teacher**: usuário com nível de acesso 1;

**Creator**: usuário com nível de acesso 2;

**Coordinator**: usuário com nível de acesso 3;

**Admin**: usuário com nível de acesso 4;

**Resource**: menor unidade do material didático;

**Lesson**: conjunto de Resources;

**Chapter**: conjunto de Lessons;

**Meeting**: par de momentos no tempo, definidos em UTC, que representam uma aula ao vivo;

**Course**: conjunto de Meetings

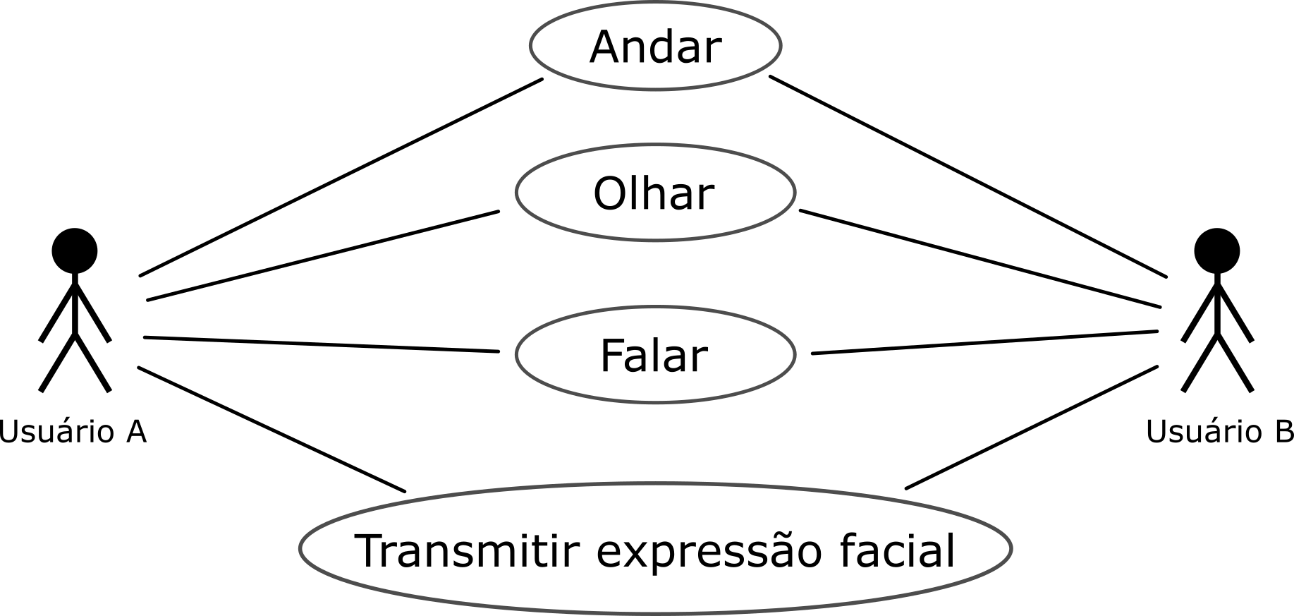
**Pack**: conjunto de Products (exceto outros Packs)

**Product**: unidade comercializável (Meetings, Courses, Lessons, Chapters and Packs);

# 2. REQUISITOS DO SISTEMA

## 2.1. Requisitos Funcionais

### 2.1.1 Diagrama de Caso de Uso



### 2.1.2 Diagrama de Classes

### 2.1.3 Descrição dos Casos de Uso

#### 2.1.3.1 Comunicação

##### 2.1.3.1.1 Ouvir outros usuários presentes

Usuário deve poder ouvir outros usuários que estiverem na mesma sala.

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | FR1: OUVIR OUTROS USUÁRIOS PRESENTES | |
| **Ator Principal** | INSTÂNCIA LOCAL | |
| **Atores Secundários** | INSTÂNCIAS REMOTAS | |
| **Pré-Condição** | 1 – O aluno está dentro da mesma sala de aula virtual que o usuário de origem do som;  2 – O host da instância local está configurado para reproduzir som;  3 – O host da instância remota está configurado para capturar som;  4 – O recurso “*mute audio*” da instância local está desabilitado;  5 – O recurso “*mute mic*” da instância remota está desabilitado. | |
| **Pós-Condição** | O som capturado pelo usuário remoto foi reproduzido claramente pelo equipamento do ALUNO. | |
| **Ações da Instância Local** | | **Ações da Instância Remota** |
|  | | Feed de áudio originado na instância remota é capturado |
|  | | Feed de áudio é compactado |
|  | | Feed de áudio é transmitido diretamente para a instância local via UDP |
| Feed de áudio é recebido | |  |
| Feed de áudio é reproduzido pelo sistema de som | |  |

