GlarAssist

Mestrado em Engenharia Informática - Computação Móvel

Tiago José Rino Oliveira

Leiria, março de 2022

GlarAssist

Mestrado em Engenharia Informática - Computação Móvel

Tiago José Rino Oliveira

Trabalho de Projeto realizado sob a orientação do Professor Nuno Carlos Sousa Rodrigues

Leiria, março de 2022

Originalidade e Direitos de Autor

O presente relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Curso de Mestrado em Computação Móvel, no ano letivo 2020/2021, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos.

Agradecimentos

Foi uma viagem longa, o Covid fez-me lembrar que a vida é frágil, que não podemos tomar nada como garantido e o isolamento deixa-nos um pouco moles, por isso, se estiver a ler isto, gostaria de agradecer à minha família por me lembrar do trabalho que ainda tenho até atingir os meus objetivos e apoio incansável, ao meu *personal trainer* que, além de saber como reajo a tristes verdades, as diz na minha cara, os meus amigos e colegas de trabalho que se preocupam mais comigo do que eu, algo que ainda não consigo perceber, complementando a minha personalidade passiva ao serem assertivos por mim, os meus amigos brasileiros por me permitirem fazer parte de um grupo com uma mentalidade não planeada que me fez descobrir que para ter um grande momento e uma boa história só precisa da vontade e do pessoal, a minha camarada que estava lá incansavelmente com um ouvido aberto quando era preciso, e a minha amiga, que partilhou a sua experiência com óculos de realidade virtual, mencionada mais abaixo.

Gostaria de agradecer aos colaboradores da GlareVision por me permitirem estudar e trabalhar com horários de trabalho tão flexíveis para me permitir crescer o meu conhecimento.

Gostaria de agradecer aos meus professores, por me darem os conhecimentos necessários para chegar a este nível técnico e ao orientador do projeto por assegurar que o relatório está num estado adequado, e pelo input no projeto.

Gostaria de lhe agradecer a si por ler o meu relatório. Espero poder partilhar um pouco da minha experiência neste campo e que aprenda alguma coisa, pois eu sei que aprendi.

Resumo

Devido à pandemia Covid-19, as empresas depararam-se com uma necessidade de operar remotamente, este relatório descreve o desenvolvimento do GlarAssist, uma aplicação de assistência remota que permite o desenho conjunto usando as tecnologias de realidade aumentada e chamada de vídeo.

Este documento fornece o fundamento teórico para realidade e virtualidade informatizadas, introduz estudos de presença, e analisa brevemente a tendência dos trabalhos da comunidade científica em termos de indústrias em tecnologias virtuais.

Neste relatório é apresentada também a metodologia de desenvolvimento ágil implementada, descreve como SCRUM foi utilizado, e como os cartões de Kanban ajudaram a organizar uma equipa em expansão.

O relatório menciona ainda como as tecnologias WebRTC, ARCore e OpenGL foram usadas para fazer uma aplicação de assistência remota usando realidade aumentada, os algoritmos usados ou criados, bem como os desafios ultrapassados para tornar a aplicação possível.

Como resultado foi possível criar uma solução que disponibiliza assistência remota com anotações, compatível com aproximadamente 98% dos dispositivos Android desde que possuam uma camara e ligação à Internet.

Palavras-chave: assistência remota, GlarAssist, realidade aumentada.

Abstract

Mainly developed during the Covid-19 pandemic, companies found the need to operate remotely, this report describes the development of GlarAssist, a remote assistance application that allows the assistant to annotate on the assisted side remotely joining Augmented Reality and Video call and allowing both sides drawing capabilities.

This document provides the theoretical background for computerized reality and virtuality, introduces existing presence studies, and briefly analyses the scientific community papers tendency in terms of industries on virtual technologies.

In this report the Agile development methodology deployed is also introduced, describes how SCRUM was used, and how Kanban boards helped organize a scaling team.

The report also mentions how technologies like WebRTC, ARCore and OpenGL rendering were used to make an augmented reality remote assistance application, the algorithms found or created, as well as the challenges overcame to make the application possible.

As a result it was created a solution that provides remote assistance to 98% of the Android devices, only requiring a camera and an internet connection.

Keywords: remote assistance, GlarAssist, augmented reality.

Índice

Lista de Imagens

Lista de Tabelas

Lista de Anexos

Lista de acrónimos

API	Application Programming Interface
AR	Augmented Reality
ARGB	Alpha Red Green Blue
ARM	Advanced RISC Machine
AV	Augmented Virtuality
CD	Continuous Delivery
CEO	Chief Executive Officer
CI	Continuous Integration
DOM	Document Object Model
GL	Graphic Library
IOS	Iphone Operating System
IP	Internet Protocol
MR	Mixed Reality
NAT	Network Address Translation
OCR	Optical Character Recognition
R&D	Research and Development
REST	Representation State Protocol
RFC	Request for Comments
RTC	Real Time Communication
SDK	Software Development Kit
SSQ	Simulator Sickness Questionnaire
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TURN	Traversal Using Relays around NAT
UDP	User Datagram Protocol
UHF	Ultra HightFrequency
US	United States
VBO	Vertex Buffer Object
VCS	Version Control System
VHF	Very High Frequency
VR	Virtual Reality

1. Introdução

Este projeto foi desenvolvido durante a pandemia Covid-19, que nos fez perceber que não estávamos prontos para ser isolados, criando uma necessidade crescente de alternativas remotas que nos permitam, enquanto indivíduos, prosseguir com as nossas vidas mesmo isoladamente.

O projeto detalha a criação, a investigação e todos os desafios que foram ultrapassados para desenvolver uma solução que fornece assistência remota para a indústria 4.0 e, por extensão a qualquer pessoa com acesso a um Smartphone ou óculos inteligentes.

A solução foi denominada GlarAssist, um projeto que visa fornecer aos seus clientes um meio remoto para ajudar utilizando Realidade Aumentada para interagir com o ambiente.

O projeto foi realizado no âmbito da empresa Glartek onde fiz parte da equipa como programador da Aplicação Android.

Este relatório divide-se em dois capítulos principais, o quadro teórico, onde descreve as bases teóricas do projeto, mais especificamente, a renderização dos desenhos conjuntos e a transmissão de vídeo. O segundo capítulo refere-se ao desenvolvimento do projeto, à análise da concorrência, assim como o desenvolvimento foi organizado, bem como descreve as tecnologias e metodologias utilizadas para resolver os problemas encontrados. No fim integra a conclusão onde é feita uma síntese do trabalho assim como é descrito o trabalho futuro planeado.

2. Enquadramento teórico

Este capítulo tem como objetivo contextualizar as tecnologias relacionadas com este projeto e introduzir conceitos relacionados com o produto aqui apresentado, começando por caracterizar realidade e virtualidade, seguido de uma breve introdução sobre distintos tipos de presença.

2.1. Realidade vs. Virtualidade, a perspetiva de um computador

O que separa a realidade de um mundo gerado por computador? Em 2018, num espetáculo de Joe Rogan, Elon Musk um empreendedor e CEO de várias empresas tecnológicas, como por exemplo, a SpaceX, uma empresa de foguetes e naves espaciais, e a Tesla, uma empresa de carros elétricos, disse: "Assumindo qualquer taxa de progresso, então jogos serão indistinguíveis da realidade (...), portanto estamos provavelmente a viver numa simulação" (Wall, 2018), levantando as questões, como realidade e virtualidade se separam? e se diferentes tipos de realidade podem ser categorizados a partir da introdução de diferentes quantidades de realidade ou virtualidade? para ambas as perguntas, a resposta está resumida na Figura 1**Erro! A origem da referência não foi encontrada.**, um diagrama que representa as diferentes realidades que são atualmente possíveis de simular através de um dispositivo computadorizado. Estas representações de realidade baseiam-se no trabalho de (Farshid et al., 2018), mas foram reordenadas pela quantidade de realidade/virtualidade presente.

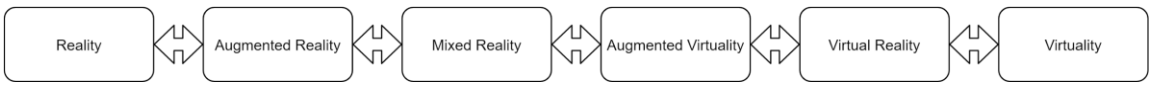

Figura 1 nomenclatura para diferentes representações de realidade tendo em conta realidade vs. virtualidade baseado no trabalho de (Farshid et al., 2018)

Tomando em conta a primeira pergunta de distinguir realidade de virtualidade, tem-se que da mesma forma que o mundo real se relaciona ao mundo material que nos rodeia, a virtualidade está relacionada a um mundo possível.

Quanto à questão da distinção de realidades por introdução de diferentes níveis de realidade / virtualidade, precisamos de explorar as diferentes categorias de realidade nos próximos subcapítulos, a começar por Realidade Aumentada.

2.1.1. Realidade Aumentada (AR)

De acordo com (Carmigniani et al., 2011) a *Augmented Reality* (AR) melhora a perceção e interação do utilizador com o mundo real, através da sobreposição de objetos virtuais no mundo real, utilizando maioritariamente o sentido visual, mas também abrange os outros sentidos como o cheiro, o toque e a audição.

A primeira referência registada a um dispositivo AR foi em (Baum, 1996), que descreve um dispositivo chamado "*Character Maker"* em que consiste num par de óculos que, durante o uso, mostra uma letra em cima da pessoa que classifica o seu caráter.

A representação de AR em dispositivos computorizados pode ser conseguida através de algumas técnicas como por exemplo, através da projeção de objetos virtuais no mundo, usando ecrãs translúcidos, ou através da renderização de objetos virtuais em cima de um fluxo de vídeo numa tela. Alguns exemplos de dispositivos baseados em projeção são lentes de contacto e óculos. Quanto a dispositivos de renderização ao vivo alguns exemplos são *smartphones* e *heads-up display*.

Os dispositivos de AR utilizam os seus sensores para determinar a sua posição e orientação em relação ao mundo, permitindo estabelecer a posição dos objetos virtuais na posição adequada (Kiyokawa, 2012; Özacar et al., 2016), um exemplo de dispositivo de renderização ao vivo está representado na Figura 2.

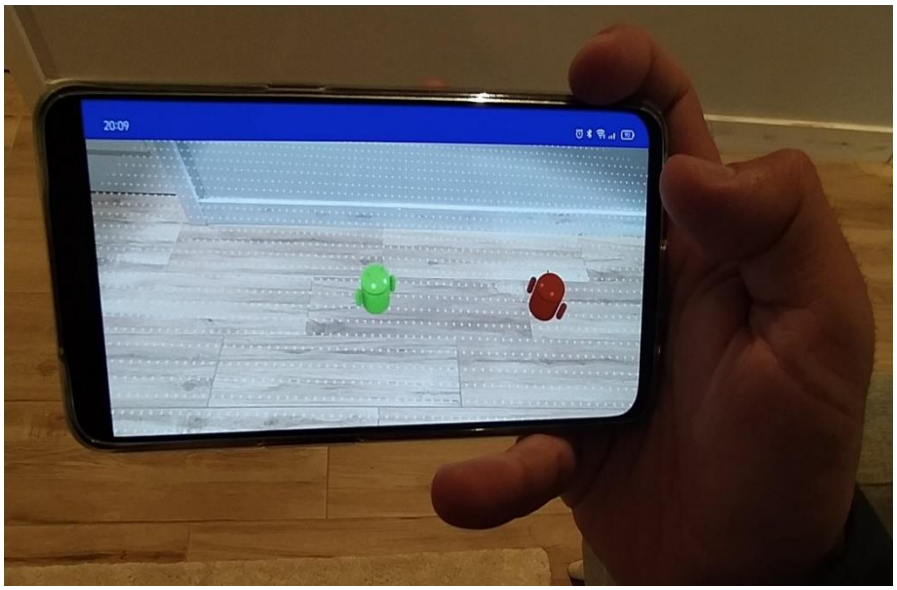

Figura 2 Representação de dispositivo compatível com AR com anotações desenhadas em cima do fluxo de vídeo

Em relação à aplicabilidade da tecnologia, um dos jogos de AR mais populares até à data é o “Pokémon Go”, onde os jogadores apanham e combatem Pokémon, um jogo similar ao *geocaching*, mas em vez da interação dos jogadores ser de visitar os locais separadamente, os jogadores são representados pelos seus avatares e interagem com outros jogadores cara a cara (Rauschnabel et al., 2017; Jovem, 2020).

2.1.2. Virtualidade Aumentada (AV)

A *Augmented Virtuality* (AV) é o oposto da AR onde, em vez de introduzir objetos virtuais no mundo real, introduz objetos do mundo real no mundo virtual ou, essencialmente, "um ambiente virtual que também integra componentes reais" (Neges et al., 2018).

A primeira experiência documentada de AV foi o *sistema "*Videoplace” que, em 1970 trouxe o conceito de um ambiente gráfico gerado por computador onde o participante se vê projetado nele. Isto faz com que o utilizador estando sozinho seja projetado em conjunto com outras pessoas ou imagens de criaturas, dando a possibilidade de interagir com essas imagens (Krueger et al., 1985).

Uma aplicação recente que usa AV é o “Behand”, um conceito interessante que consiste na utilização de uma câmara para capturar e compreender a posição da mão do utilizador através da imagem. Esta informação de posição da mão é então usada para interpretar os gestos da mesma e usada para interagir com objetos virtuais tridimensionais (Caballero et al., 2010).

2.1.3. Realidade Virtual (VR)

Virtual Reality (VR) é uma realidade 3D imersiva gerada por computador que, com a ajuda de alguns dispositivos, permite ao utilizador sentir-se dentro de um mundo virtual. O conceito pretende criar uma experiência em que o utilizador perde a perceção da tecnologia e vê a experiência como real.

Durante uma sessão de VR, devido ao utilizador se sentir presente, o mesmo tem a possibilidade de interagir no espaço virtual como na vida real e sentir a experiência como um lugar visitado em vez de uma experiência fotográfica (Serrano et al., 2013; Zuniga Gonzalez et al., 2021).

Uma das primeiras menções ao VR foi por (Baltrušaitis, 1977) quando descreve a perspetiva e a ideia de criar uma expansão infinita num pequeno espaço, descrito como "a multiplicação de mundos artificiais".

Uma das primeiras aplicações VR foi o “Sensorama Simulator”, representado na Figura 3, o aparato é descrito na forma de uma patente dos EUA descrevendo um dispositivo que estimula os sentidos dos utilizadores para simular uma experiência. O dispositivo consiste numa caixa à volta da cabeça do utilizador e aborda 4 dos 5 sentidos humanos, nomeadamente visuais através da projeção no invólucro para a cabeça, som utilizando som binaural, cheiro usando uma substância estimulante do odor e, táctil induzindo pequenas vibrações ou choques para simular movimentos e impactos (Heilig, 1962).

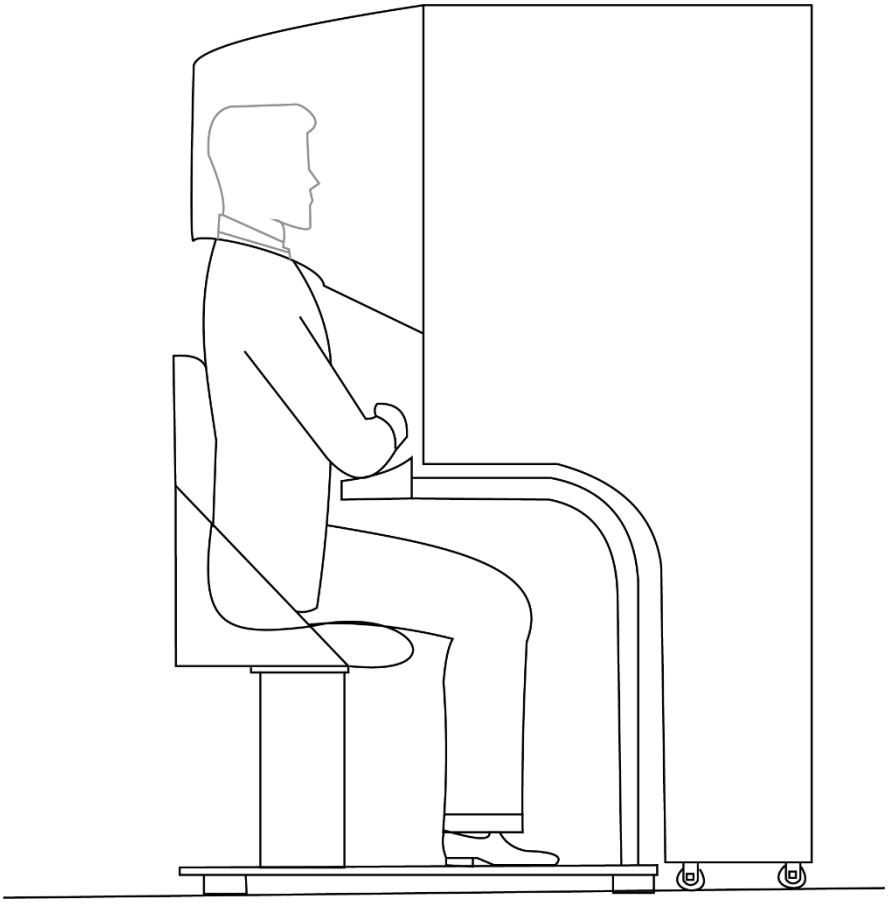

Figura 3 Representação da vista lateral do aparelho Sensorama

No decorrer da pesquisa deste tema, um tema recorrente do uso de VR é a causa de náuseas por uso prolongado.

Um método de quantificar a doença é o uso do Questionário de Simulação à Doença ou Simulator Sickness Questionnaire (SSQ), que quantifica os sintomas numa escala de 0-3,

nas categorias de náusea, oculomotor e desorientação, permitindo um resultado quantitativo de uma doença (Kennedy et al., 1993; Walter et al., 2019).

Tendo em conta um estudo de 2016 sobre o dispositivo VR "Oculus Rift", usando SSQ através de dois experimentos que expôs as pessoas a 15 minutos de jogos com o dispositivo montado na cabeça, o resultado da experiência foi que, 56% das 36 pessoas, 18 homens e 18 mulheres sofreram de enjoo, sendo 77,78% do sexo feminino (Munafo et al., 2017).

De acordo com um estudo de Eletroencefalograma com objetivo de analisar uma estrada curva virtual de 40 minutos projetada numa parede à volta de um carro parado concluiu que os indivíduos que estavam sentados no carro atrás da projeção receberam as mesmas perceções que aqueles num ambiente real (C.-T. Lin et al., 2007).

O enjoo por movimento pode ser descrito como inconsistências entre o que o utilizador sente e o que vê, o enjoo de movimento resultante de um ambiente VR corresponde a algo que é visto, mas não sentido, isto inclui movimento visualmente percebido na ausência de movimento real (Kim et al., 2018).

De acordo com um estudo exploratório (Patrão et al., 2020), sobre como lidar com o enjoo do movimento em VR, foi concluído que existem três principais indutores de enjoo:

- Alterar a orientação sem o sinal do utilizador.
- Modificação do campo de visão ou taxa de *zoom*.
- Variação da aceleração ou da velocidade.

Sugere ainda três metodologias principais sobre o desenvolvimento de VR para reduzir os sintomas:

- A imersão deve ser mantida.
- Evitar animações de câmaras.
- Adicionar um quadro de referência.

2.1.4. Realidade Mista (MR)

Segundo (Speicher et al., 2019), a definição de *Mixed Reality* (MR) é por si só uma discordância de especialistas, e como conclusão do seu estudo chegou a 3 definições distintas:

- MR é um superconjunto de AR, uma vez que é a composição de objetos reais e virtuais num ecrã.
- O MR é uma versão alargada da AR onde o ambiente à sua volta pode ser compreendido e manipulado.
- MR é um termo de marketing e um sinónimo de AR.

Um exemplo de MR foi apresentado na conferência Xbox na feira E3, onde o jogo Minecraft foi mostrado a correr no Microsoft HoloLens, um dispositivo capaz de MR. Nesta demonstração foi integrado um mundo virtual em torno do utilizador, representado na Figura 4 (Miranda Sanchez, 2015).

Figura 4 Xbox E3 2015 Minecraft usando Realidade Mista em HoloLens

2.1.5. Tendências

De acordo com a pesquisa de (Muñoz-Saavedra et al., 2020), que analisou publicações desde o ano de 2000, filtradas com as seguintes restrições:

- Tópicos de VR e AR,
- centrando-se nos Estados Unidos da América e União Europeia,
- sobre os temas da pesquisa, saúde, educação e indústria.

Após a filtragem, quantificação e agregação das publicações, os investigadores chegaram às seguintes conclusões:

- pesquisa é o campo mais referenciado,
- a saúde é o campo de aplicação mais popular,
- as publicações são mais frequentes no nível escolar do ensino secundário,
- a manutenção é o campo de estudo mais popular nesta indústria tecnológica.

Na Figura 5 encontra-se uma representação gráfica baseada nos resultados deste estudo, demonstrando o número de publicações de AR e VR por tema, no eixo y, por cada ano, no eixo x.