##### 2.1.3.1.2 Ver outros usuários presentes

Usuário deve poder ver outros usuários que estão na mesma sala. A sequência de ações do sistema abaixo deve ser executada para cada frame capturado pela webcam, numa frequência mínima de 24 hertz.

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | FR2: VER OUTROS USUÁRIOS PRESENTES | |
| **Ator Principal** | INSTÂNCIA LOCAL | |
| **Ator Secundário** | INSTÂNCIA REMOTA | |
| **Pré-Condição** | 1 – O aluno está dentro da mesma sala de aula virtual que o usuário a ser visto;  2 – O host da instância remota está configurado para capturar vídeo. | |
| **Pós-Condição** | 1 – O avatar do usuário remoto tem sua posição no ambiente virtual da instância local sincronizada com a posição escolhida no ambiente virtual da instância remota;  2 – O rosto do avatar do usuário remoto na instância local corresponde à expressão facial do usuário remoto. | |
| **Ações do Ator Principal** | | **Ações do Ator Secundário** |
|  | | Frame capturado pela webcam |
|  | | Rosto do usuário remoto é identificado no frame |
|  | | Frame é reduzido ao rosto (crop) |
|  | | Frame é compactado (ffmpeg) |
|  | | Mapa de bits do rosto do avatar local é atualizado com o frame |
|  | | Frame é enviado diretamente ao peer via UDP |
| Frame é recebido | |  |
| Mapa de bits do rosto do avatar do usuário remoto é atualizado com o frame recebido | |  |

##### 2.1.3.1.3 Falar

Usuário deve poder transmitir discurso pelo sistema. Seu microfone capta o áudio, o sistema comprime o fluxo e o envia diretamente (P2P) aos usuários conectados à mesma sala.

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | FR3: FALAR | |
| **Ator Principal** | INSTÂNCIA LOCAL | |
| **Atores Secundários** | TODAS INSTÂNCIAS REMOTAS NA MESMA SALA | |
| **Pré-Condição** | 1 – O host da instância local está configurado para capturar vídeo;  2 – O recurso “*mute mic*” na instância local não está ativado. | |
| **Pós-Condição** | 1 – Todos os usuários presentes na mesma sala virtual que não tiverem o recurso “*mute audio*” ativado ouvem o áudio capturado pelo host local. | |
| **Ações do Ator Principal** | | **Ações do Atores Secundários** |
| Áudio é capturado pelo microfone do host local | |  |
| Feed de áudio é compactado | |  |
| Feed de áudio é enviado para todos os hosts remoto da mesma sala via UDP | |  |
|  | | Feed de áudio é recebido |
|  | | Feed de áudio é reproduzido pelo host simulando direção e intensidade da fonte de som com base na posição do avatar de origem |

##### 2.1.3.1.4 On/Off Microfone

O Usuário deve poder ativar e desativar seu próprio microfone, através de um botão constantemente visível no HUD. Ao clicar nesse botão, o sistema deve desativar o microfone, o fluxo de áudio e o subsistema de compactação de áudio.

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | FR7: ON/OFF MICROFONE | |
| **Ator Principal** | ALUNO | |
| **Ator Secundário** | PROFESSOR | |
| **Pré-Condição** | 1 – O recurso “*mute mic*” não está bloqueado | |
| **Pós-Condição** | 1 – O microfone no host do aluno está ativado/desativado; | |
| **Ações do Ator** | | **Ações do Sistema** |
| Usuário aciona a função “mute\_mic” (default *M*) | |  |
|  | | Desativar/ativar microfone do host |
|  | | Alterar o ícone do microfone no HUD para microfone com/sem barra |

##### 2.1.3.1.5 Andar

O usuário poderá controlar a posição de seu avatar no ambiente 3D através do teclado, assim como os ângulos de seu ponto de vista através do mouse.

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | FR8: ANDAR | |
| **Ator Principal** | ALUNO | |
| **Ator Secundário** | PROFESSOR | |
| **Pré-Condição** | O usuário deve estar dentro de uma sala virtual | |
| **Pós-Condição** | O avatar do usuário deve estar na posição escolhida | |
| **Ações do Ator** | | **Ações do Sistema** |
| Acionar funções de movimento (defaults *A, D, S, W*) | |  |
|  | | Movimentar o avatar do usuário conforme a função de movimento escolhida |

## 2.2. Requisitos Não-Funcionais

### 2.2.1 Tempo de resposta

Para oferecer as funcionalidades desejadas, é importante que a comunicação entre instâncias cliente seja a menor possível. Não é razoável decidir um valor fixo para o tempo de viagem dos pacotes, já que, num cenário real, este tempo vai depender de uma rede probabilística (a Internet), e da localização geográfica dos usuários conectados pelo sistema.

Entretanto, no quadro da demonstração que se pretende fazer ao final do projeto, espera-se que a latência entre os dois computadores conectados esteja na ordem de dezenas de milissegundos, quanto que a latência entre as instâncias cliente e a instância no servidor remoto, localizado em um *datacenter*, em São Paulo, seja de até 200 milissegundos.

### 2.2.2 Uso de memória

O programa, tanto no servidor quanto no cliente, não deve utilizar mais do que 500 MB de memória principal.

Além disso, dada a liberdade de manipulação de memória oferecida por C++, medidas específicas devem ser tomadas para evitar vazamento de memória(*memory leak*).

### 2.2.3 Uso de espaço em disco

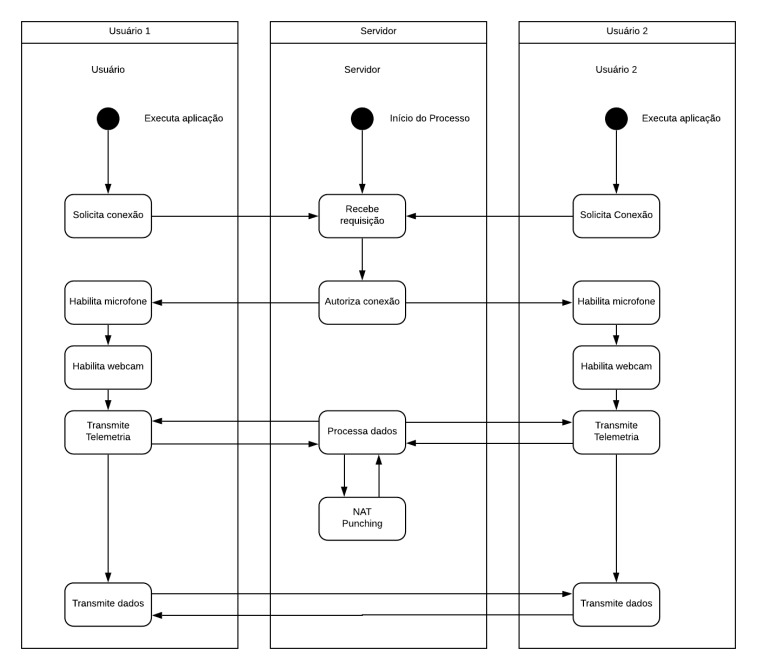
O arquivo de instalação não deve ultrapassar 200 MB.

O espaço ocupado pelo sistema cliente, uma vez instalado, não deve ultrapassar 400 MB.

### 2.2.4 Uso de recursos de processamento no servidor

A instância em execução no servidor não deve ultrapassar 20% dos recursos de processamento da máquina virtual, exceto durante a instanciação do programa.

## 2.3. Diagrama de Atividades



## 2.4. Protótipo







## 2.5. Cronograma

Cronograma do Projeto AViS em 2019.2

* 12/08: Interface com o usuário;

Demonstrar distinção entre cógigo rodando no cliente vs no servidor (UE4)

* 21/08: Introdução;

Diagrama de Casos de Uso;

Diagrama de Classes;

Linux Server

* 28/08: Riscos – Vitor;

Arquitetura da aplicação atual – Bernardo;

Diagrama de Sequência – Weuller;

Interface com o servidor NodeJS da Alloy API (input/output) - Bernardo

* 04/09: Descrição atual – Vitor;

Problemas encontrados – Vitor;

Tecnologias utilizadas e APIs – Bernardo;