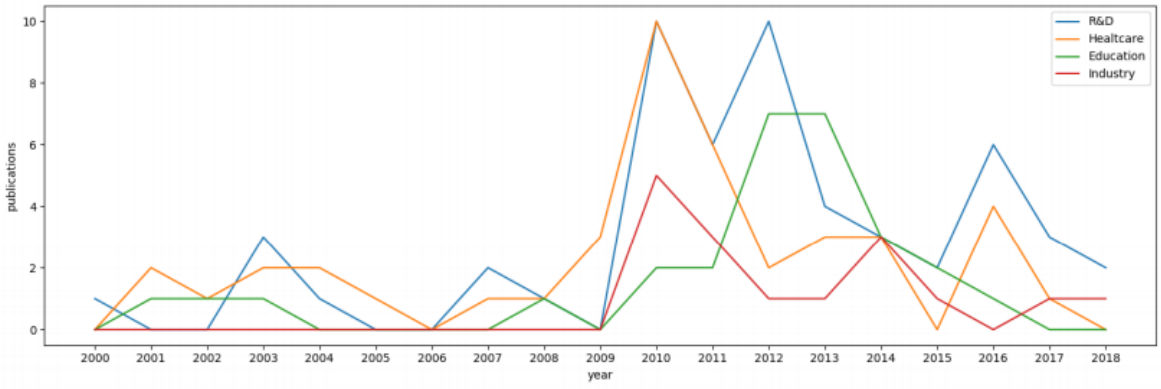

Figura 5 Publicações AR VR por tópico agrupado por ano

2.2. Presença virtual e as semelhanças com a realidade

De acordo com (Schloerb, 1995), a presença física é a existência de algo numa região no espaço e no tempo e ninguém pode estar fisicamente presente numa localização separada espacialmente. Se o operador conseguir completar objetivamente uma tarefa com sucesso, então é chamada de presença objetiva. A perceção física de estar presente é designada por presença subjetiva.

Outra forma de presença é a telepresença, que representa estar objetivamente presente num lugar fisicamente separado. Consequentemente, a telepresença subjetiva é a manipulação sensorial do operador remoto para tornar a telepresença indistinguível da presença física.

Um subtipo de telepresença subjetiva é a presença virtual, onde o operador interage com um ambiente remoto, e um exemplo de presença virtual é um dispositivo que reproduz os movimentos das mãos do operador num braço robô remoto, e o vídeo do braço robô é

reproduzido numa sala escura posicionado de forma estratégica em frente ao operador, tornando o operador incapaz de distinguir entre o seu braço e o braço robótico.

2.2.1. Videoconferência

Esta tecnologia foi introduzida em 1964 pela AT&T sob a forma de dois circuitos de televisão ligados por cabo, atualmente esta tecnologia ainda é usada como um mecanismo de entrevista ou apresentação em direto por empresas de televisão através de ligações de rádio UHF ou VHF (Early et al., 1993).

A tecnologia de videoconferência proliferou-se comercialmente com a introdução do "CU-SeeMe", uma solução de chamada de vídeo da Apple (Han & Smith, 1997). O serviço foi popularizado por alguns serviços como o Skype, iChat, e outros softwares de vídeo de baixo custo, que apenas requerem uma ligação à Internet. No ano de 2011, foi criado o WebRTC, uma tecnologia que facilitou o desenvolvimento de aplicações com comunicação em tempo real, fornecendo uma interface ao programador para aceder ao mecanismo audiovisual do computador, assim como disponibiliza ao mesmo uma interface para realizar chamadas multimédia através do navegador web (Suciu et al., 2020).

2.2.2. Telepresença

"A experiência de estar presente num local no mundo real, longe do seu ambiente físico imediato" (Mair, 1997).

Um exemplo de telepresença objetiva foi o momento que aconteceu no Festival de Música e Artes de Coachella, em abril de 2012, onde um holograma à escala real de Tupac Shakur cantou as suas canções duas décadas após a sua morte. Esta ilusão, foi conseguida com o posicionamento de vidro no palco em ângulo refratando a imagem de ecrãs escondidos para o público. As animações gráficas dos movimentos do artista falecido foram baseadas em vídeos antigos (Harris, 2013; Maass, 1999; Tsukayama, 2012).

Um grande avanço e exemplo de telepresença subjetiva foi a introdução do *YouTube 360*, agora renomeado *YouTube VR*, uma funcionalidade de vídeos de 360 graus disponível na plataforma *YouTube*, possibilitando a qualquer utilizador detentor de uma câmara compatível com um campo de visão de 360 graus gravar e partilhar os seus vídeos, assim como ao utilizador detentor de um dispositivo VR assistir ao vídeo num modo imersivo (Verma, 2015).

Com a pandemia de Covid, o ano de 2020 foi um impulsionador para um novo tipo de eventos, nomeadamente eventos virtuais, onde os intervenientes do mesmo estão telepresentes, e eventos híbridos onde alguns participantes estão telepresentes e outros fisicamente presentes. Jim Sharpe, o CEO da empresa *"*Aventrique" disponibiliza uma plataforma de eventos que suporta tanto eventos híbridos como virtuais. Jim Sharpe acredita que o futuro é híbrido devido ao legado da pandemia. Uma das principais provas é *a "*Computex", uma feira de tecnologias de informação, comunicação e internet das coisas estreada em 1981, onde depois de ter sido cancelado em 2020, se converteu a um evento híbrido em 2021 (Copans, 2020; Hampel et al., 2020; T. Lin & Chao, 2021).

2.3. Síntese

Este capítulo pode ser resumido nestes pontos:

- A realidade, comparada com a virtualidade, é o mundo real para um mundo possível, e cada abordagem descrita antes, por exemplo, AR, VR, MR e AV, são meios para trazer elementos do seu mundo base para o outro. Estas abordagens podem ser usadas para experienciar telepresença.
- A realidade virtual é um mundo completo e pretende fazer com que o utilizador acredite que está nesse mesmo mundo.
- Realidade aumentada é a inserção de objetos virtuais no mundo real.
- Virtualidade aumentada é a inserção de objetos reais num mundo virtual.
- A Realidade Mista ou é um superconjunto de AR com objetos reais e virtuais num mundo, uma versão estendida de AR, onde o ambiente à sua volta é compreendido e ajustável, ou é apenas um termo de marketing para realidade aumentada, de acordo com as definições atuais.
- As tecnologias AR e VR são atualmente aplicadas nos cuidados de saúde e estudadas maioritariamente para implementar na indústria de Manutenção.
- A comunicação audiovisual pode ser facilmente disponibilizada aos utilizadores através de plataformas como o Skype e iChat e para programadores de software através do protocolo WebRTC.
- A presença de um utilizador é considerada objetiva se o utilizador consegue resolver a tarefa designada e subjetiva se a perceção for semelhante.

3. GlarAssist - Assistência Remota

GlarAssist é um produto de assistência remota que combina videoconferência e Realidade aumentada (AR) num dispositivo móvel computorizado, sendo o motor de AR o ARCore da Google, a tecnologia de videoconferência WebRTC e o dispositivo computorizado um dispositivo Android.

A assistência remota é constituída pelo assistente, também conhecido como perito ou operador remoto, e do lado assistido, conhecido por cliente, emissor ou utilizador.

O assistente pode ser qualquer navegador de internet compatível com o protocolo WebRTC.

Este capítulo está organizado em 3 secções, nomeadamente, análise competitiva, metodologias de desenvolvimento do projeto, e funcionalidades do produto em conjunto com os obstáculos ultrapassados ou contornados.

3.1. Requisitos

Antes do início do projeto, estes foram os requisitos especificados:

1. *Streaming* de vídeo bem-sucedido através de AR.
2. Suporte ao desenho de AR no lado assistente e emissor.
3. Comunicação áudio bidirecional bem-sucedida.
4. Recuperação automática de comunicações após quebra de rede.
5. Integração fluída com o desenvolvimento paralelo em curso tanto na versão Web como de servidor e sinalização WebSocket.

3.2. Análise Competitiva

Com base na análise competitiva do Glartek, atualmente existem cerca de 18 concorrentes para assistência remota, representados anonimamente no Anexo B descrevendo as principais funcionalidades suportadas. A Figura 6 representa um gráfico de barras que agrega o número de concorrentes pelas funcionalidades suportadas.

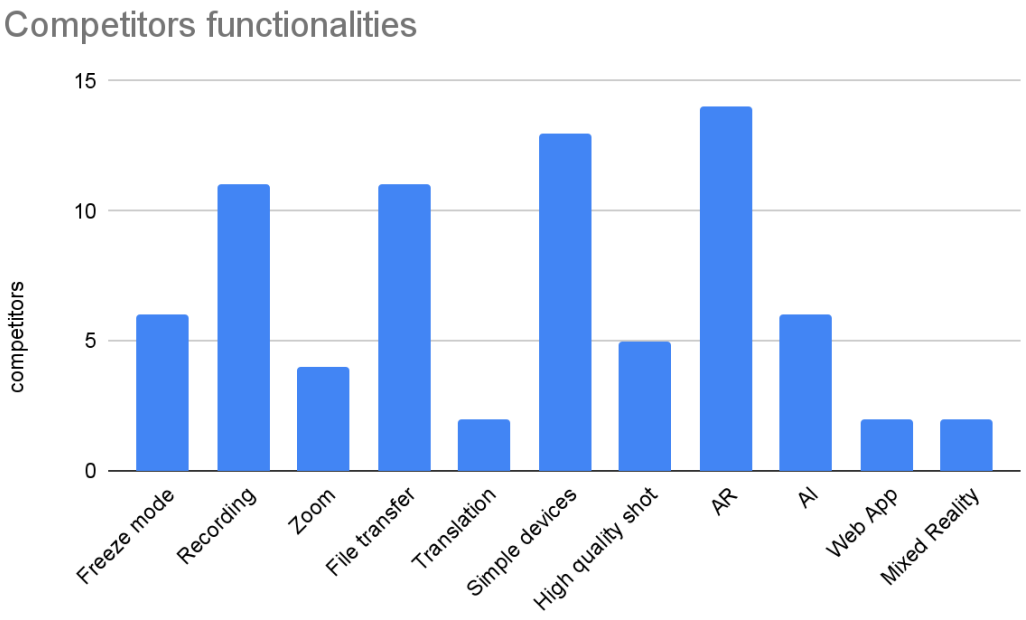

Figura 6 Concorrentes agrupados por funcionalidades

As funcionalidades mais comuns entre os concorrentes são a Realidade Aumentada (AR) e o suporte para óculos inteligentes. Alguns concorrentes usam Inteligência Artificial (IA) para classificar a chamada no final com uma etiqueta, ou para treinar um modelo com conversas de texto de chamadas passadas e usado para responder automaticamente a futuras questões de clientes, alguns concorrentes usam IA para rotular objetos vistos através da câmara usando *Deep Learning* para deteção de objetos (Brownlee, 2019). Outra característica diferenciadora de alguns concorrentes é o uso de técnicas para inserir o especialista no mundo de forma semelhante à de "Behand", mencionada no capítulo anterior, através de técnicas de remoção do fundo do fluxo de vídeo remoto em conjunto com a sobreposição no fluxo de vídeo local, criando uma experiência de Realidade Mista, permitindo objetos remotos sobrepostos no lado do cliente representados na Figura 7.

Figura 7 A realidade mista de Help Lightning © 2015-2021 Help Lightning, Inc. | All rights reserved.

Alguns concorrentes em vez de desenvolverem uma aplicação nativa Android ou iOS, criaram uma aplicação *Web Progressive* (Sheppard, 2017) que permite usufruir de funcionalidades nativas do dispositivo numa página *web* através de um *browser* compatível, e.g. aceder à camara num dispositivo Android. Um exemplo de uma API Web Progressive é o WebAR (*Augmented Reality for the Web*, 2021), representada na Figura 8, que utiliza uma biblioteca JavaScript onde, no caso de *smartphones*, traduz as instruções para as respetivas *APIs* nativas de AR, permitindo que se a página for acedida com um dispositivo compatível como um *smartphone*, utilize as suas funcionalidades AR. No caso de outros dispositivos incompatíveis apresenta uma página *web* normal. Isto permite disponibilizar ao cliente um único endereço de Internet e não requer qualquer instalação de aplicação, o site adapta-se ao dispositivo.

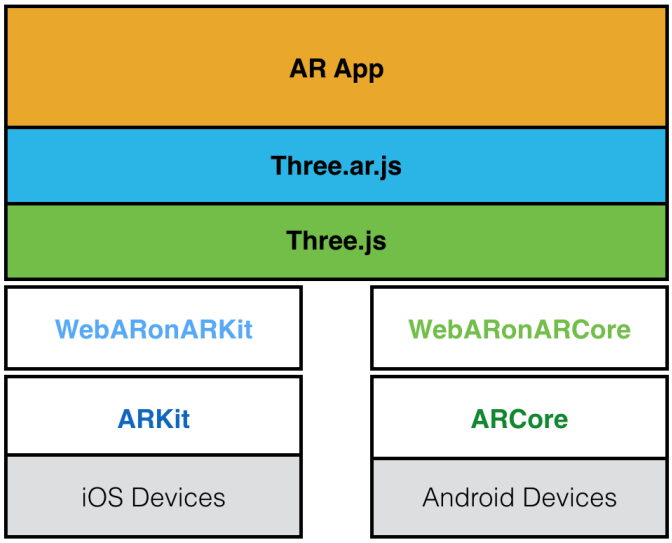

Figura 8 Three AR - biblioteca JavaScript para desenhar AR em navegadores móveis suportados

O modo de Congelamento é a capacidade de parar o fluxo de vídeo no momento escolhido e desenhar em cima dessa imagem. Essa abordagem permite pausar a imagem e dar assistência com base nesse mesmo momento passado, permitindo assistência mesmo que o utilizador esteja a mover-se ou quando o motor AR não está disponível. Algumas empresas estendem esta funcionalidade e, ao mesmo tempo que congelam a imagem usam o obturador da câmara para capturar o mesmo momento em alta resolução. Segue-se o envio dessa imagem para o lado remoto, nomeado como captura de alta qualidade, permitindo um maior detalhe na assistência daquele momento capturado com uma resolução de imagem superior que a disponível no fluxo de vídeo. É assim possível mostrar textos que podem ser de difícil legibilidade devido à compressão de vídeo, o que também permite em áreas de baixa largura de banda enviar uma imagem ao especialista e fornecer uma melhor assistência usando apenas voz e essa imagem.

Algumas empresas fornecem tradução ao vivo para a comunicação áudio e texto e a gravação é publicitada como uma funcionalidade de formação, assim como uma ferramenta para resolver problemas recorrentes.

3.3. Metodologias de Desenvolvimento

De acordo com (Poppendieck, 2007) "À medida que a competitividade aumenta na indústria de desenvolvimento de software, os intervenientes destacam-se pela melhor maneira de criar bom Software rapidamente, repetidamente e coerentemente".

Antes de encarar o processo de desenvolvimento, o primeiro passo é decidir a metodologia de desenvolvimento mais adequada. Tomando em conta o ritmo necessário para permitir ao projeto ser competitivo, o número de elementos de equipa reduzido, variando de 2 a 3 pessoas, e a falta de definição inicial, decidiu-se seguir os princípios presentes no “Agile Manifest”, desenvolver usando “Scrum”, e organizar as tarefas num quadro Kanban.

Os quatro princípios do Agile Manifest são "Indivíduos e interações sobre processos e ferramentas, software funcional sobre documentação abrangente, colaboração com o Cliente sobre negociação de contratos e Resposta à mudança em prole de um plano" (*Manifesto for Agile Software Development*, 2001).

Um quadro Kanban é uma representação visual das tarefas e do seu progresso de acordo com o seu estado. Tendo em conta a Figura 9, as tarefas são criadas e, quando prontas a ser implementadas, são inseridas na área "Open" com pelo menos um título e uma prioridade definida, o programador quando disponível escolhe uma tarefa com a maior prioridade disponível, move-a para a área "In progress" e começa a desenvolver. Quando feita a tarefa essa é movida para a área "Review" onde, se aprovada, é fechada e desaparece. Se não cumpre os requisitos e precisa de revisão, reverte para a área "Open" (Ordysiński, 2013).

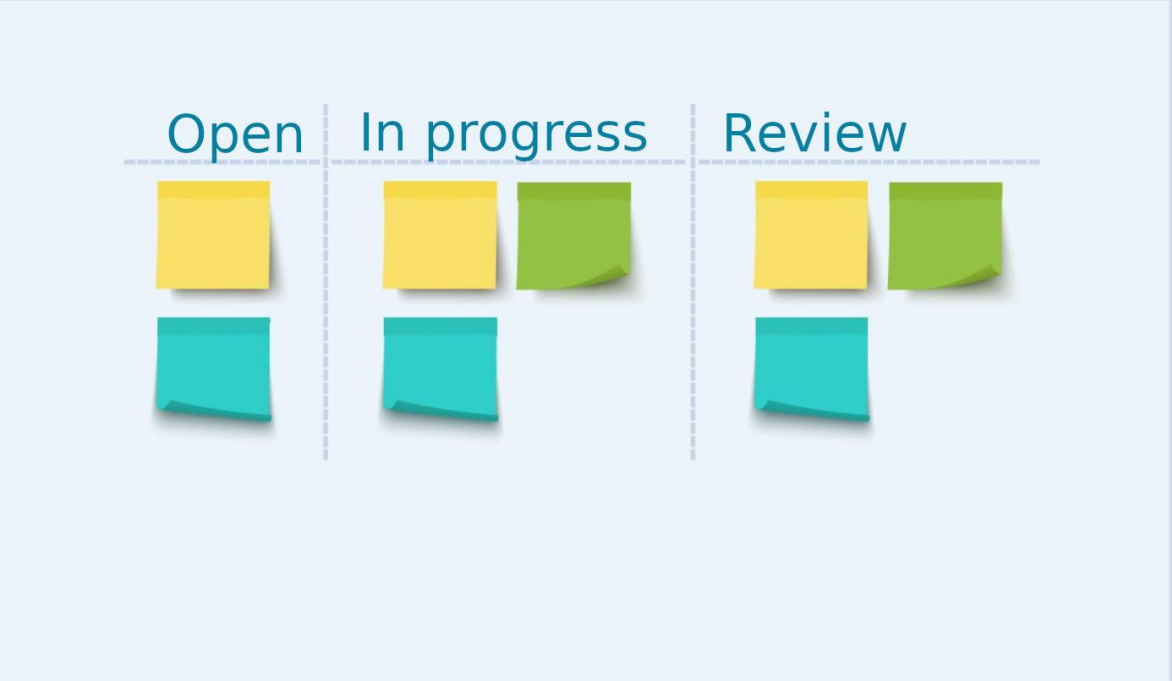

Figura 9 Quadro Kanban

Apesar do projeto começar com este quadro Kanban com os rótulos “Open”, “In progress” e “Review”, eventualmente cresceu com o tamanho da equipa, descrito mais à frente neste capítulo.

Git é um sistema de controlo de versões distribuído (VCS) para ficheiros, tendo como exemplo código informático, o código está em todos os nós, e o controlo da versão é um sistema que regista o estado das alterações num ponto de tempo definido, (Somasundaram, 2013). Os principais conceitos do Git são:

- Um repositório é o diretório que está a catalogar as versões.
- Uma *branch* é o código relacionado com a questão que está a ser desenvolvida e cria um histórico numa área isolada.
- *Merge* é o ato de juntar o dois *branches*, geralmente na direção do *branch* relacionado à tarefa concluída para o *branch* comum.
- O *Merge Request* (MR) é um estado de pré *merge* que incentiva a discussão com os membros da equipa sobre o código que está a ser pedido para juntar.

No mundo de desenvolvimento de software, o código por si só, são apenas as instruções escritas numa linguagem e sintaxe compreensíveis pela máquina, conhecida como linguagem de programação. O código é essencialmente usado para gerar um programa através de um processo de compilação e distribuição para o servidor ou para uma loja de aplicações.

Integração Contínua (CI) é a automatização da compilação e teste do software e entrega contínua (CD) é a automatização do processo de distribuição (Arachchi & Perera, 2018).

Para controlar as versões é usada a suite Gitlab, que utiliza Git como VCS, disponibiliza fluxos CI/CD, bem como fornece a gestão do projeto com a atribuição de etiquetas, permitindo que as referidas etiquetas sejam usadas como quadros Kanban, assim como outras funcionalidades e.g. Wiki para documentação.

A primeira versão da aplicação foi desenvolvida como prova de conceito e as suas principais funcionalidades desenvolvidas nas primeiras 5 tarefas, descritas na Tabela 1 e no Anexo A mais detalhadamente. Uma das primeiras mudanças de processo de desenvolvimento foi realizada durante esta primeira iteração, sendo a última vez que foi previsto e reportado o tempo gasto em cada tarefa, algo que acredito que ajudou na

criatividade de resolução de problemas, reduzindo a pressão de prever a quantidade de tempo e posterior stress de tentar cumprir o tempo previsto.

Nome	Tempo gasto
Construir UI de Assistência Remota	21h10
Desenhar notações AR em um avião	24h
Construa a Biblioteca API REST	10h
Implementar cliente WebSocket	10h
Implementar o streaming WebRTC	N/T

Tabela 1 As primeiras tarefas definidas

À medida que a equipa cresceu e com o objetivo de lançamentos constantes e incrementais, procedeu-se a algumas mudanças durante o desenvolvimento da aplicação.

A primeira mudança foi a unificação do repositório Git onde se passou a ter um único repositório em vez de usar um repositório para cada componente, nomeadamente, *Backend*, Web e aplicação Android. Isto resolveu um problema onde uma tarefa teria de ser copiada para cada repositório de cada componente da aplicação e em tarefas onde envolvia múltiplos componentes subiam de forma independente e desalinhada gerando mais problemas, uma consequência positiva foi a possibilidade de ter apenas um fluxo de CI/CD para lançar todas as plataformas.

Algumas desvantagens nesta abordagem são quando ocorre um conflito de *merge*, e o desenvolvedor designado não tem conhecimento do código feito nas outras plataformas o que exige que a equipa de desenvolvimento pare e faça uma chamada de grupo para juntar as alterações. Outro desafio é a utilização do "Android Studio" – o software de desenvolvimento Android, para a fusão não reconhece as linguagens de programação das outras plataformas, tornando os erros de sintaxe no *merge* mais frequentes.

Um pequeno desafio com esta abordagem surgiu no "Android Studio" depois de juntar todos os projetos a um repositório onde a pasta base não era classificada como uma Aplicação Android e precisava de ser aberta numa subpasta específica, mas os ficheiros de dados do repositório estavam na pasta raiz, por isso, ao abrir a pasta raiz como um projeto reconheceria o VCS, mas não como um projeto, e ao abrir na subpasta reconheceria como sendo um projeto, mas não um repositório de Git. Isto foi resolvido utilizando o *sparse*

checkout na subpasta que permitia o controlo de versões no “Android Studio” enquanto aberto na subpasta, mantendo apenas a pasta Android no repositório, poupando espaço de armazenamento na máquina dos desenvolvedores.

Prontamente após o crescimento da equipa, pela inclusão das equipas de produto, comercial e de marketing, foram adicionados alguns passos ao quadro de Kanban. Os primeiros passos adicionados foram para a definição do produto, nomeadamente “Ideia” onde a tarefa é criada no Gitlab, brevemente descrito, por vezes apenas um título, em seguida, passa para "Story definition” onde é criada a descrição e protótipos de alta-fidelidade, bem como são estabelecidas as funcionalidades e requisitos que definem uma tarefa como concluída. Após decisão executiva a tarefa ou volta para "Story definition" ou passa para “Approved user story” onde será movida pela equipa de Produto para o estado “Open” ou é movida para "Backlog” uma nova zona de tarefas que podem ser desenvolvidas depois de todas as tarefas que estão no estado "Open" terem ido para o estado "Review". É geralmente usada para questões menos importantes ou problemas de rápida resolução.

Uma grande mudança no entrar da quarentena de 2020, foi a introdução da metodologia Scrum, um modelo que define passos, e os papéis de cada elemento da equipa para organizar um projeto (*Scrum Guide | Scrum Guides*,2020).

Scrum é uma estrutura de trabalho baseada em quatro etapas:

1. Um “Product Owner” encomenda o trabalho para um problema complexo e coloca num *backlog* de produto.
2. A Equipa Scrum transforma uma parte do trabalho em algo de valor.
3. A Equipa Scrum e as suas partes interessadas inspecionam os resultados e ajustam-se para o próximo Sprint.
4. Repetir

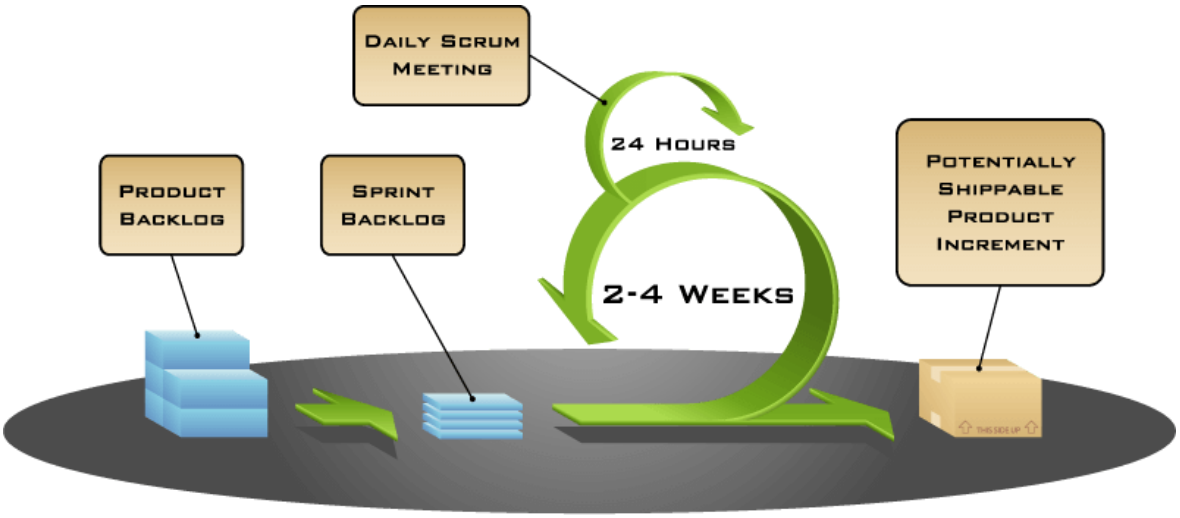

Figura 10 modelo básico de Scrum

Existem 5 momentos principais definidos nesta estrutura de trabalho, representadas graficamente na Figura 10, os pontos seguintes enumera-os e descreve como foram adaptados às necessidades da nossa equipa.

Sprint planning – momento onde as tarefas são apresentadas aos programadores e as questões feitas à equipa do produto, bem como é promovida a discussão tendo em conta as limitações técnicas e, se necessário, adaptada a definição para contornar essas mesmas limitações.

Sprint - onde as tarefas anteriormente apresentadas são desenvolvidas, atualmente durante uma semana, mas com planos de mudança para duas semanas.

Daily Scrum - onde cada equipa de desenvolvimento fala com o gestor do projeto, revela o seu progresso, e revela se está bloqueado de alguma forma, assim como é discutido o que precisa ser feito para contornar o problema, bem como definir quais são os próximos desenvolvimentos do dia.

Sprint Review - realizada no final da semana, onde cada desenvolvedor apresenta à equipa de produto e marketing as tarefas concluídas.

Sprint Retrospective – de periodicidade esporádica, mas em eventos em que algo não está ideal ao nível da gestão de projeto, é agendada uma reunião extraordinária para discutir as mudanças necessárias a ser implementadas. Um exemplo foi a necessidade de uma melhor documentação das tarefas definidas, devido a algumas tarefas que envolvem todas as

plataformas exigirem a necessidade de definir a estrutura de mensagem para o protocolo de comunicação. Mais tarde, foi decidido introduzir um passo extra do quadro de Kanban entre o estado "Open" e "In progress" denominado "Preparing" onde uma entrada Wiki é definida para a tarefa, especificando todas as mensagens de integração. Também foi introduzida uma nova etiqueta para cada plataforma para verificar facilmente as plataformas envolvidas na tarefa e marcar a completude para cada plataforma, por exemplo, "android" e "android ready". Quando todas as plataformas marcadas estiverem prontas, o último utilizador a terminar a sua parte do problema marca a sua plataforma como completa e move o problema para "Review".

Uma grande parte da conclusão da tarefa é o teste, algo a que aprendemos a dar mais atenção, uma vez que os desenvolvedores são incrivelmente cuidadosos com os testes e seguem sempre o caminho normal. Algumas alterações no processo para melhorar a qualidade da aplicação, foram, a introdução de um elemento na equipa para realizar testes no fim de cada tarefa assim como após o *merge* na *branch* principal, permitindo uma redução nos problemas de integração, momento onde apesar de muitos problemas serem revelados no fluxo CI/CD quando quebram a sintaxe, outros geram comportamentos inesperados detetáveis apenas testando manualmente.

3.4. Interface de Utilizador

Esta secção refere-se ao progresso da aplicação Android em termos de interface e experiência do utilizador.

A aplicação teve 2 versões de interface de utilizador, a primeira tinha o objetivo de provar se as funcionalidades propostas inicialmente eram possíveis, e a versão atual, que foi criada após a introdução de um membro de design na equipa.

Uma das necessidades básicas da aplicação é de o informar quando alguma operação está a decorrer ou quando algo correu mal, levando ao desenvolvimento de mensagens de estado que são exibidas a informar a mudança desse mesmo estado, por exemplo quando não existe ligação à Internet deve apresentar uma mensagem de erro. A primeira aplicação tinha 2 tipos de mensagens de utilizador, uma para informação, e outra para erros, sob a forma de uma barra horizontal azul para informação, representada na Figura 11, e outra de cor vermelha para o erro, representada na Figura 12.

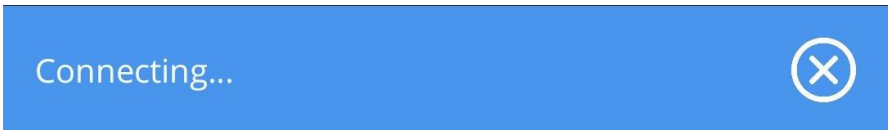

Figura 11 Representação de mensagem de informação para conexão a decorrer

Figura 12 Representação de mensagem de erro para falta de internet

A primeira iteração da interface era simplesmente uma página com um campo de texto para introduzir um número de chamada para o utilizador se juntar à sessão, o "CREATE ROOM" servia apenas como um atalho *web* para uma página onde era possível gerar números de chamada, representado na Figura 13.

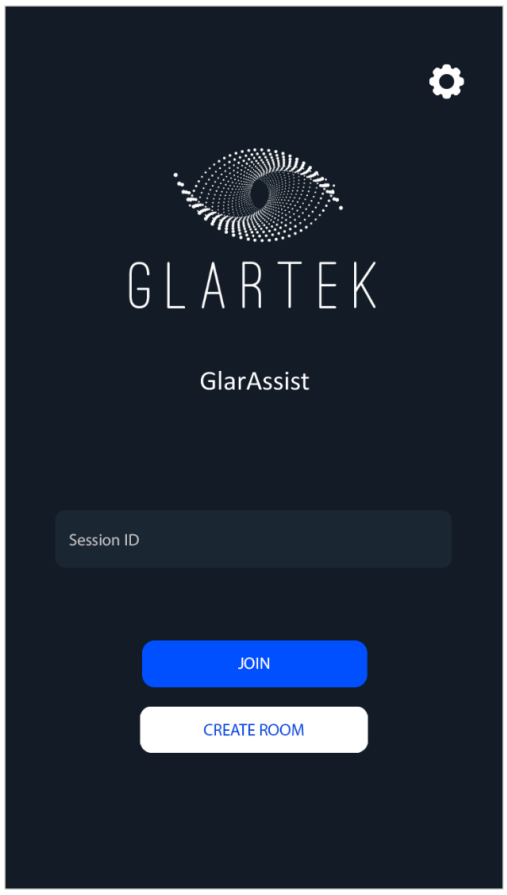

Figura 13 Primeiro protótipo de ecrã principal para a primeira iteração da aplicação

Depois de clicar no botão Join, a aplicação mostrava uma mensagem de informação com o texto "Connecting…".

Após se juntar numa sessão, o utilizador é direcionado para a página de assistência, onde permanecerá em estado de espera por utilizadores, representado na Figura 16, até que um utilizador remoto se junte à sala, transitando para o estado de assistência representado na Figura 17.

A interface tinha dois indicadores visuais para o estado da ligação, o estado de ligação com o serviço, no canto superior esquerdo e o estado de ligação com o utilizador na zona superior ao centro, ambos de cor verde quando ligado, laranja quando em progresso e branco quando desligado.

A nível das extremidades direitas, na posição superior, existia um botão que permite transitar entre dispositivos de saída de áudio, como altifalante, auricular ou dispositivo Bluetooth quando disponível, representados na Figura 14, e no canto inferior direito, permitia transitar entre microfone ligado e desligado, representado na Figura 15.

Figura 14 Ícones de estados de som para a primeira iteração da aplicação

Figura 15 Ícones de estados de microfone para a primeira iteração da aplicação

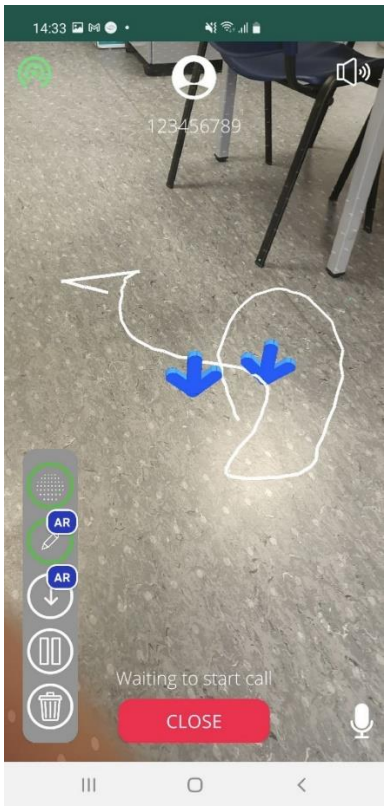

Figura 16 Ecrã de espera de assistente para a primeira iteração da aplicação

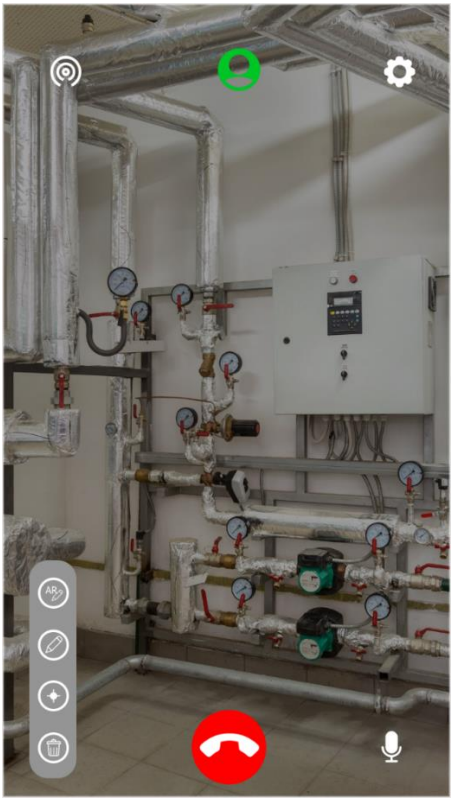

Figura 17 Protótipo de chamada para a primeira iteração da aplicação

Ainda no ecrã de assistência, a zona inferior esquerda contém o menu de ações relativas ao desenho representado na Figura 17 e descrito de cima para baixo:

- Botão de alternar a visibilidade da grelha AR, vista no chão do fluxo de vídeo representado na Figura 16.
- Botão de desenho de linha, quando permite desenhar linhas.
- Botão de desenho de setas, que quando selecionado possibilita colocar setas e alterna com o botão de linha.
- Botão de pausa que alterna entre o modo de congelamento, e o modo de direto.
- Botão de limpar, que elimina todos os desenhos.

No ecrã de espera pelo assistente, estava representado o código de sessão na zona inferior ao centro, e o botão representaria o fecho em vez de um botão de desligar a chamada.

A segunda e atual interface de utilizador, melhorou o design seguindo as diretrizes do Material Design um sistema de design criado pela Google (Material Design, n.d.) assim como adaptou o fluxo para ser escalável.

O ecrã inicial foi redesenhado para ter um cartão central onde toda a navegação da aplicação é feita até ao ecrã de assistência, representado na Figura 18. O ecrã principal está representado na Figura 19.

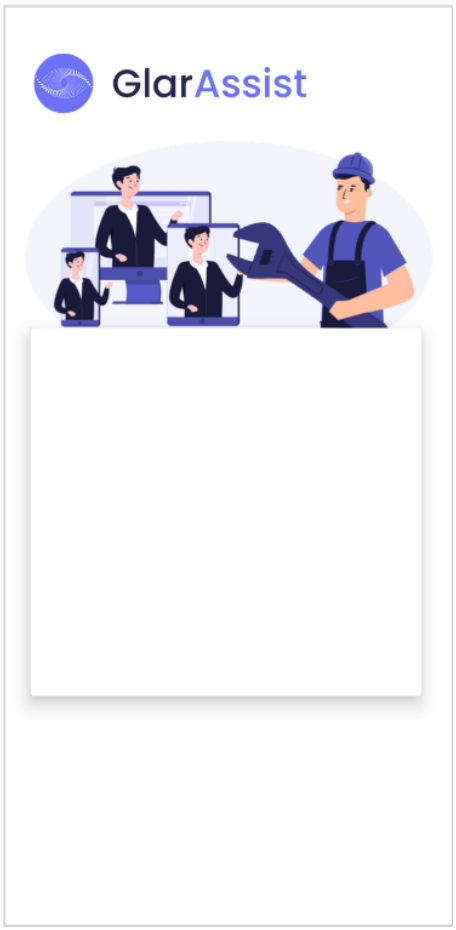

Figura 18 Protótipo do modelo de tela inicial

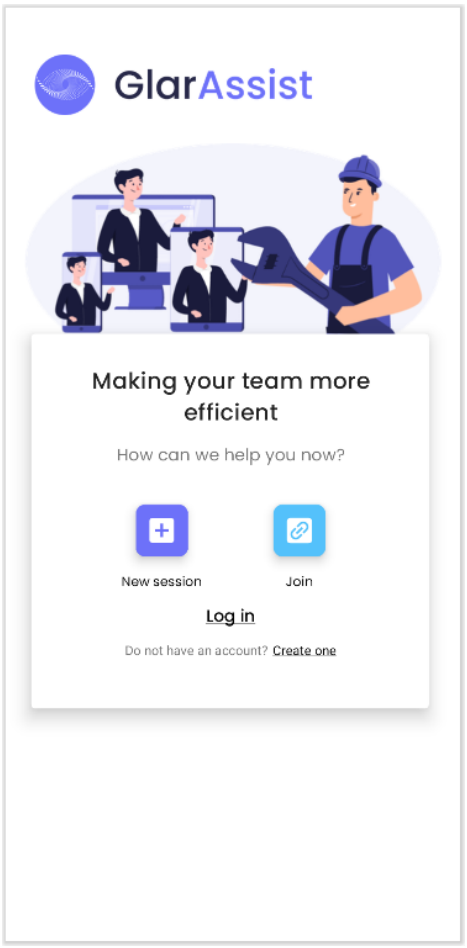

Figura 19 Protótipo de tela inicial

Seguindo a Figura 20, existem os seguintes caminhos possíveis na tela inicial:

- Colorido a azul existe o fluxo do registo, onde o utilizador que não tem uma conta clica para criar uma escrevendo as credenciais pretendidas, e em caso de falha de receção do e-mail, o utilizador pode solicitar outro.
- Colorido a vermelho está o caminho de entrada, onde o utilizador que já tem uma conta, clica em ”Log in”, escrevendo as suas credenciais, e no caso de se esquecer da palavra-passe pode recuperá-la por e-mail usando um mecanismo de recuperação de senha.
- Colorido a amarelo é o caminho para o qual o utilizador é levado após solicitar uma redefinição de senha, começando pelo caminho de entrada, e onde precisará de introduzir uma nova palavra-passe.

- Colorido a verde é o caminho para o qual o utilizador é conduzido depois de um outro utilizador lhe ter criado uma conta e precisará de introduzir a palavra-passe pretendida.

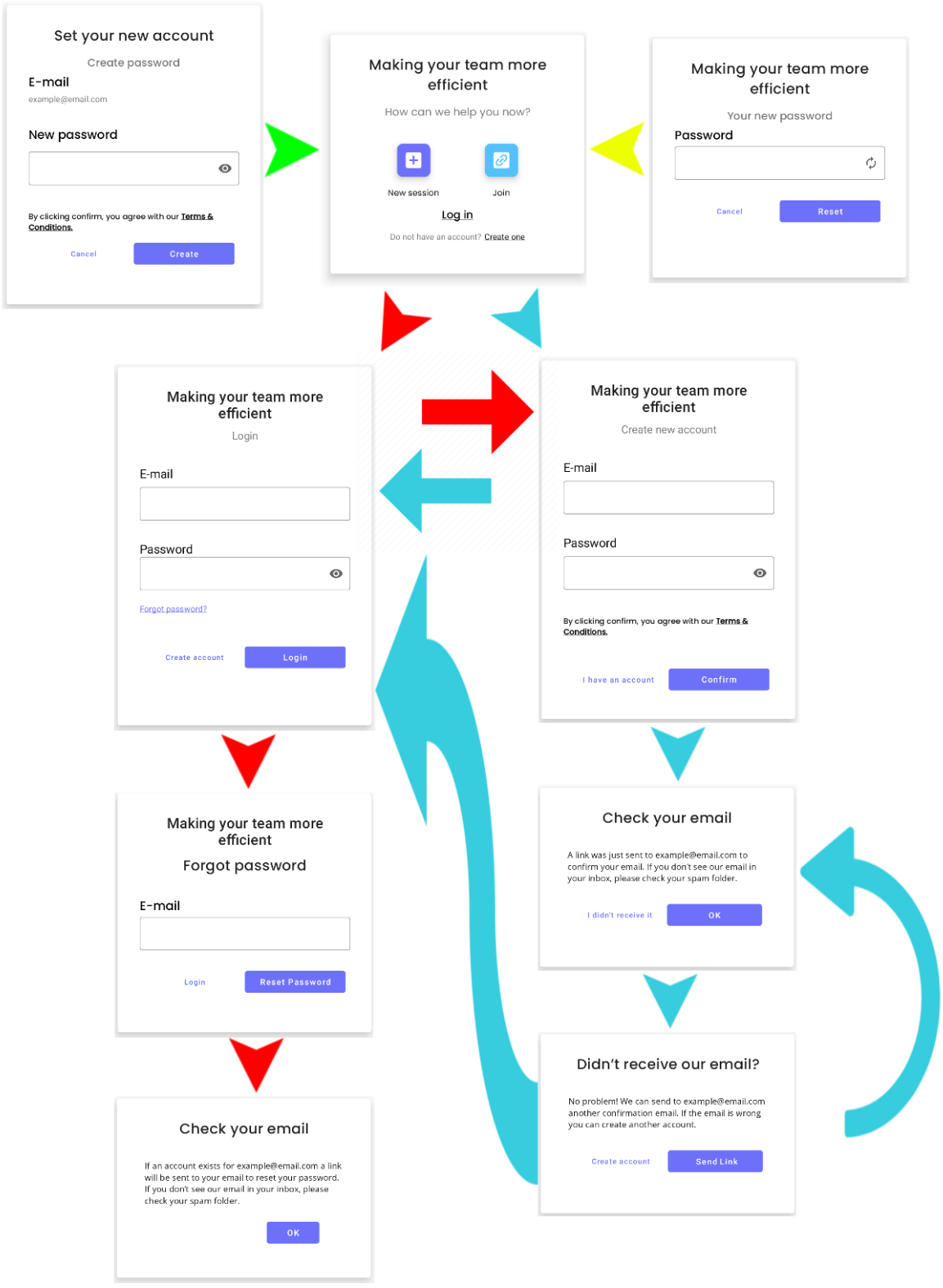

Figura 20 Esquema de navegação do ecrã inicial

Ainda no ecrã principal existem dois caminhos principais para ir ao ecrã de assistência, criando uma sessão, clicando no botão de “New session” ou juntando-se a uma sessão existente, clicando no botão “Join”. O ecrã de entrada na chamada está representado na Figura 21 onde é permitido ao utilizador introduzir um número de chamada para que possa participar na mesma. Depois de se juntar ou seguindo o caminho criar uma sessão, o utilizador é conduzido a um ecrã de escolha do papel na chamada onde pode optar por ser assistente, assistindo o fluxo de vídeo transmitido remotamente ou assistido, partilhando o seu vídeo, representado na Figura 22.