Requisitos não funcionais – Bernardo;

Aplicativos disponíveis no projeto – Weuller;

Considerações sobre o Banco de Dados utilizado;

Sincronização de posições dos avatars

* 11/09: Situação proposta – Vitor:

Descrição dos casos de uso;

Componentes do SW - Weuller | Bernardo;

Integrar OpenCV;

Acesso ao disco (input/output)

* 18/09: Limitações operacionais – Vitor;

Considerações de hardware e software – Bernardo;

Diagrama de Pacote – Weuller;

Crop + resize imagem de exemplo em disco;

* 25/09: Considerações legais;

Políticas organizacionais;

Diagrama de Estado – Weuller;

Embasamento teórico;

Identificar coordenadas de rosto em imagens em disco

* 02/10: Demonstrar acesso à webcam;

Demonstrar acesso ao microfone;

* 09/10: Demonstrar NAT Punching
* 16/10: Transmitir/receber streams binários entre clientes via UDP;
* 23/10: Descrição da tecnica utilisada para levantamento dos requisitos;

Gerar buffer de vídeo com rosto do usuário;

Enviar buffer de vídeo P2P via UDP

* 30/10: Projeto de Teste;

Instalação do Software;

Análise dos Resultados;

Conclusão;

Referências;

Anexos;

Glossário;

Apêndice;

Receber buffer de vídeo FoIP;

Decodificar buffer de vídeo em frames FoIP;

Aplicar frames FoIP ao avatar correspondente;

* 06/11: Gerar buffer de áudio compactado;

Enviar buffer de áudio P2P via UDP

* 13/11: Receber buffer de áudio VoIP;

Reproduzir audio recebido;

Conclusão da implementação

# Implementação

O código fonte do projeto está integralmente disponível no GitHub, assim como *releases* pré-compiladas, no seguinte endereço: <https://github.com/alloy-city/AViS>

Para implementar o protótipo de demonstração, ou MVP, foi necessário estudar as partes relevantes da vasta documentação de Unreal Engine 4. Essa documentação cobre quase toda a API exposta pelo mecanismo de jogo. Eventualmente, nos deparamos com métodos e atributos, as vezes até mesmo classes inteiras, que não são mencionadas. Nesses casos raros, é sempre possível consultar o código, diretamente, que, embora não seja totalmente aberto, é acessível para consulta. Mesmo quando a documentação cobre o recurso que estamos utilizando, a maneira mais efetiva se assegurar que compreendemos como UE4 funciona é através do código fonte. O programa AViS é, portanto, desenvolvido com ao lado do código do mecanismo UE4, em uma única solução no Visual Studio.

A modelagem do ambiente virtual, que representa uma sala de aula, foi feita no próprio Editor UE4, utilizando modelos geométricos simples, disponíveis em qualquer instalação recente do mecanismo. Para essa tarefa da implementação, foi necessário estudar como UE4 simula geometria 3D, iluminação, texturas e materiais.

Em seguida, implementou-se um avatar, controlável pelo usuário, com base na classe *Pawn* exposta pelo mecanismo de jogo. Essa classe conta com um ponto de vista na sala virtual, que pode ser manipulado pelos mecanismos de entrada de comandos, como mouse e teclado.

Para que a esfera, que representa a cabeça do avatar, seja devidamente capaz de comportar o mapa de bits correspondente a um quadro de rosto, recebido pela rede, foi necessário realizar um procedimento chamado *UV unwrapping*. Este procedimento associa coordenadas de renderização de texturas a um modelo 3D. Neste caso em particular, o modelo 3D é uma esfera, mas as texturas são mapas de bits quadrados, de 64 por 64 pontos. Foi preciso portanto definir as coordenadas das texturas na esfera de forma que o rosto sofra a menor deformação geométrica possível. Esta tarefa foi realizada no software de modelagem 3D Blender.