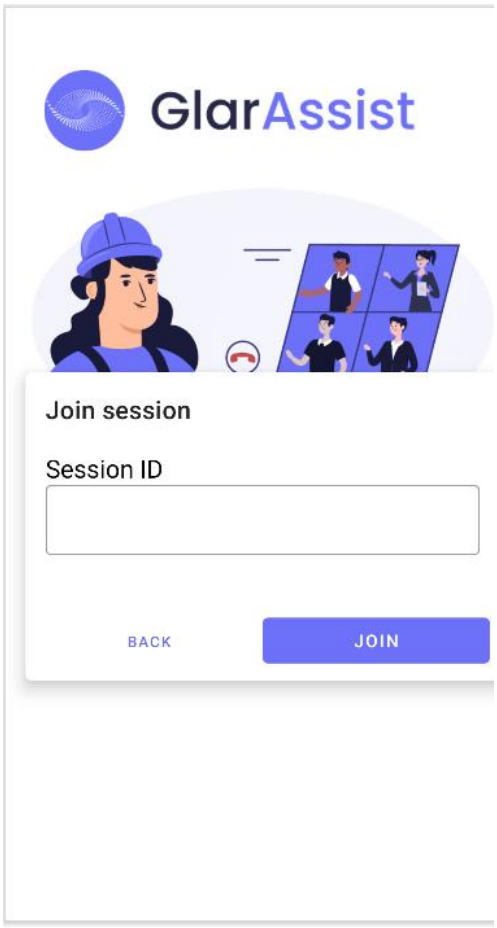

Figura 21 Protótipo de ecrã de entrada na sessão

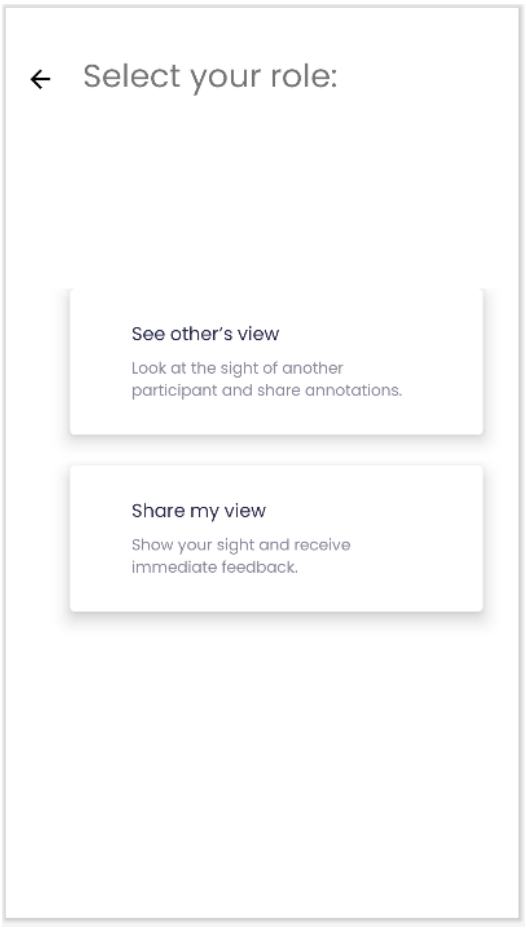

Figura 22 Protótipo de escolha de papel na sessão

Caso o utilizador escolha ver a vista remota, o mesmo é levado à versão móvel da aplicação. Caso o utilizador opte por partilhar o seu fluxo de vídeo, é levado ao ecrã de assistência representado nas combinações de retrato, paisagem, colapsado e expandido das Figuras 23 a 26 onde está representado:

- No canto superior direito, um contador de tempo representando a gravação de chamada a decorrer.
- No canto superior esquerdo mostra o número de assistentes presentes na chamada e apresenta um botão alternador entre o modo AR e o modo 2D, detalhado na secção 3.5.5.
- Nos cantos inferiores apresenta menus expansíveis com ações do utilizador.
- Na parte inferior central existe um botão de alternar entre o modo de congelamento e o modo de direto, detalhado na secção 3.5.4.

O menu de ações à esquerda é separado por ações primárias em baixo e ações secundárias no topo, representadas na Figura 24, estas ações incluem, de baixo para cima:

- Apagar desenhos
- Alternar o modo de caixas de texto
- Botão de menu de escolha de cor
- Botão de receção de arquivo, detalhado na secção 3.5.3
- Botão de chat
- Botão de captura
- Botão de gravação, detalhado na secção 3.5.8
- Botão de reconhecimento de texto no vídeo (OCR)

As ações do menu direito incluem, de baixo para cima:

- Botão de definições
- Botão de lanterna
- Botão de microfone
- Botão para terminar a chamada

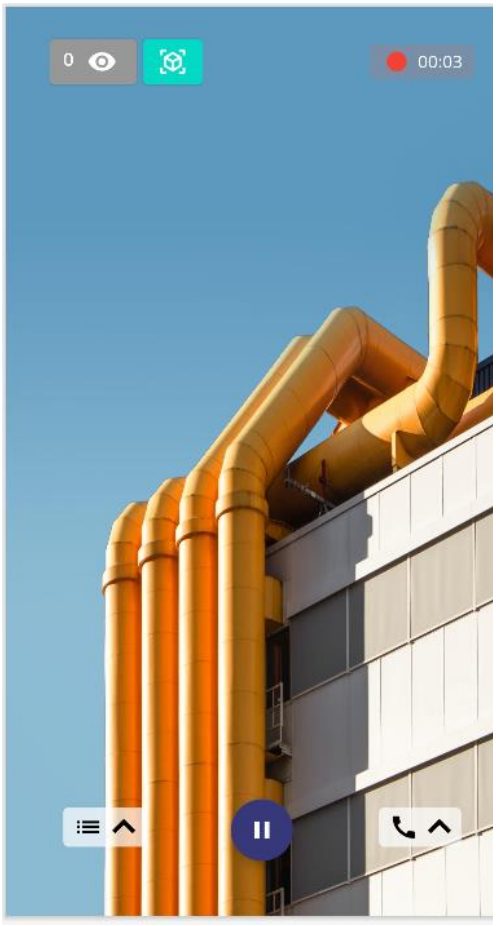

Figura 23 Protótipo de Ecrã de assistência de menus colapsados em modo retrato

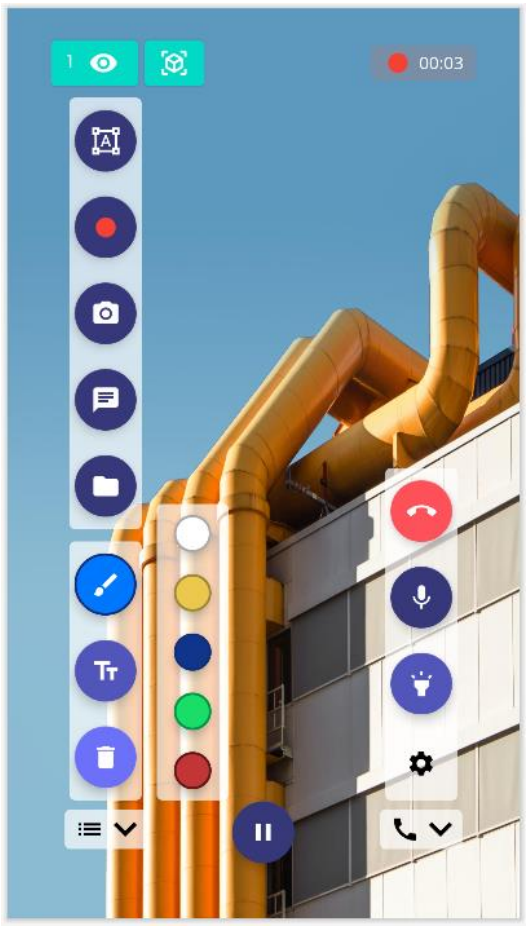

Figura 24 Protótipo de Ecrã de assistência de menus expandidos em modo retrato

A disposição dos ícones foi adaptada com base no modo vertical para um modo horizontal devido a limitações de espaço, onde o menu esquerdo passou a conter um botão para mais opções para as ações secundárias como representado na Figura 26.

Figura 25 Protótipo de Ecrã de assistência de menus colapsados em modo paisagem

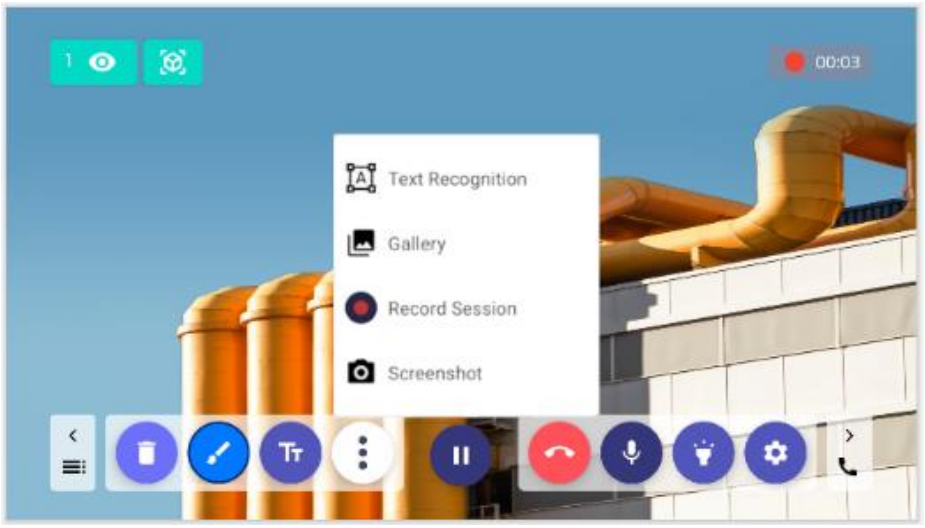

Figura 26 Protótipo de Ecrã de assistência de menus expandidos em modo paisagem

Estes botões de ações mencionados anteriormente são ações para funcionalidades descritas nas próximas secções.

3.5. Tecnologias e funcionalidades

Estando cientes do mercado e com o processo de desenvolvimento definido, esta secção foca-se nas tecnologias e funcionalidades que foram desenvolvidas para fazer com que a aplicação resolva os requisitos definidos pela equipa de produto.

3.5.1. WebRTC

A primeira iteração do GlarAssist usou o WebRTC como tecnologia de comunicação para enviar o vídeo do assistido para o assistente e o áudio bidireccionalmente. O WebRTC também foi utilizado para envio de eventos com o estado da aplicação e para o envio do desenho. Para estabelecer uma a ligação WebRTC, é necessário trocar primeiro os dados de ligação utilizando uma ligação WebSocket, tal como esquematicamente representado na Figura 27.

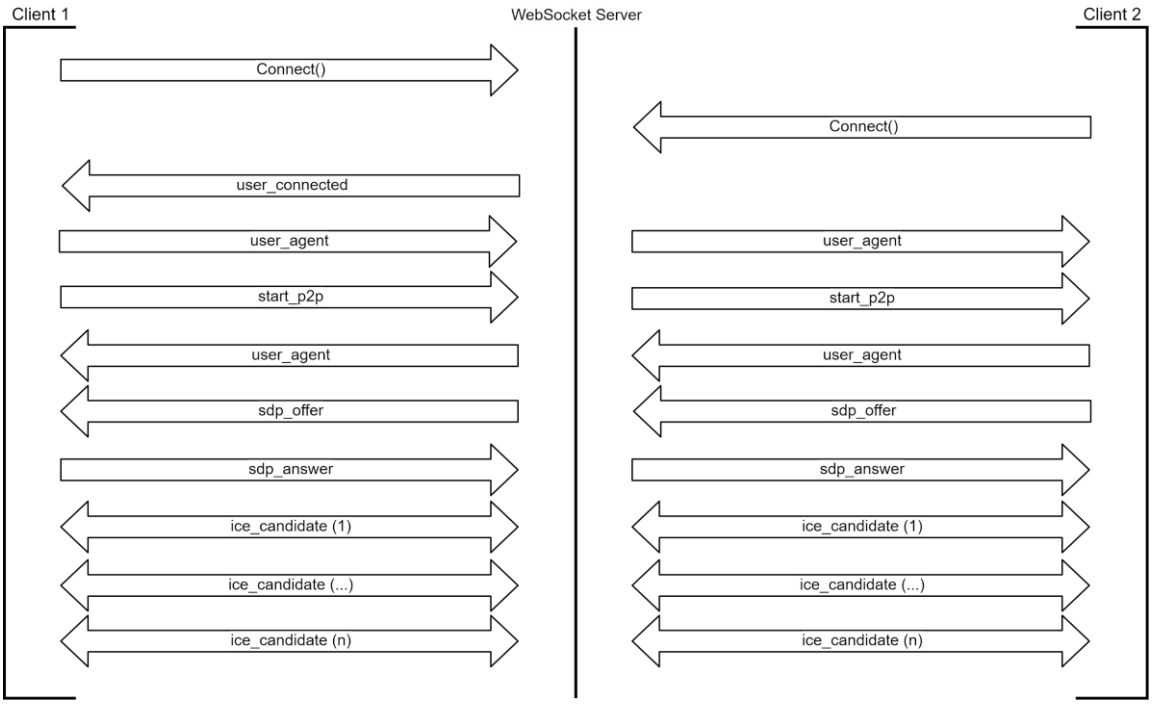

Figura 27 Troca de mensagens por WebSocket

A troca de mensagens anterior permite o estabelecimento de uma conexão WebRTC para dois utilizadores pelo caminho mais curto possível, basicamente:

- Dois utilizadores entram num canal privado através WebSocket, identificado pelo identificador da chamada e fazem uma troca de dados da sessão que inclui informações da videoconferência.
- Cada utilizador contacta servidores ICE para descobrir os seus IP locais, posteriormente partilhados com o utilizador remoto através do canal WebSocket.
- Finalmente, dependendo da rota de ligação disponível entre os utilizadores, podem ser utilizados os seguintes protocolos:

- Session Traversal Utilities for NAT (STUN) para ligações diretas ou,
- Traversal Using Relays around NAT (TURN) para ligações não diretas quando por trás de um sistema de segurança ou NAT (*WebRTC 1.0: Real-Time Communication Between Browsers*, 2021).

Após a troca de detalhes da ligação, o caminho mais curto possível é escolhido, e uma faixa para áudio e vídeo criada com um identificador único.

A faixa de áudio é atribuída ao dispositivo do microfone e a faixa de vídeo é atribuída ao vídeo do utilizador assistido, convertido em formato I420, um formato de vídeo que descreve um pixel em 3 camadas diferentes, a de luminância, e as de cores vermelha e azul (*I420 Yuv Pixel Format*, n.d.).

O vídeo que está a ser enviado é um direto da câmara com objetos virtuais desenhados por cima com a escala e posição alinhadas ao mundo real, conseguido utilizando o ARCore da Google.

3.5.2. Google ARCore

O SDK da Google para realidade aumentada (ARCore) foi o que possibilitou a colocação de anotações no mundo real.

O primeiro passo para interagir com o mundo é o mapeamento, alcançado através do ARCore que acompanha a posição do dispositivo móvel à medida que se move e, em conjunto com as suas câmaras e outros sensores, gera uma compreensão do mundo real através da identificação de pontos de interesse. Os pontos de interesse são cantos de objetos presentes no local e o ARCore rastreia as suas posições à medida que o dispositivo se move. O ARCore também usa a deteção de superfícies planas para construir a compreensão do mundo à sua volta. (*ARCore Overview | Google Developers*, n.d.). Estas superfícies planas são representadas como planos no mundo compreendido pelo ARCore e são rastreados pelo movimento do dispositivo da mesma forma que os pontos de interesse. Os objetos rastreáveis podem ser utilizados para combinar a interação do utilizador com uma posição no mundo interpretado, como demonstrado na Figura 28 e na Figura 29, utilizando um teste de colisão que projeta um raio para a visão do mundo sobre a perspetiva da câmara, devolvendo todos os planos ou pontos de características intersetados (*Fundamental Concepts | ARCore | Google Developers*, n.d.).

Tanto os pontos de interesse como os planos serão designados por objetos rastreáveis daqui para a frente.

Figura 28 Demonstração de teste de colisão ângulo 1

Figura 29 Demonstração de teste de colisão ângulo 2

Este ponto intercetado pode ser usado para inserir âncoras com objetos 3D associados, onde no caso do GlarAssist é usado para inserir uma seta em cima do mundo real usando o seguinte algoritmo:

- Ao tocar no ecrã um evento é desencadeado e colocado numa fila.
- A cada *frame* do vídeo:
    - Retirar o primeiro evento da fila.
    - Realizar um teste de colisão usando as coordenadas do ecrã do evento.
    - Caso algum objeto localizável seja intercetado, criar uma âncora e armazená-lo na lista de setas.
    - Iterar a lista de setas e desenhar no topo do fluxo de vídeo.

Este ponto intercetado pelo algoritmo acima também pode ser usado para desenhar linhas. Isto pode ser conseguido utilizando a sequência de eventos de colisão desencadeados pelo movimento do dedo através do ecrã.

O componente de desenho de linha AR foi baseado na experiência da Google “Just a line” (Google Creative Lab, 2018) que traça a linha numa posição paralela à câmara. O desenho da linha no espaço foi conseguido usando o teste de colisão contra objetos rastreáveis em vez da posição paralela da câmara. O tipo de evento gerado pelo clique no ecrã, foi usado da seguinte forma:

- Quando o utilizador começa a desenhar, um evento de pouso de dedo é desencadeado e são despoletados uma série de pontos que compõem um caminho.

- Todos os eventos seguintes enquanto o utilizador mantém o clique são eventos de movimento que insere o ponto encontrado pelo teste de colisão numa sequência de pontos 3D que compõe a linha.

Após cada ponto colocado na linha, o objeto de desenho responsável pela mesma é marcado para ser atualizado, o que, em conjunto com a utilização de um único *Vertex Buffer Object* (VBO), permite desenhar a linha no ecrã de uma vez, em vez de desenhar cada linha ponto a ponto no ecrã. O objeto de desenho é redesenhado por cada ponto adicionado à linha. O VBO atua como um pré processador de imagem onde permite gerar a imagem da linha na criação da mesma em vez de gerar uma nova todas as *frames* do vídeo.

Esta aproximação tem um grande atraso entre o que se pretende desenhar com o que é exibido na tela em AR, mais percetível quando o utilizador desenha uma linha pois o *frame* de vídeo seguinte só será mostrado após o anterior terminar de desenhar todos os pontos da linha no VBO. Para mitigar esta questão, o desenho foi paralelizado, fazendo-o funcionar num fluxo de execução secundário, e otimizado da seguinte forma:

- Os eventos de desenho foram limitados a 30 por segundo por utilizador.

- Dependendo do método de desenho selecionado:

    - Colocação de seta:

        - Usar apenas evento do levantar o dedo do ecrã.

        - fazer um teste de colisão, e se for bem-sucedido, criar uma âncora e colocá-la na lista de setas para ser desenhada no futuro.

    - Desenho de linha:

        - Fazer um teste de colisão com o evento de pousar o dedo no ecrã.

- Se o teste de colisão for bem-sucedido é gerada uma nova linha e inserida na estrutura de linhas.
- Se o utilizador mantiver o clique, o teste de colisão será realizado continuamente, e os pontos captados são usados para definir a última linha criada.
- Quando o utilizador levantar o dedo, a última linha da estrutura de linhas, é marcada como concluída.

Esta abordagem permitiu paralelizar o desenho do utilizador e o desenho da tela uma vez que o desenho é tratado por eventos paralelos desencadeados pelo clique do utilizador no ecrã, não interferindo com o fluxo de desenho da tela que corre no ciclo principal. O fluxo secundário desenha no VBO as linhas com base nos pontos atuais, e a tela principal desenha sobre os VBO mais recentes, melhorando o atraso entre o desenho do utilizador e a representação da ação.

Depth API

Com o 1.18 da versão ARCore veio a introdução da Depth API, uma abordagem que usa um algoritmo de profundidade baseado no movimento do dispositivo em conjunto com a perspetiva da câmara e de outros sensores. Em vez de planos e pontos de interesse, fornece uma imagem de mapa de profundidade por cada *frame* que corresponde a uma conversão direta entre 0 a 8 metros com base na percentagem de vermelho presente no pixel associado. Por outras palavras, o mapa de profundidade é uma imagem que, se sobreposta com o fluxo de imagem da câmara, quanto mais perto o objeto está em cada ponto, mais vermelho aparecerá no mapa de profundidade que pode ser usado para conhecer cada profundidade do pixel sobreposto. Este algoritmo tem uma precisão ideal na faixa de 0,5 a 5 metros (Google, 2021).

A API de profundidade vem com um exemplo de caso de uso que permite esconder os objetos ancorados quando por trás de algum objeto no mundo real, denominado de oclusão representado na Figura 30.

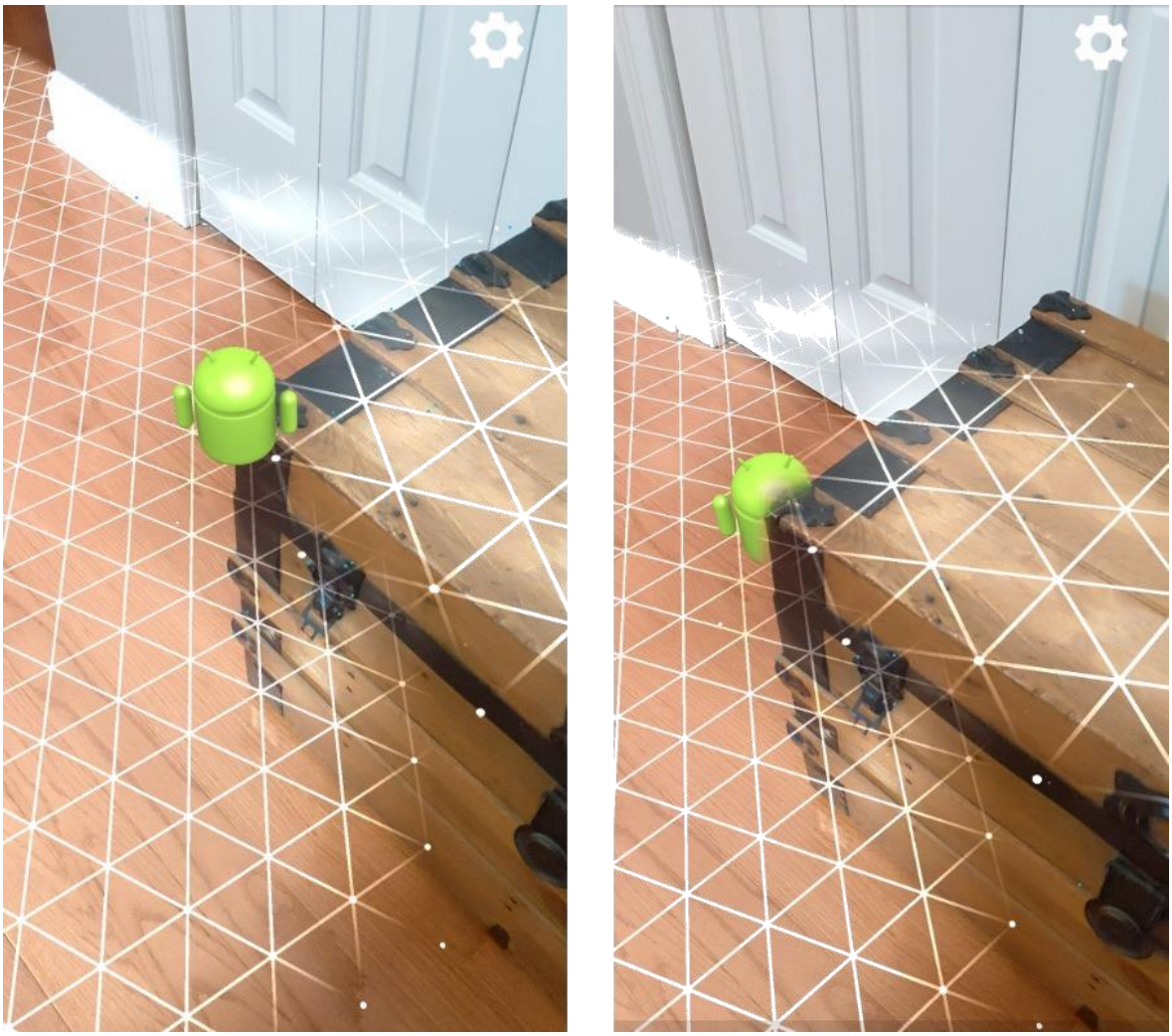

Figura 30 Depth API representação de oclusão

Isto só foi aplicado às setas uma vez que a integração com a linha não foi bem-sucedida no tempo disponível.

Neste momento, o Depth API tinha como objetivo um melhor mapeamento dos planos verticais, o que requereu a adição de código para os representar corretamente, mas o problema a corrigir era a irregularidade da linha nos espaços intermédios com falta de planos no processo de mapeamento, por exemplo quando se tenta mapear os planos de uma superfície irregular como uma transação de janela para parede representada na Figura 31.

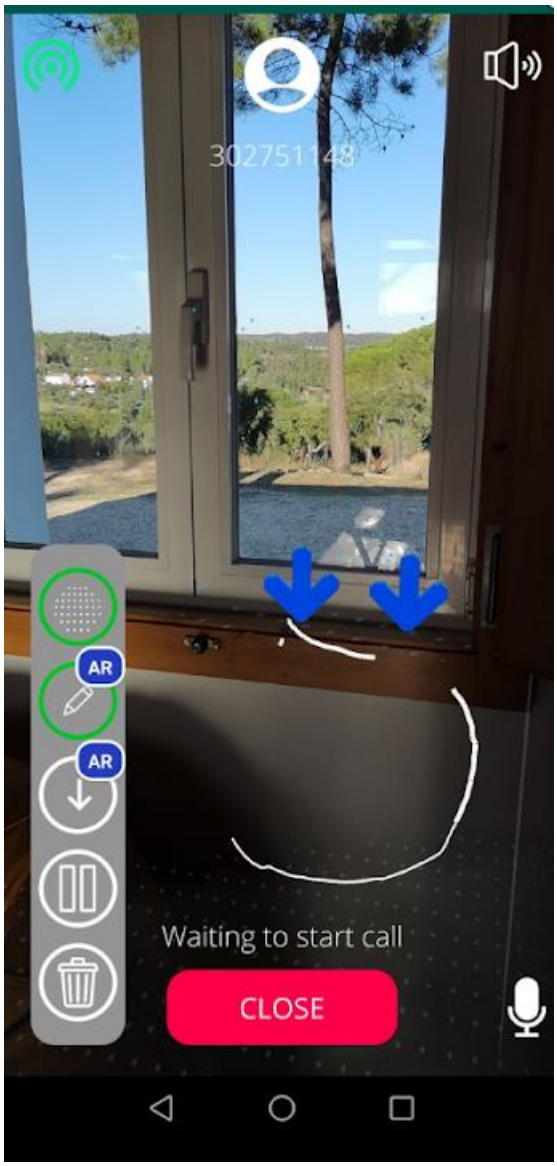

Figura 31 Problema de continuidade da linha

É aí que o mapa de profundidade pode ser usado para extrair a distância aproximada à superfície intersectada no ponto clicado, pois complementando o referido anteriormente cada pixel no mapa de profundidade é representado por um inteiro de 16 bits, em que os 13 bits menos significativos contêm a distância em milímetros à superfície estimada do plano de imagem da câmara (Google, 2021).

Para descobrir o ponto no mundo, é gerado um vetor 3D usado para intercetar a posição clicada, perpendicular ao dispositivo. O mesmo vetor é multiplicado pela distância de profundidade e convertido para a posição do mundo real.

Esta técnica, embora não suportada por todos os dispositivos, se tiver um dispositivo compatível com Depth API permite desenhar continuamente em qualquer ponto do ecrã, o algoritmo foi implementado desta forma:

- O dispositivo suporta a Depth API?
    - obter a distância aproximada com base no último mapa de profundidade e ponto tocado.
    - utilizar a distância para calcular a coordenada do mundo mapeada (ponto aproximado).
- O dispositivo suporta ARCore?
    - Executar um teste de colisão para encontrar o ponto de intersecção mais próximo com um objeto localizável (ponto exato)
    - No caso de uma seta, apenas o ponto exato é usado
    - No caso de desenho de linha, é utilizado de preferência o ponto exato. O ponto aproximado dos dispositivos compatíveis com a Depth API só é utilizado se o teste de colisão não encontrar o ponto.

Esta técnica melhorou de forma significativa o desenho, uma vez que se um dispositivo suporta a profundidade API o desenho será contínuo.

Para melhorar a probabilidade de sucesso, a última distância do último ponto de colisão é guardada e utilizada em caso de falha tanto no teste de colisão como na descoberta da profundidade no caso de desenho de linha, permitindo o desenho contínuo mesmo nos dispositivos incompatíveis com Depth API, o que significa que os dispositivos não suportados pela Depth API apenas requerem uma distância ao ponto da linha para se conseguir um desenho contínuo e sem interrupções.

Com o uso da última distância durante falhas de mapeamento, foi possível o resultado representado na Figura 32.

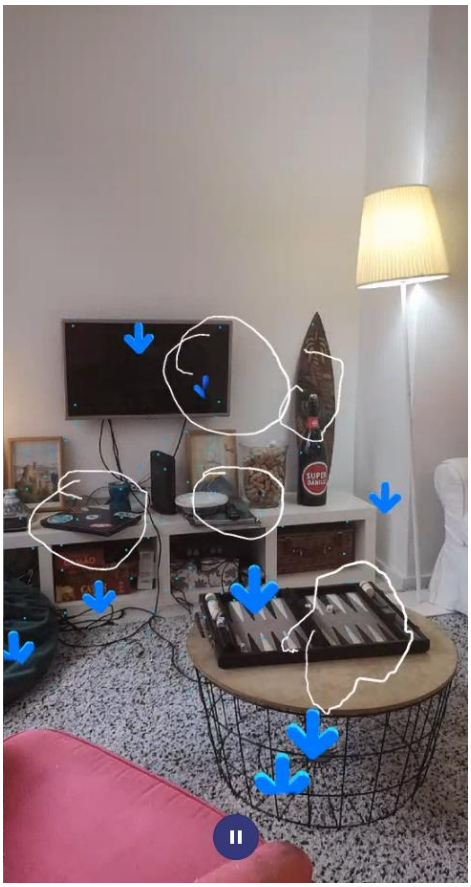

Figura 32 Depth API + teste de colisão na vista frontal

Embora a continuidade da linha tenha sido corrigida se olharmos para o lado tem alguma oscilação, como se vê na Figura 33, o problema foi derivado do uso de um dispositivo com suporte à Depth API tentar executar tanto o teste de colisão, como o cálculo pela distância utilizando o mapa de profundidade, e num ambiente irregular, há apenas alguns testes de colisão com sucesso tornando o resultado inconsistente.

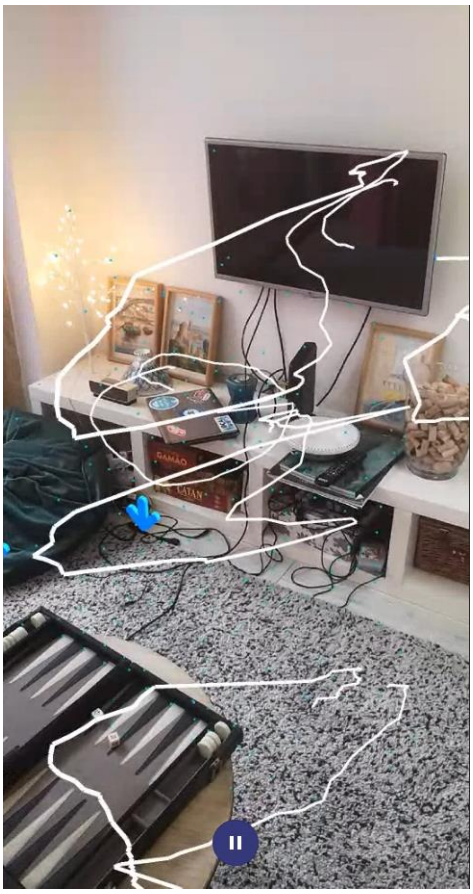

Figura 33 Depth API + teste de colisão vista lateral

Para resolver este problema ao desenhar uma linha em dispositivos com suporte à Depth API apenas é solicitada a posição obtida pelo cálculo com o mapa de profundidade, reduzindo a oscilação lateral.

3.5.3. Transferência de ficheiros

Um ficheiro é basicamente, um pedaço de dados codificados em blocos de "0" (zeros) e "1" (uns) conhecidos como *bytes*, e para enviar um ficheiro requer que esses blocos sejam lidos na fonte, enviados ao recetor e escritos pela mesma ordem.

Para partilhar ficheiros, foi adicionado um mecanismo de receção de ficheiros através do mecanismo de canal de dados do WebRTC, onde é enviada uma primeira mensagem, indicando os metadados de ficheiro, nomeadamente um número único de identificação, nome e tamanho. Após o envio de metadados várias mensagens são enviadas num formato binário que permitem a separação do ficheiro em blocos, onde os primeiros 36 *bytes* de cada bloco representam um *id* único e o restante do bloco representa o pedaço do arquivo. Este método de transferência de ficheiros é representado na Figura 34 onde enquanto os blocos

estão a ser recebidos permite ao utilizador uma noção do progresso representado numa barra de progresso na Figura 35.

Neste cenário, o progresso é definido como o número de *bytes* escritos sobre a quantidade total de *bytes* correspondentes ao tamanho do ficheiro, previamente partilhado na fase de metadados, consequentemente conhecido pelo recetor. O progresso é calculado no final da receção de cada pedaço do bloco de dados.

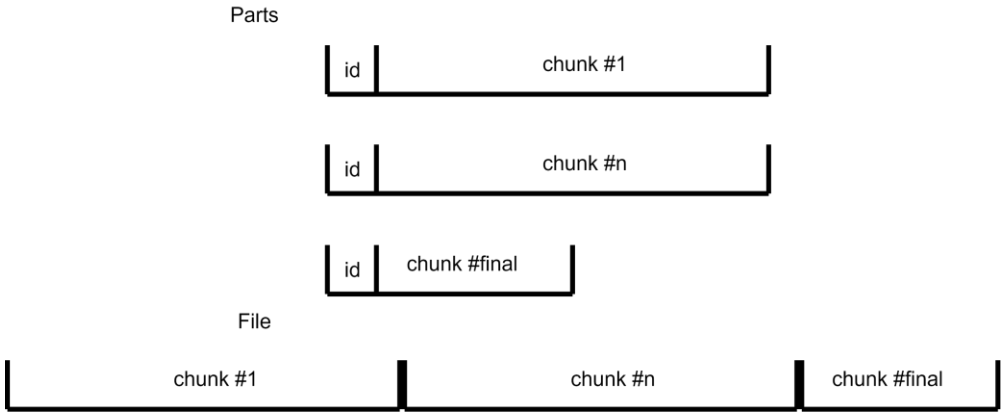

Figura 34 Estrutura de transferência de ficheiros por blocos

Figura 35 Barra de progresso de transferência de ficheiros

3.5.4. Modo de congelação

O problema com a AR ser usada para desenhar no mundo real num dispositivo portátil é o movimento que dificulta a utilização dessas capacidades no lado remoto especialmente em redes com atrasos significativos.

Uma solução é capturar uma imagem da câmara no momento necessário e desenhar em cima da imagem capturada em 2D.

A solução implementada extrai a imagem mais recente removendo a representação da grelha após 2 *frames* do congelamento, onde no primeiro *frame* a grelha é desativada e no segundo quadro a imagem é desenhada sem a grelha, algoritmo representado na Figura 36.

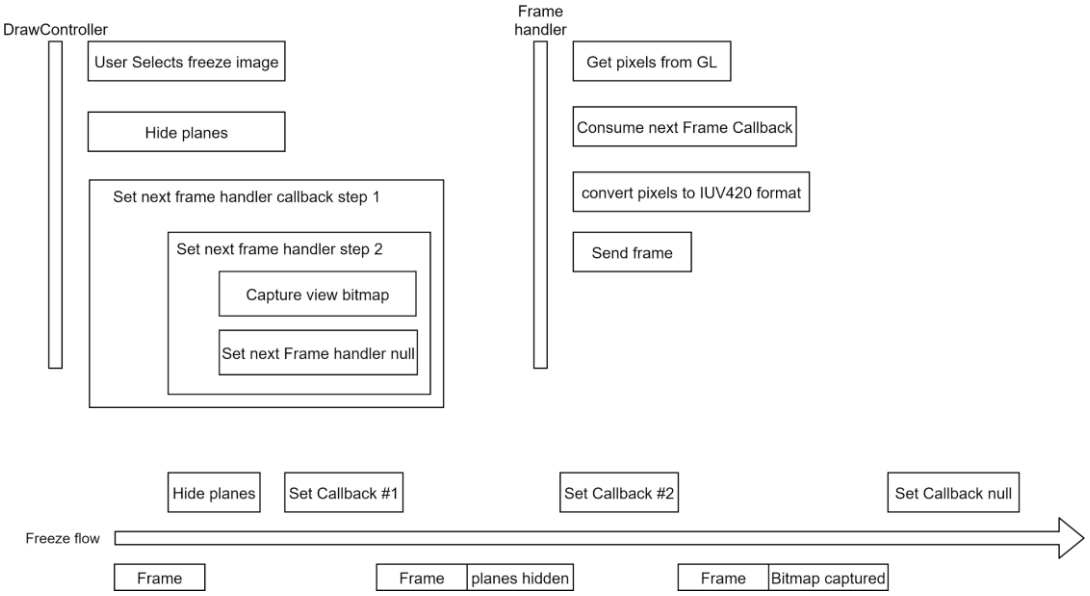

Figura 36 Fluxo de captura do modo de congelação

Isto introduziu alguns desafios uma vez que a implementação para enviar vídeo baseia-se na captura dos pixels desenhados, mencionado anteriormente, e posterior conversão num formato aceitável para o WebRTC. Por estes motivos foram consideradas duas abordagens para enviar o vídeo do modo congelado:

- Capturando o *bitmap* desenhado que leva cerca de 600 milissegundos (aproximadamente meio segundo) por *frame*.
- Utilizar a mesma abordagem que o vídeo de AR, obtendo os pixéis desenhados na tela.

Usando a primeira abordagem, o frame possibilitava o desenho através de métodos nativos disponíveis no Android chamado "*Canvas*", que possibilita o desenho da linha usando um caminho, uma largura e uma cor.

A segunda abordagem foi escolhida devido à transição entre diferentes modos não alterar a lógica de extração de *frames*, e o vídeo ficou mais fluído devido ao *frame* rate. Esta abordagem tem um grande inconveniente, o modo de congelação teve de ser desenhado utilizando a biblioteca OpenGL para simplificar a extração de pixéis.

A primeira abordagem foi tentar adaptar o código responsável pelo 3D disponível para o modo AR, e tentar adaptar-se a uma tela 2D, esta abordagem não foi possível no tempo disponível, pelo que a abordagem seguida foi a utilização da biblioteca de desenho baseada em OpenGL (ChillingVan, 2020). O inconveniente desta biblioteca é a função de desenho de caminho, presente na abordagem nativa, estar em falta. Para substituir a função

em falta, foi utilizada a funcionalidade de desenhar linha similar que separa o caminho em pares de pontos fazendo possível desenhar um caminho por partes.

Devido ao desenho de cada segmento da linha ser representado por 2 pontos, levou a outra limitação descoberta da biblioteca, uma vez que o caminho para desenhar cada segmento da linha piscaria 30 vezes por segundo ao exceder o limiar de cerca de 100 segmentos de linha, traduzido em cerca de 3 segundos de desenho antes que todos os desenhos começassem a piscar. Isto ocorre devido à sobrecarga adicionada por quebrar o desenho em partes menores que causa o número de pedidos de desenho a ser efetuados de 1 para centenas, demorando mais tempo a desenhar do que desenhando um traçado completo, intensificado pelo fato de ocorrer 30 vezes por segundo. Isto foi resolvido gerando uma nova imagem de fundo a cada 50 segmentos de linha com os mesmos desenhados por cima da última imagem de fundo, uma vez que 50 pontos demoram cerca de 2 segundos a serem gerados e a imagem de fundo leva cerca de meio segundo a ser gerada, dando tempo suficiente para dois desenhos de linha em simultâneo antes de esgotar os recursos do sistema e causando linhas a piscar.

3.5.5. Desenho 2D

De acordo com a "ARInsider", em 2019 existiam 400 milhões de dispositivos compatíveis com AR Core representando cerca de 16% quando comparado com os 2,5 mil milhões de dispositivos Android ativos no mesmo ano (Bol, 2019; Kerns, 2019).