Para ter acesso à webcam, integramos ao projeto o software de visão de máquina OpenCV. O primeiro desafio dessa tarefa foi aprender a utilizar as funcionalidades mais elementares do OpenCV, isoladamente. No nosso caso, escolhemos capturar o vídeo da webcam e apresentá-lo, sem modificação alguma, de volta ao usuário, em uma janela. Em seguida, ao tentar integrar o OpenCV a um projeto baseado em UE4, descobrimos algumas colisões entre as duas ferramentas. Uma função OpenCV tem o mesmo nome de uma classe da biblioteca de funções matemáticas chamada Kismet Math. Em iterações futuras do projeto, conflitos como este poderão ser evitados através do mecanismo de subsistemas de UE4. Este mecanismo é usado internamente para estruturar os diferentes componentes UE4. É possível utilizar essa mesma estrutura para integrar OpenCV ao projeto, criando para ele um espaço de memória privado, diferente do espaço global. Essa solução, entretanto, requer a reestruturação de uma parte substancial da implementação. Optou-se por renomear a função no código fonte do OpenCV e utilizar essa versão alterada do programa. É preciso manter em vista essa modificação e implementar uma solução definitiva antes de tentar atualizar OpenCV para uma versão mais recente.

Uma vez que OpenCV e UE4 estavam trabalhando juntos, em um mesmo projeto, implementou-se, um a um, os métodos responsáveis por tratar o fluxo de vídeo no cliente que o captura. A saber:

1. Captura-se um quadro
2. Identifica-se as coordenadas e o raio de um rosto no Quadro
3. Recorta-se o Quadro utilizando as coordenadas encontradas
4. Redimenciona-se o rosto para 64 por 64 pontos
5. Compacta-se o rosto em JPG com índice de qualidade 80
6. Envia-se o rosto compactado ao objeto responsável por comunicação em rede.

Este processo foi implementado na classe Webcam

A classe StreamService é responsável por enviar os quadros processados pela rede. Essa classe foi projetada para processar 30 quadros por segundo, o que a classifica como ponto de estrangulamento de performance em potencial. Ela deve iniciar um loop de observação, para aguardar conexões de rede. Para que o programa não pare até que uma conexão seja estabelecida, o loop de observação é executado em um thread paralelo ao programa principal.

Os bits correspondentes ao rosto, já compactado, são enviados à máquina remota. Nela, uma função exposta pela API UE4 chamada ImportBufferAsTexture2D. Essa função recebe um buffer correspondente a uma imagem, expande a imagem caso ela esteja compactada, e cria um objeto da classe nativa Texture2D. A cada quadro recebido pela rede, essa função é chamada. Se ela retornar um objeto da classe Texture2D válido, aplica-se a textura ao avatar correspondente ao usuário que gerou o quadro em questão.

Todo esse processo cria a impressão de que um determinado avatar, presente na sala de aula virtual, tem o rosto do usuário que o controla.

Finalmente, para implementar a funcionalidade de transmissão de voz, foi necessário aprender a utilizar os recursos de processamento de áudio do mecanismo de jogo. O primeiro passo é ganhar acesso ao microfone do usuário através da ativação do plugin Voice, desenvolvido também pela Epic Games. Ao assegurar que os usuários estão em uma mesma sessão (USession), UE4 se encarrega de enviar os pacotes de áudio compactados, para todos os usuários conectados à mesma sessão. Entretanto, para aumentar a sensação de presença, alteramos o comportamento do mecanismo para que a voz, no usuário remoto, seja reproduzida a partir de um ponto específico na sala de aula virtual, e não como um som ambiente independente da simulação. Para isso, utilizamos a classe nativa USoundCue, que pode ser posicionada numa cena virtual e emitir sons. Os sons emitidos por um objeto dessa classe contam com diversos recursos de processamento de som em tempo de execução, entre eles atenuação diferenciada entre os dois canais. Notemos que UE4 reproduz os sons do programa em execução em dois canais. Esquerdo e direito. Ao atenuar por exemplo o lado direito mais do que o esquerdo ao reproduzir um determinado som, o usuário tem a sensação de que o som está vindo do seu lado esquerdo.

Para usufruir dos recursos de atenuação diferenciada oferecidos pela classe USoundCue, instancia-se um objeto dessa classe juntamente com o avatar, mantendo os atributos de coordenadas espaciais do objeto sincronizados com as coordenadas da cabeça do avatar. Durante a instanciação do avatar e do seu componente de som, definimos uma relação de parentalidade espacial entre os dois objetos. Dessa forma, os cálculos dessa sincronização são delegados ao mecanismo de jogo.

Alcançou-se todos esses passos de implementação na versão 0.5 do cliente AViS. Tanto o código fonte quanto um arquivo compactado contendo todos os arquivos necessários para a demonstração da tecnologia estão disponíveis no repositório do projeto.