Para aumentar a compatibilidade de dispositivos Android, foi introduzido um novo método de desenho para o fluxo de vídeo de uma forma semelhante ao modo de congelação, utilizando a biblioteca OpenGL. Os requisitos definidos para o desenho 2D foram: desenhar setas mantendo-as visíveis durante 2 segundos na posição do ecrã e o desenho de linha seria apresentado como uma cauda representada na Figura 37.

Figura 37 Linha em modo 2D

Com objetivo de desenhar o fluxo de vídeo da câmara com desenhos 2D por cima e de forma a manter a compatibilidade com o método de extração de pixels, e com preferência do uso da mesma biblioteca utilizada para o modo de congelação, escolheu-se OpenGL para desenhar o fluxo de vídeo na tela. Esta abordagem estende todas as limitações do modo de congelamento para além de não ser possível desenhar superfícies preenchidas em cima do fluxo da câmara já desenhado em OpenGL, limitando o desenho a uma linha de 1 pixel de largura, definido pela norma como sendo a largura máxima garantida por todos os dispositivos (Silicon Graphics Inc., 1991).

O desenho da seta é feito traçando-o linha a linha, como representado na Figura 38, onde em termos da seta foi dividido por duas partes, o corpo e a seta representado por uma linha colorida de um pixel de largura.

Desenhar o corpo da seta envolve encontrar os limites do corpo da mesma e desenhar linhas até que esses limites sejam preenchidos, mais detalhe acerca do algoritmo pode ser encontrado no Anexo C. O triângulo de seta para baixo envolve encontrar a zona inferior do corpo da seta e a extremidade esquerda da ponta da seta seguido de iterar movendo 1 pixel para baixo e um pixel dentro até a distância entre as fronteiras horizontais ser zero, algoritmo descrito no Anexo D.

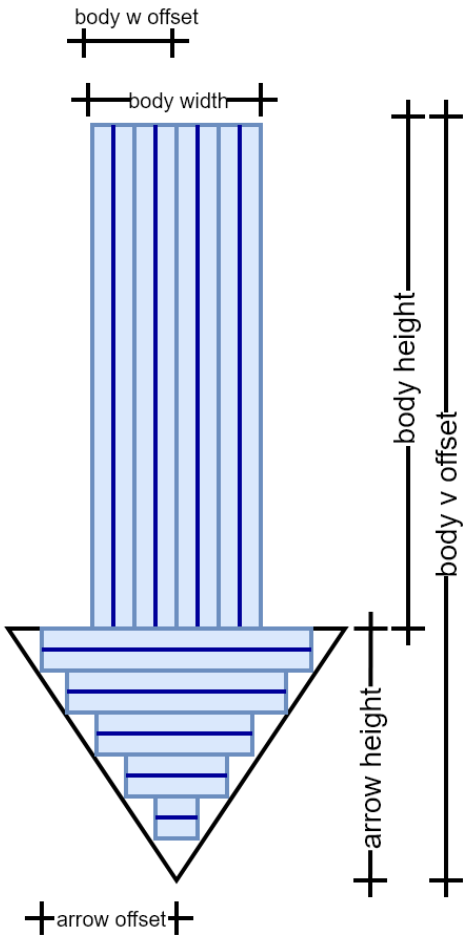

Figura 38 Mecanismo de desenho da seta 2D

No caso do desenho da cauda, conceptualmente quebrou-se a linha em partes menores com largura decrescente por cada ponto, a ideia geral é de recalcular os pontos cada vez que um evento de desenho é desencadeado, um ponto é inserido no início de uma lista e é agendada uma tarefa para um segundo no futuro para remover o último ponto da lista. A largura das linhas parciais é decidida na fase de desenho. Após alguma experimentação a equação de largura com base na posição do ponto na lista, é dada por $width = 20e^{-\frac{20}{512}x}$ exemplificado na Figura 39, sendo x o índice de linha. O desenho segue o mesmo princípio do modo de congelação onde apenas altera a largura com base no índice.

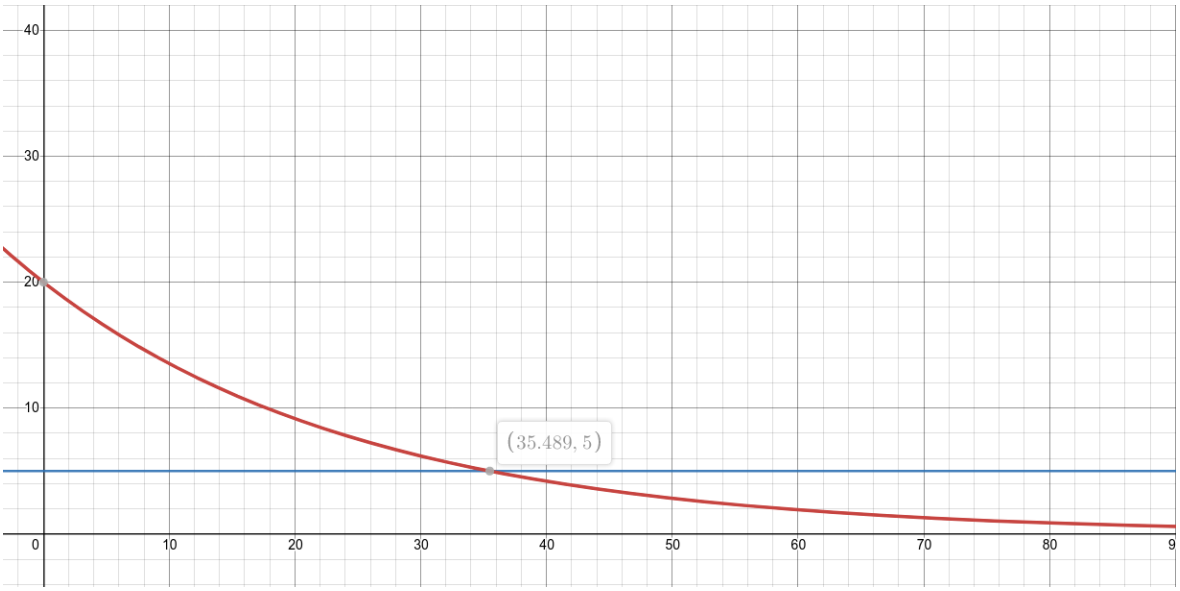

Figura 39 Função de cálculo de cauda de desenho com base na posição do ponto na lista

Devido à limitação da largura da linha, a cauda não estava a ser desenhada corretamente em alguns dispositivos, uma vez que se em qualquer ponto da cauda que excedesse a largura máxima suportada pela linha, desenharia a essa largura máxima, o que no caso de ser 5 pixels, só a partir do índice 36 é que começaria a diminuir em tamanho. Para contornar esta questão foram desenhadas várias linhas paralelas (*line_parts*) até que a largura desejada fosse atingida.

O primeiro passo foi descobrir qual é a largura máxima da linha do dispositivo, que através do GL_LINE_WIDTH_RANGE permite a descoberta da largura de linha mínima e máxima suportada pelo dispositivo.

O passo seguinte foi calcular o número mínimo de linhas necessárias para atingir a largura desejada (*needed parts*). Com o número de peças necessárias, é calculada a largura de cada parte da linha de modo que, quando multiplicadas pelas partes necessárias, tenha a largura desejada (*parts width*). Depois de conhecer as peças necessárias e a largura das partes, descobrimos o vetor de desfasamento (*offset vector*) que descreve um vetor perpendicular à linha desejada que representa o caminho que as partes da linha devem seguir lado a lado para alcançar a linha desejada.

Por outras palavras, para descobrir onde começar a desenhar as partes das linhas desde o ponto central pretendido, e para garantir que com as partes da linha lado a lado em cada linha terminaria a uma distância simétrica do centro, foi usada a seguinte fórmula

$\frac{partsWidth}{2}(neededParts - 1)$. Esta compensação será usada para escalar o vetor unitário transposto, melhor descrito como um vetor paralelo ao *line vector*, representado na Figura 40.

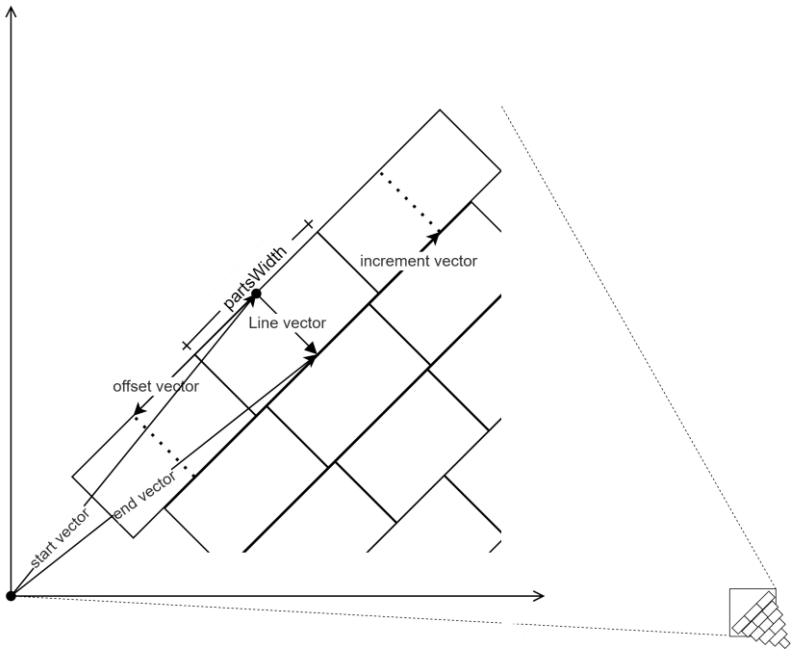

Figura 40 Esquema de composição do desenho de linhas em 2D

O vetor unitário transposto é dimensionado para cada parte necessária e a posição do início e fim de cada parte de linha é definida pela soma deste vetor escalonado com a posição de início e fim da linha desejada central, resultando na posição de cada parte de linha com base nos vetores de início e de fim.

Basicamente a linha foi dividida em partes de tamanho mais pequenas que a largura máxima suportada pelo dispositivo.

3.5.6. Servidor de multimédia

Devido a múltiplos fatores corporativos, foi necessário mudar a implementação para permitir:

- Chamada de conferência.
- Partilha de ficheiros multiutilizador.
- Registos históricos, incluindo gravações de chamadas.

- Vários utilizadores de desenho simultâneos.

A solução atual não está otimizada para este cenário uma vez que os números de escalabilidade são calculados por:

- Ligações RTC: $\frac{(n-1)n}{2}$
- Número de transmissões de vídeo: $(n-1)n$
- Número de transmissões de upload e download por dispositivo: $n-1$

onde *n* são os dispositivos RTC ligados, conhecidos como nós, exemplificados para quatro utilizadores na Figura 41.

Para resolver o registo histórico existem duas abordagens, escolha um nó para gravar e carregar toda a atividade ou incluir outro nó responsável por isso.

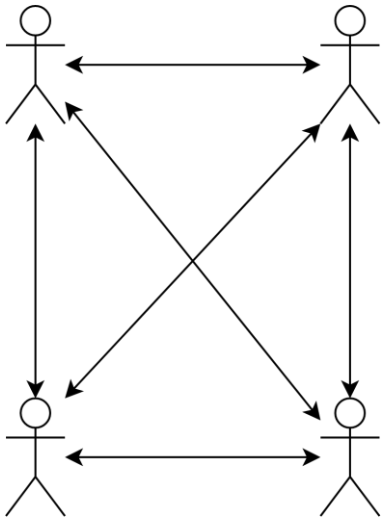

Figura 41 Diagrama de escalabilidade WebRTC para 4 dispositivos

A abordagem adotada foi a utilização de um servidor de multimédia, que em termos de escalabilidade, é calculado por:

- Ligações RTC: n
- Número de transmissões de vídeo: n^2
- Número de fluxos de upload por dispositivo: 1
- Número de fluxos de download por dispositivo: $n-1$

Onde *n* são os dispositivos RTC ligados exemplificados para quatro utilizadores na Figura 42.

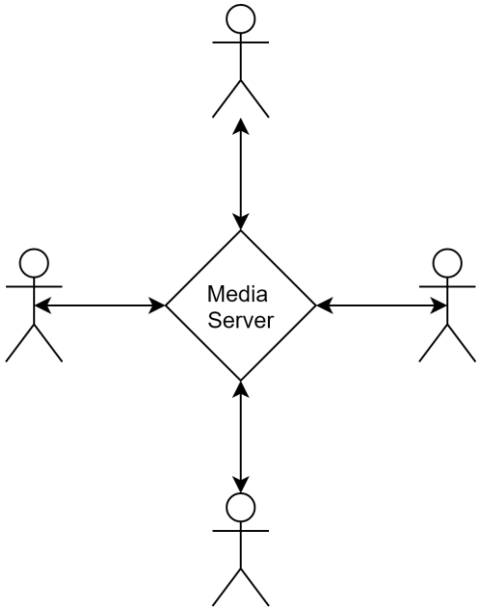

Figura 42 Diagrama de escalabilidade do servidor de multimédia para 4 dispositivos

Analisando para os números de escalabilidade representados na Figura 43, embora as ligações de fluxo RTC sejam mais elevadas com um servidor multimédia, os números de transmissões por dispositivo são significativamente mais baixos, permitindo uma utilização inferior de largura de banda, juntando-se ao facto de o dispositivo móvel apenas enviar o vídeo, reduzindo a sua transmissão a um único fluxo de vídeo.

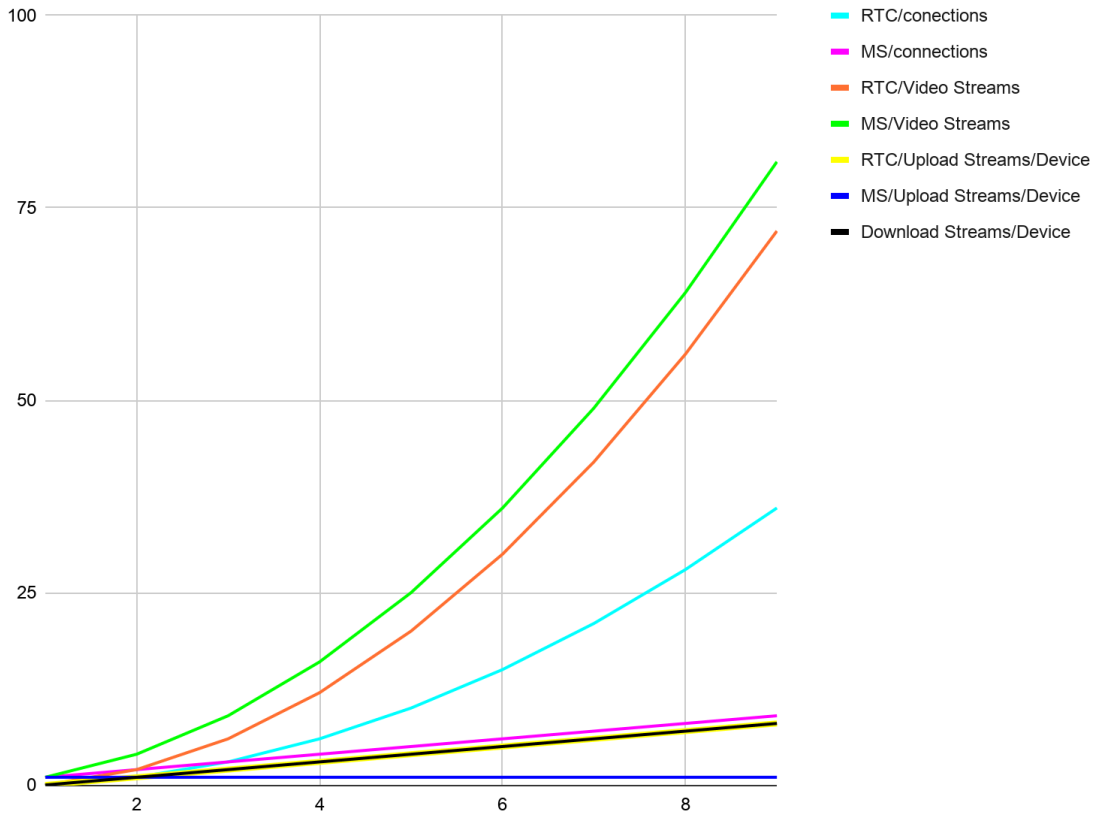

Figura 43 Representação gráfica para ligações comparando WebRTC/Servidor de multimédia

Em comparação com o WebRTC, há mais ou o mesmo atraso usando o servidor de multimédia, uma vez que o WebRTC só precisa de um intermediário de vídeo no máximo, mas pode usar o servidor STUN para encontrar uma rota mais direta. A abordagem de servidor multimédia tem sempre um representante intermediário.

Modularidade

Aproveitando a mudança para permitir uma migração mais fácil no futuro, o módulo WebRTC foi extraído para uma interface personalizada que separa a zona da transmissão de vídeo em fases distintas, representada na Figura 44.

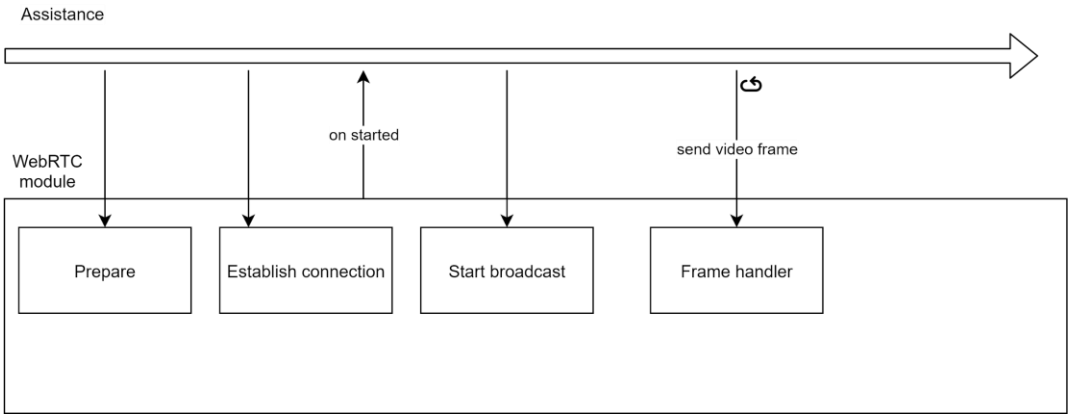

Figura 44 Ciclo de vida do módulo WebRTC

Existem três fases principais para iniciar a ligação RTC:

1. *Prepare*, inicia a abertura do ecrã de assistência, onde normalmente são criados os objetos necessários.
2. *Establish connection*, iniciada após o *prepare*, onde todos os serviços necessários devem ligar-se, por exemplo, *sockets*, servidor de multimédia, etc.
3. *Start Broadcast*, iniciada após o serviço informar a sua prontidão, iniciando a faixa de vídeo e áudio, e inicia a extração e envio de pixeis de vídeo.

Após a fase de transmissão iniciar, cada *frame* é lidado dentro do módulo RTC, facilitando uma mudança futura de serviço.

Cada módulo tem eventos para informar a aplicação sobre mensagens recebidas, alterações remotas de estado do utilizador e um evento para informar quando o serviço está pronto para começar a transmitir.

Migração de canais de dados

Antes da migração, foram utilizados vários canais de dados com diferentes rótulos para cada ação, nomeadamente:

- point para o desenho de seta remota,
- draw para desenho de linha remota,
- delete para apagar todos os desenhos,

- file para ficheiros de descarregamento ou os seus metadados dependendo da codificação,
- notify para informações gerais,
- notify_ready para a primeira mensagem indicando as informações do emissor como o tamanho do ecrã e características do dispositivo.

O fornecedor do servidor de multimédia permite a criação de canais de dados, mas gerados com identificadores não descritivos, pelo que foi adotada uma convenção para enviar a mensagem em formato JSON, com uma propriedade chamada "channel" preenchido com o rótulo da mensagem e uma propriedade nomeada "message" com o conteúdo da comunicação. Em última análise, o modelo utiliza a mesma estrutura que o previsto para os canais de dados, mas o método de transmissão é semelhante a um tópico de paradigma de *publish/subscribe* que permite aos assinantes com o expresso interesse num tipo de evento a ser notificado acerca dos mesmos. Este é gerado por um editor ao subscrever um tópico T, tornando-se membro de um grupo T, e a publicação de um evento sobre o tema T, que se traduz em conformidade na transmissão desse evento entre os membros de T (Eugster et al., 2003). O *id* do canal foi usado como tópico e o número do emissor como identificação.

Integração HoloLens

Devido ao servidor de multimédia escolhido não suportar o Windows com ARM, a arquitetura presente no *HoloLens 2*, foi necessário encontrar uma solução para suportar esse dispositivo. A abordagem principal foi utilizar um retransmissor dos dados WebRTC para o fornecedor de serviços multimédia.

A primeira abordagem foi fazer um servidor JavaScript personalizado. Devido ao fornecedor suportar *web* através de um SDK JavaScript, o servidor conteria:

- Servidor WebRTC para comunicações vídeo/áudio com os *HoloLens.*
- Servidor WebSocket para troca de dados de sinalização.
- Servidor TURN/STUN para a descoberta do WebRTC.
- SDK do fornecedor de serviços multimédia para o retransmitir aos outros utilizadores.

O principal problema nesta abordagem deveu-se ao facto da biblioteca do fornecedor de multimédia estar excessivamente ligado aos elementos visuais do navegador, tornando-o incapaz de funcionar num servidor Node.js.

A abordagem seguinte envolveu a transferência de cada *frame* num formato de lista de *bytes*, transferi-lo através de *sockets* e retransmiti-lo para o fornecedor de servidor de multimédia. Em primeiro lugar foi testado o protocolo TCP, embora não exista qualquer limite no respetivo RFC (Fette <ifette+ietf@google.com>, n.d.) a implementação dividiria os dados em pedaços de 64 KB. Isto causaria um atraso significativo na chegada do *frame*, bem como a adição de uma carga significativa nos recursos do *HoloLens*, introduzindo a necessidade de limitar a contagem de *frames*, uma vez que o tempo que cada *frame* levaria a ser transferido excederia o tempo necessário para que o fluxo de vídeo fosse percetivelmente fluido.

O protocolo UDP, apesar de ter ajudado na carga de transmissão, introduziu novos desafios para permitir a transmissão de dados, uma vez que requer uma ligação direta entre dispositivos que não poderia ser assegurado num ambiente empresarial.

Houve uma tentativa de utilizar o protocolo "Wi-Fi Direct" que é uma especificação para a conectividade direta de dispositivos onde um dispositivo descobre o outro tal como descobriria uma rede sem fios, introduzindo uma palavra-passe ou pressionando um botão e os dispositivos ficam ligados (Neagu, 2021; Wi-Fi Alliance, 2020) mas o seu SDK para os *HoloLens* é incompatível com ARM.

A abordagem implementada envolveu a utilização da biblioteca WebRTC entre os *HoloLens* e o dispositivo Android, onde os *frames* de vídeo em vez de serem desenhados no ecrã do dispositivo Android, seriam transmitidos para o fornecedor de serviços multimédia. O principal desafio nesta abordagem foi a conversão de formato, que não é diretamente compatível com o servidor multimédia, resultando numa imagem em tons de cinza necessitando de uma biblioteca nativa Android para converter para o formato RGB suportado pelo fornecedor de serviços multimédia.

3.5.7. Dispositivos simples

Os *smart glasses* são dispositivos que permitem ao utilizador receber assistência com as mãos livres para realizar as instruções indicadas pelo operador. A equipa acabou por

chamar estes dispositivos de simples, uma vez que necessitaria de uma versão simplificada da aplicação.

Foram consideradas duas abordagens para suportar este tipo de dispositivos, criar uma nova aplicação dedicada a cada dispositivo simples, ou adaptar a atual para suportar este tipo de dispositivos.

Uma vez que os dispositivos simples suportados são todos baseados em Android, devido ao tamanho da equipa, em conjunto com a dificuldade na manutenção, a primeira abordagem foi descartada. A abordagem seguida foi adaptar a aplicação existente, criando um fluxo de execução paralelo quando este tipo de dispositivos é detetado, reutilizando o maior número possível de componentes para reduzir problemas futuros.

A primeira iteração foi concebida para funcionar com um dispositivo com três botões, e a navegação simplificada para 2 ecrãs, um da sessão de assistência e outro de Juntar à sessão, representado na Figura 45, similar ao atual ecrã de entrar, mas adaptado para conter apenas um botão de entrar e a interface adaptada para permitir a navegação usando botões direcionais.

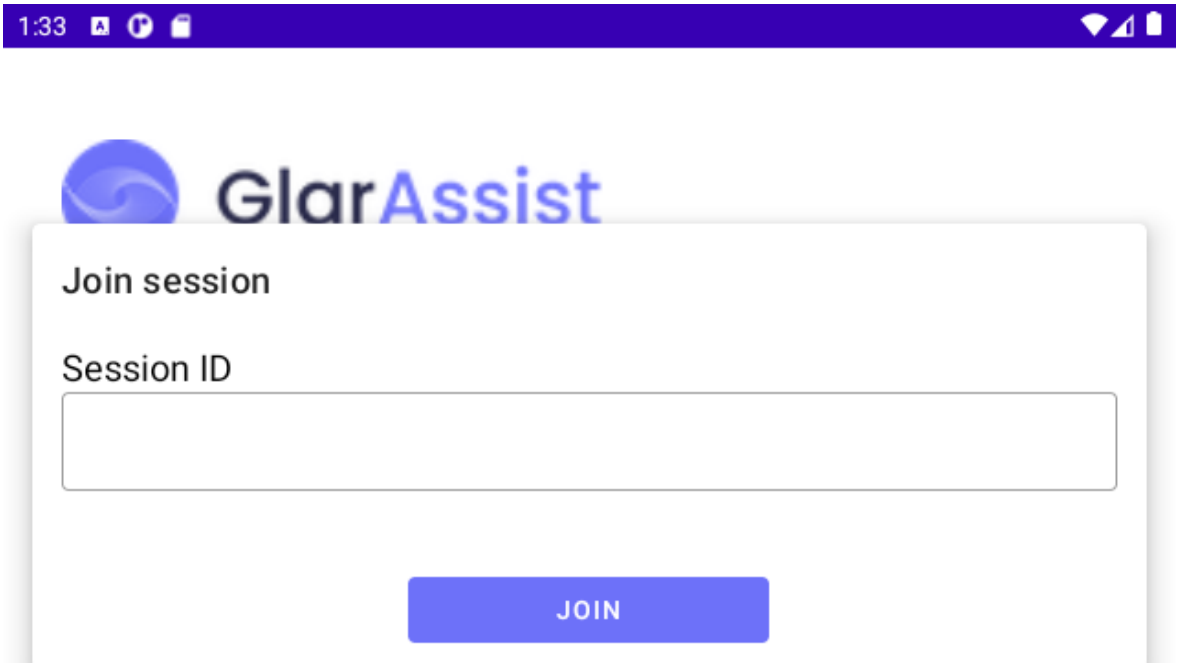

Figura 45 Ecrã para entrar na sessão em dispositivos simples

O segundo ecrã é um ecrã de assistência simplificado representado na Figura 46, composto pelo fluxo de vídeo com anotações, apenas desenhável pelo lado remoto, e botões de lanterna, microfone e congelação.

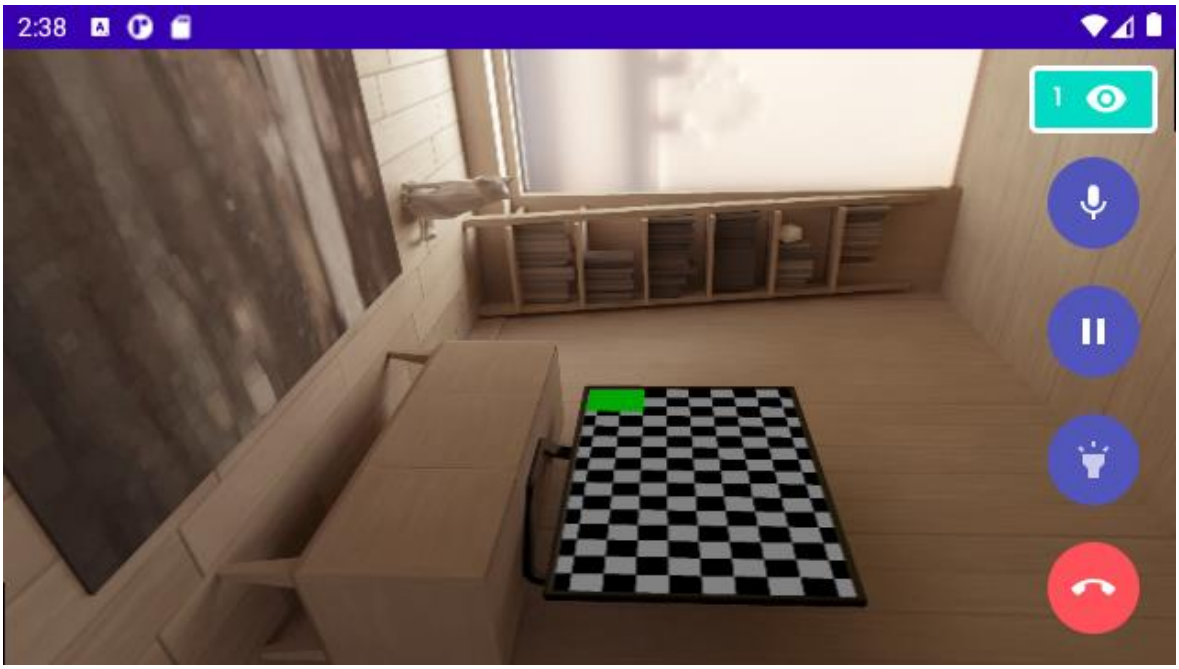

Figura 46 Ecrã de assistido em dispositivos simples

Os dispositivos simples previstos apenas funcionam em modo 2D, modo este bloqueado na orientação de retrato, devido à rotação do ecrã em assistência não estar implementada no momento de desenvolvimento desta funcionalidade, isto apressou a necessidade de implementar a mesma, que foi conseguida nos *shaders* gráficos multiplicando a posição base com a matriz de rotação se o dispositivo for rodado (Overvoorde, 2019).

Embora a navegação do ecrã de entrada na chamada tenha sido implementada com o tornar do botão entrar como alvo de foco e colocando uma ação de submissão no campo de texto para o número de chamada, o ecrã assistido tinha os ícones de navegação com um design fora do normalizado representado na Figura 47.

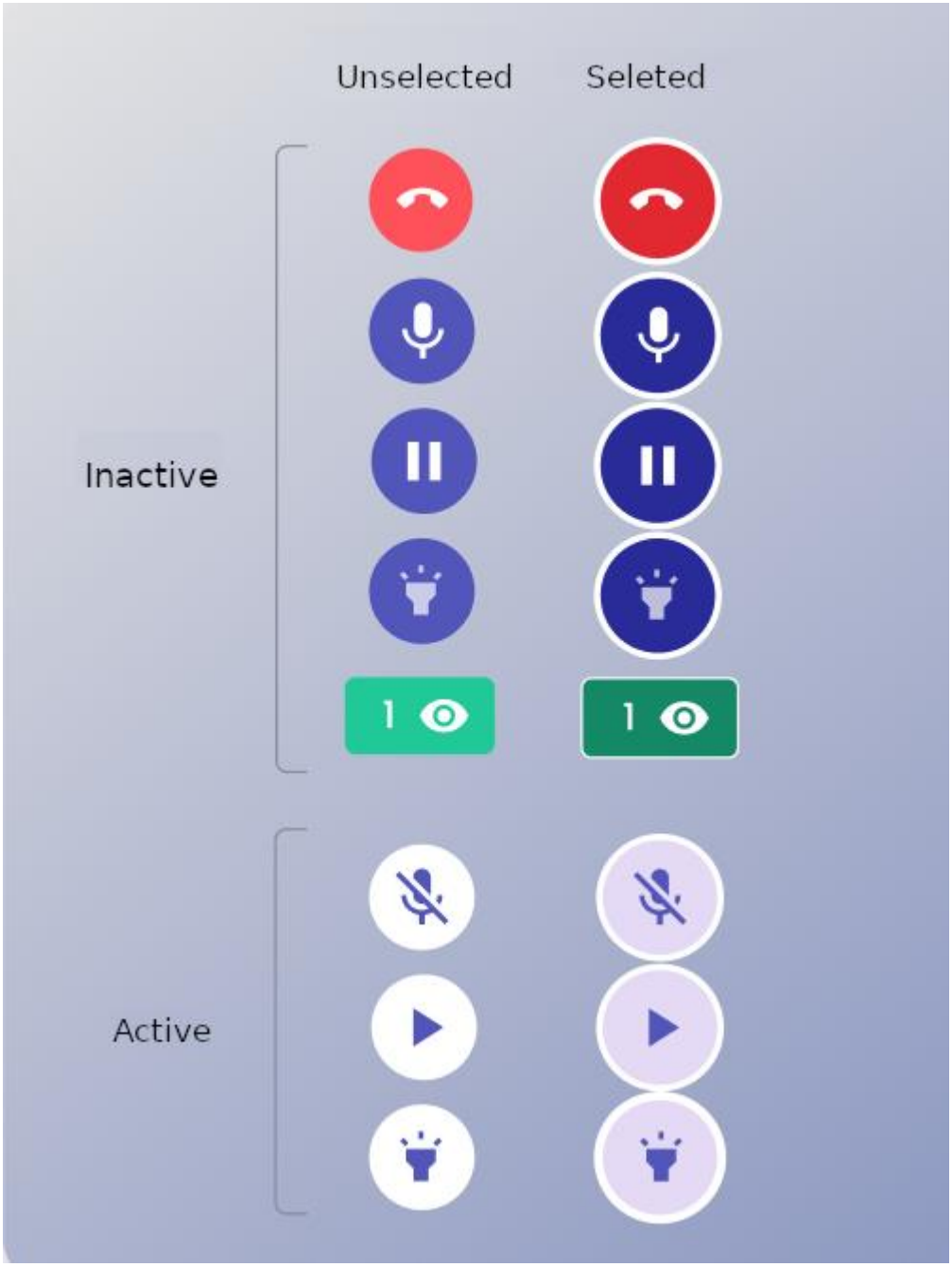

Figura 47 Design de botões de dispositivos simples

As cores e ícones foram implementados programaticamente, modificando, consoante o estado dos mesmos:

- A imagem e a cor do ícone.
- A cor de fundo do botão para o estado pretendido.
- A borda branca quando selecionada, implementada colocando por baixo de cada botão um anel redondo ligeiramente maior do que o botão cuja visibilidade seria dependente do seu estado selecionado.

Na Figura 48 está representado um protótipo de alta-fidelidade com o botão de microfone ativo, o botão de pausa ativo e selecionado, e o botão da lanterna selecionado e inativo.

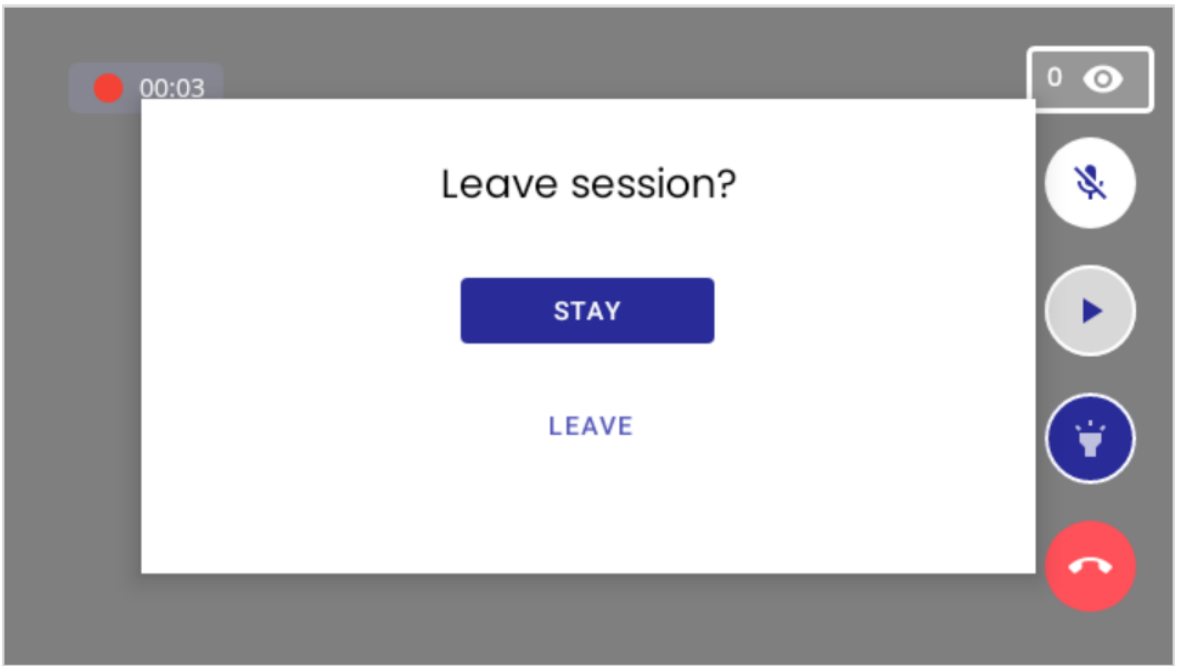

Figura 48 Protótipo de layout de botões para dispositivos simples

O próximo desafio veio com um dispositivo simples que, assim como o dispositivo anterior, contem um pequeno monitor na frente de um olho, mas só aceita a voz como interação.

Este dispositivo usa vários auxílios para evitar que o utilizador se perca durante a navegação por voz, nomeadamente:

- Um menu de ajuda adaptado à aplicação atualmente aberta,
- Os botões ou têm um número predefinido atribuído ou comandos de voz ao lado do botão correspondente, brevemente identificados na abertura do menu de ajuda com uma etiqueta ao lado do mesmo.

O principal desafio com esta abordagem foi trabalhar com tradução, múltiplos comandos, e a ajuda para cada botão, uma vez que não seriam compatíveis uns com os outros porque:

- A tradução não foi pensada pelo fabricante na documentação disponibilizada, uma vez que o comando é definido como um texto no botão ou uma etiqueta

escondida contendo vários comandos. O único método padrão seria usar etiquetas de botões com o sistema de tradução nativo ou mudar o conteúdo de etiqueta programaticamente cada vez que qualquer botão for clicado, ou qualquer estado for alterado.

- Vários comandos num botão são possíveis e é documentado pelo fabricante.
- Os comandos de ajuda eram possíveis, mas só eram utilizáveis se fossem utilizados um por botão, uma vez que geraria artefactos visuais tentando mostrá-los todos no lado de cada botão.

Para mitigar estas incompatibilidades, foi:

- Criado o conceito de comandos principais e secundários onde foram criados vários botões empilhados com tamanho de 1 por 1 pixel, quase invisíveis, que conteriam os comandos secundários sem sobreposições visuais, e os botões visíveis laterais continham o comando principal.
- Reiniciados os comandos cada vez que o estado mudava, por exemplo, comandos como congelar/descongelar ou lanterna ficaria no botão primário e pausa/direto ou ligar/desligar luz no secundário e o os comandos alternariam para os seus opostos e com base no estado atual da aplicação.

Estes passos resolvem todos os problemas levantados, pois:

- Múltiplos comandos são suportados através da separação dos mesmos pelo principal e outros.
- A tradução seria colocada em tempo de execução, permitindo buscar no idioma atual aquando da mudança do estado,
- Os artefactos gerados pelos múltiplos comandos num botão desapareceram visto que o botão principal visível apenas contem o comando principal.

Esta abordagem por consequência é compatível com os comandos de ajuda preenchendo o mesmo com os comandos principais na mudança de estado.

Outro tipo de dispositivo são os óculos transparentes, um tipo de dispositivo que contem lentes transparentes e a imagem é projetada nessas mesmas lentes, que levantam um potencial problema da abordagem a seguir. Coloca-se o mundo no topo da visão do utilizador ou apenas desenhamos as anotações no ecrã? este último seria um grande desafio uma vez

que o fluxo da aplicação é desenhar para o ecrã e, em seguida, extrair o resultado da renderização para enviar através do fluxo de vídeo. Se quisermos desenhar apenas as anotações no ecrã dos óculos, teria que se renderizar anotações com vídeo para os espectadores remotos e as anotações apenas para o utilizador local, trazendo problemas de alinhamento e uma grande penalidade de desempenho. Foi decidido não avançar com esta abordagem, mas foi recomendado implementar pelo fabricante uma visão colapsada para estes dispositivos específicos por forma a removerem do foco visual primário do operador, que também não foi implementado devido à necessidade de migrar todos os *shaders* para um VBO, brevemente descrito em capítulos anteriores, antes de renderizar para o ecrã o que exigiria muito mais esforço do que o disponível para a adaptação.

3.5.8. Gravação

Devido a uma decisão corporativa, foi necessário implementar um mecanismo para ter registos multimédia da assistência. Isto foi conseguido utilizando uma API do servidor de multimédia. A integração com o *backend* foi direta, um pedido de REST para iniciar e outro para parar, notificando os utilizadores remotos da alteração do estado de gravação. Após algumas iterações deste mecanismo no final o servidor acabou por ser responsável por informar através do WebSocket todos os utilizadores na sala que a gravação foi iniciada por um utilizador.

Ao desenvolver-se, assumiu-se que, existindo apenas uma fonte de vídeo na chamada o servidor media colocaria a referida fonte como vista principal, mas ao testar foi encontrado um comportamento inesperado onde, por vezes, o vídeo seria muito menor do que o esperado. Mais tarde foi descoberto que, por omissão, o fornecedor de serviços multimédia criaria o layout como uma chamada de vídeo e colocaria no fundo o utilizador que falou por último, mesmo não contendo uma faixa de vídeo, representada na Figura 49. Para mitigar este problema, o utilizador móvel enviaria um pedido REST para criar um *layout* personalizado que preenchesse todo o ecrã ao entrar na sala ou quando alguém começasse a gravar e não deixasse espaço para os outros utilizadores que não fossem de vídeo.

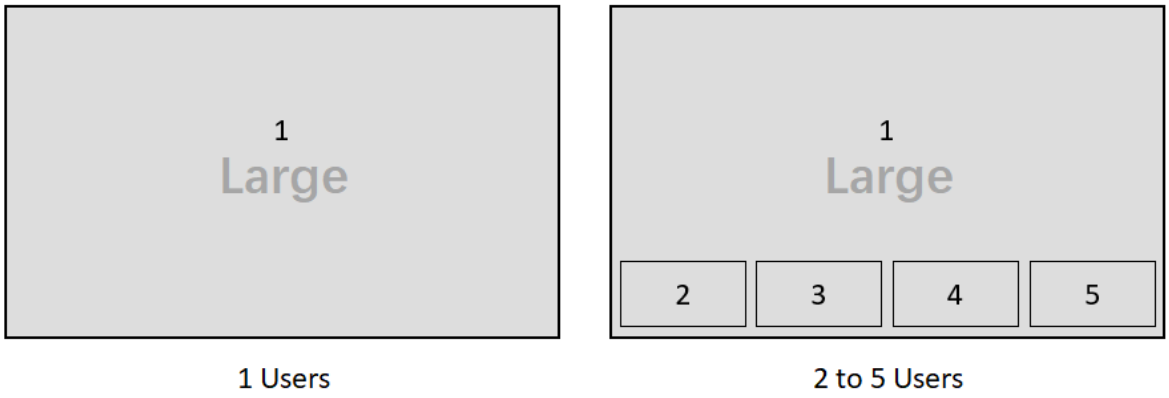

Figura 49 Esquema de gravação padrão do servidor de multimédia

Devido a esta correção, isto levou a um *bug* que causava pedidos recursivos ao WebSocket porque não existia relação entre o utilizador do WebSocket e o utilizador REST apesar de serem o mesmo, fazendo com que quando alguém acionasse a gravação todos os utilizadores presentes na chamada receberiam o evento de atualização, incluindo o utilizador que iniciou a gravação. Quando um dispositivo Android recebe uma atualização deste tipo faz um pedido para atualizar o *layout* colocando o seu vídeo na frente, que geraria outro evento de atualização. Para resolver este problema, os pedidos de registo REST foram migrados para pedidos de WebSocket e o utilizador que enviaria os pedidos não receberia um evento de atualização.

Depois de algum tempo em produção, notou-se que várias gravações ficaram por várias horas, sem qualquer utilizador lá. Para mitigar esta situação, foi estabelecido um tempo de gravação máximo e, em vez de cada gravação fazer parte da sessão, seria parte do utilizador que tornou a gravação indisponível para utilizadores não registados, pois reentrando na sala não seria capaz de parar a sua gravação, uma vez que o número de identificação da sessão é único cada vez que o cliente entra na sala.

Outro tipo de gravação implementada foi a gravação local também conhecida como gravação tutorial, destinada a usar todos os métodos de representação presentes como AR, 2D, e de congelamento para gravar um vídeo tutorial local para futura partilha.

Para manter a compatibilidade, a gravação local é mais um fornecedor que extrai as *frames* da mesma forma que enviaria para o servidor multimédia. A parte de gravação foi conseguida usando a API de Media Recorder, disponível desde o primeiro lançamento Android.

O primeiro obstáculo notado foi que ao extrair os *frames* estava a ter um impacto significativo no desempenho uma vez que por cada *frame* ter-se-ia que extrair os pixels do vídeo, converter em *Bitmap*, um formato de imagem em memória, e colocá-lo no arquivo de vídeo. O desempenho foi melhorado através da paralelização do processo em que o desenho e a extração foram efetuados no fluxo principal, mas o resto da operação foi feito num fluxo em recursivo separado, representado na Figura 50.

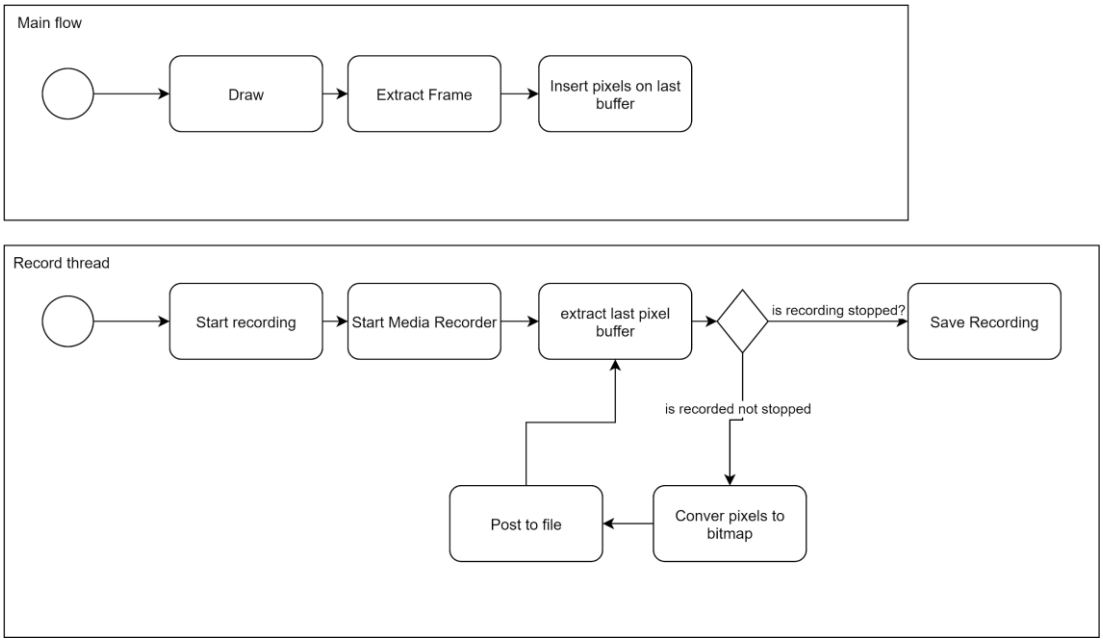

Figura 50 Esquema de fluxo otimizado para gravação offline

Para manter a compatibilidade com um leque mais alargado de dispositivos, os descodificadores de áudio e vídeo foram definidos como padrão e o microfone definido como modo de reconhecimento de voz, aproveitando vários microfones se presentes e recuando para uma simples gravação de microfone, caso contrário. Foi definida uma resolução de 1280x720px, uma vez que resoluções fora do padrão muitas vezes causariam comportamentos inesperados.

3.5.9. Resultados

Tendo em conta os requisitos inicialmente propostos, a transmissão de vídeo com AR incluído, foi possível através do desenho e extração de pixels usando OpenGL como descrito nas secções 3.5.2, 3.5.4 e 3.5.5.

Em termos de suporte para desenho AR no lado do assistente, este foi conseguido usando o ARCore da Google para encontrar a posição clicada do mundo, OpenGL para

desenhar descrito na secção 3.5.4, e o desenho remoto foi conseguido usando o modelo de *publish/subscribe* de mensagens mencionado na secção 3.5.6.

A comunicação áudio bidirecional foi fornecida pelo SDK do fornecedor multimédia descrito na secção 3.5.6.

A recuperação automática de comunicações em condições adversas, é tratada pelo SDK do servidor de multimédia, mas existem 2 abordagens para futura melhoria nesta frente:

- Reduzir os *frames* por segundo com largura de banda baixa.
- Congelação de alta qualidade onde o modo de congelação é sugerido ao utilizador aquando baixa largura de banda, a imagem enviada uma vez para o lado remoto e o desenho feito em simultâneo tanto no assistido como nos assistentes em vez de ser transmitido em vídeo, permitindo que a transmissão de vídeo seja desligada.

A adaptação bem-sucedida para trabalhar com o desenvolvimento paralelo em curso da versão *Web* foi conseguida através da definição de estruturas e protocolos entre as duas partes para manter uma compatibilidade na implementação, nomeadamente:

- Na comunicação de multimédia, ao habilitar a definição de interoperabilidade do Servidor de multimédia.
- Na troca de mensagens, estabelecendo previamente o protocolo de comunicação tanto por WebSocket como por *publish/subscribe*, assim como a pré definição da estrutura da mensagem a ser enviada e compreendida por ambas as partes.

Conclusão

Este é um projeto em curso, no momento de escrita a aplicação está em produção num estado estável, e capaz não só de desenhar anotações no mundo real usando AR, mas também em superfícies 2D para dispositivos menos capazes ou apenas por conveniência.

Um servidor de multimédia intermediário como uma metodologia de comunicação, embora mais cara, permite uma escalabilidade linear, onde em contrapartida o WebRTC tem uma escalabilidade exponencial devido à sua natureza ponto a ponto.

A tecnologia Depth API permite um processo de mapeamento quase instantâneo, mas vem com o custo de precisão.

Covid foi um grande impulsionador para lançar software de assistência remota, como analisado no capítulo de estado da arte.

O HoloLens é uma tecnologia impressionante, mas devido às decisões da Microsoft, é, no momento de escrita impraticável para trabalhar com esta aplicação devido à incompatibilidade da biblioteca do servidor de multimédia com a arquitetura do dispositivo, tendo em consideração que a única alternativa encontrada foi usar o dispositivo Android como retransmissor, e o plano do provedor do servidor de multimédia para lançar a biblioteca num curto espaço de tempo, decidiu-se travar o desenvolvimento deste dispositivo.

É particularmente desafiante diagnosticar código para OpenGL no Android devido às mensagens de erro que estão representadas como códigos numéricos, tornando a descoberta do ponto de erro específico demorado.

Alguns exemplos de melhorias planeadas, são os comandos de voz do dispositivo simples serem agnósticos do fornecedor, permitindo uma melhor compatibilidade e diversidade em línguas. Outra alteração planeada é os eventos do estado da gravação serem fornecidas pelo prestador de serviços de gravação, permitindo ter um estado atual de gravação mais preciso. O próximo grande marco envolve o desenvolvimento em iOS, que utiliza o ARKit e tem suporte para lanterna no modo AR.

Referências

ARCore overview | Google Developers. (n.d.). Retrieved 2 December 2020, from https://developers.google.com/ar/discover

Arachchi, S. A. I. B. S., & Perera, I. (2018). Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management. *2018 Moratuwa Engineering Research Conference (MERCon)*, 156–161. https://doi.org/10.1109/MERCon.2018.8421965

ARCore overview | Google Developers. (n.d.). Retrieved 2 December 2020, from https://developers.google.com/ar/discover

Augmented Reality for the Web: Using WebAR to Speed Up Cross-Platform Mobile Development. (2021, January 15). Oursky Posts. https://blog.oursky.com/2021/01/15/augmented-reality-for-the-web-using-webar-to-speed-up-cross-platform-mobile-development/

Baltrušaitis, J. (1977). Anamorphic art / by Jurgis Baltrušaitis ; translated by W. J. Strachan. In *Anamorphic art*. Harry N. Abrams.

Baum, L. F. (Lyman F. (1996). *The Master Key An Electrical Fairy Tale Founded Upon the Mysteries of Electricity*. http://www.gutenberg.org/ebooks/436

Bol, M. (2019, May 13). *ARCore Reaches 400 Million Devices*. AR Insider. https://arinsider.co/2019/05/13/arcore-reaches-400-million-devices/

Brownlee, J. (2019). *Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python*. Machine Learning Mastery.

Caballero, M. L., Chang, T.-R., Menéndez, M., & Occhialini, V. (2010). Behand: Augmented virtuality gestural interaction for mobile phones. *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, 451–454. https://doi.org/10.1145/1851600.1851704

Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., Damiani, E., & Ivkovic, M. (2011). Augmented reality technologies, systems and applications. *Multimedia Tools and Applications*, *51*(1), 341–377. https://doi.org/10.1007/s11042-010-0660-6

ChillingVan. (2020, August 1). *Android-openGL-canvas*. https://github.com/ChillingVan/android-openGL-canvas

Copans, V. (2020). *How Data Will Drive the Shift to Hybrid Events of Live and Virtual*. com

Early, S. H., Kuzma, A., & Dorsey, E. (1993). The VideoPhone 2500—Video Telephony on the Public Switched Telephone Network. *AT&T Technical Journal*, *72*(1), 22–32. https://doi.org/10.1002/j.1538-7305.1993.tb00519.x

Eugster, P. Th., Felber, P. A., Guerraoui, R., & Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Computing Surveys*, *35*(2), 114–131. https://doi.org/10.1145/857076.857078

Farshid, M., Paschen, J., Eriksson, T., & Kietzmann, J. (2018, July 26). *Go boldly!: Explore augmented reality (AR), virtual reality (VR), and mixed reality (MR) for business | Elsevier Enhanced Reader*. https://doi.org/10.1016/j.bushor.2018.05.009

Fette <ifette+ietf@google.com>, I. (n.d.). *The WebSocket Protocol*. Retrieved 19 February 2021, from https://tools.ietf.org/html/rfc6455

Fundamental concepts | ARCore | Google Developers. (n.d.). Retrieved 6 December 2020, from https://developers.google.com/ar/discover/concepts

Google. (2021, August 11). *Depth API overview for Android | ARCore | Google Developers*. https://developers.google.com/ar/develop/java/depth/overview

Google Creative Lab. (2018). *Just a Line by Google Creative Lab | Experiments with Google*. https://experiments.withgoogle.com/justaline

Hampel, A., Costa, L., Mak, E., & Suthers, B. (2020, October 7). Live vs Virtual vs Hybrid Events: Which is best for you? *Alive Events Agency*. https://aliveeventsagency.com.au/live-vs-virtual-vs-hybrid-events/

Han, J., & Smith, B. (1997). CU-SeeMe VR immersive desktop teleconferencing. *Proceedings of the Fourth ACM International Conference on Multimedia*, 199–207. https://doi.org/10.1145/244130.244199

Harris, M. (2013). The Hologram of Tupac at Coachella and saints: The value of relics for devotees. *Celebrity Studies*, *4*(2), 238–240. https://doi.org/10.1080/19392397.2013.791054

Heilig, M. L. (1962). *Sensorama simulator* (United States Patent No. US3050870A). https://patents.google.com/patent/US3050870A/en

I420 yuv pixel format. (n.d.). Retrieved 29 November 2020, from https://www.fourcc.org/pixel-format/yuv-i420/

Kennedy, R. S., Lane, N. E., Berbaum, K. S., & Lilienthal, M. G. (1993). Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. *The International Journal of Aviation Psychology*, *3*(3), 203–220. https://doi.org/10.1207/s15327108ijap0303_3

Kerns, T. (2019, May 7). There are now more than 2.5 billion active Android devices. *Android Police*. https://www.androidpolice.com/2019/05/07/there-are-now-more-than-2-5-billion-active-android-devices/

Kim, H. K., Park, J., Choi, Y., & Choe, M. (2018). Virtual reality sickness questionnaire (VRSQ): Motion sickness measurement index in a virtual reality environment. *Applied Ergonomics*, *69*, 66–73. https://doi.org/10.1016/j.apergo.2017.12.016

Kiyokawa, K. (2012). Trends and Vision of Head Mounted Display in Augmented Reality. *2012 International Symposium on Ubiquitous Virtual Reality*, 14–17. https://doi.org/10.1109/ISUVR.2012.11

Krueger, M. W., Gionfriddo, T., & Hinrichsen, K. (1985). VIDEOPLACE - an artificial reality. *ACM SIGCHI Bulletin*, *16*(4), 35–40. https://doi.org/10.1145/1165385.317463

Lin, C.-T., Chuang, S.-W., Chen, Y.-C., Ko, L.-W., Liang, S.-F., & Jung, T.-P. (2007). EEG Effects of Motion Sickness Induced in a Dynamic Virtual Reality

Environment. *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 3872–3875. https://doi.org/10.1109/IEMBS.2007.4353178

Lin, T., & Chao, L. (2021, April 27). *COMPUTEX 2021 Hybrid Embarks on an Intelligent Journey with AI Tech Giants*. https://www.businesswire.com/news/home/20210426005996/en/COMPUTEX-2021-Hybrid-Embarks-on-an-Intelligent-Journey-with-AI-Tech-Giants

Maass, U. (1999). *Device for displaying moving images in the background of a stage* (United States Patent No. US5865519A). https://patents.google.com/patent/US5865519/en

Mair, G. M. (1997). Telepresence-the technology and its economic and social implications. *1997 International Symposium on Technology and Society Technology and Society at a Time of Sweeping Change. Proceedings*, 118–124. https://doi.org/10.1109/ISTAS.1997.658870

Manifesto for Agile Software Development. (2001). https://agilemanifesto.org/

Material Design. (n.d.). Material Design. Retrieved March 17, 2022, from https://material.io/design/,

Miranda Sanchez. (2015, June 15). *E3 2015: Everything Announced at Microsoft's E3 Conference - IGN*. https://www.ign.com/articles/2015/06/15/everything-announced-at-microsofts-e3-conference

Munafo, J., Diedrick, M., & Stoffregen, T. A. (2017). The virtual reality head-mounted display Oculus Rift induces motion sickness and is sexist in its effects. *Experimental Brain Research*, *235*(3), 889–901. https://doi.org/10.1007/s00221-016-4846-7

Muñoz-Saavedra, L., Miró-Amarante, L., & Domínguez-Morales, M. (2020). Augmented and Virtual Reality Evolution and Future Tendency. *Applied Sciences*, *10*(1), 322. https://doi.org/10.3390/app10010322

Neagu, C. (2021, April 14). Simple Questions: What Is WiFi Direct? How Does It Work? *Digital Citizen*. https://www.digitalcitizen.life/simple-questions-what-wifi-direct-how-does-it-work/

Neges, M., Adwernat, S., & Abramovici, M. (2018). Augmented Virtuality for maintenance training simulation under various stress conditions. *Procedia Manufacturing*, *19*, 171–178. https://doi.org/10.1016/j.promfg.2018.01.024

Ordysiński, T. (2013). *Kanban based information management in organization*. *63*, 10.

Overvoorde, A. (2019, January). *OpenGL - Transformations*. https://open.gl/transformations

Özacar, K., Hincapié-Ramos, J. D., Takashima, K., & Kitamura, Y. (2016). 3D Selection Techniques for Mobile Augmented Reality Head-Mounted Displays. *Interacting with Computers*, *29*. https://doi.org/10.1093/iwc/iww035

Patrão, B., Pedro, S., & Menezes, P. (2020). *How to Deal with Motion Sickness in Virtual Reality*. The Eurographics Association. https://doi.org/10.2312/pt.20151201

Poppendieck, M. (2007). Lean Software Development. *29th International Conference on Software Engineering (ICSE'07 Companion)*, 165–166. https://doi.org/10.1109/ICSECOMPANION.2007.46

Rauschnabel, P. A., Rossmann, A., & tom Dieck, M. C. (2017). An adoption framework for mobile augmented reality games: The case of Pokémon Go. *Computers in Human Behavior*, *76*, 276–286. https://doi.org/10.1016/j.chb.2017.07.030

Schloerb, D. (1995). A Quantitative Measure of Telepresence. *Presence*, *4*, 64–80. https://doi.org/10.1162/pres.1995.4.1.64

Scrum Guide | Scrum Guides. (2020, November). https://scrumguides.org/scrum-guide.html

Serrano, B., Botella, C., Baños, R. M., & Alcañiz, M. (2013). Using virtual reality and mood-induction procedures to test products with consumers of ceramic tiles. *Computers in Human Behavior*, *29*(3), 648–653. https://doi.org/10.1016/j.chb.2012.10.024

Sheppard, D. (2017). Introduction to Progressive Web Apps. In D. Sheppard (Ed.), *Beginning Progressive Web App Development: Creating a Native App Experience on the Web* (pp. 3–10). Apress. https://doi.org/10.1007/978-1-4842-3090-9_1

Silicon Graphics Inc. (Ed.). (1991). *glLineWidth—OpenGL 4 Reference Pages*. https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glLineWidth.xhtml

Speicher, M., Hall, B. D., & Nebeling, M. (2019). What is Mixed Reality? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–15). Association for Computing Machinery. https://doi.org/10.1145/3290605.3300767

Suciu, G., Stefanescu, S., Beceanu, C., & Ceaparu, M. (2020). WebRTC role in real-time communication and video conferencing. *2020 Global Internet of Things Summit (GIoTS)*, 1–6. https://doi.org/10.1109/GIOTS49054.2020.9119656

Tsukayama, H. (2012, April 18). How the Tupac 'hologram' works. *Washington Post*. https://www.washingtonpost.com/business/technology/how-the-tupac-hologram-works/2012/04/18/gIQA1ZVyQT_story.html

Verma, S. (2015, March 13). *A new way to see and share your world with 360-degree video*. Blog.Youtube. https://blog.youtube/news-and-events/a-new-way-to-see-and-share-your-world/

Walter, H., Li, R., Munafo, J., Curry, C., Peterson, N., & Stoffregen, T. (2019). *APAL Coupling Study 2019* [Data set]. https://doi.org/10.13020/XAMG-CS69

WebRTC 1.0: Real-Time Communication Between Browsers. (2021, June). https://w3c.github.io/webrtc-pc/

Wi-Fi Alliance (Ed.). (2020). *Wi-Fi Direct Specification*. 200.

Young, N. (2020). Pokémon GO in the Field: Digital Gaming Apps and Ethnographic Methods. *Undefined*. /paper/Pok%C3%A9mon-GO-in-the-Field%3A-Digital-Gaming-Apps-and-Young/3f6d1c8c440b06f1e904375c2c50620de7a03c9a

Zuniga Gonzalez, D. A., Richards, D., & Bilgin, A. A. (2021). Making it Real: A Study of Augmented Virtuality on Presence and Enhanced Benefits of Study Stress Reduction Sessions. *International Journal of Human-Computer Studies*, *147*, 102579. https://doi.org/10.1016/j.ijhcs.2020.102579

Anexos

Anexo A Issues do primeiro repositório do GlarAssist

Issue ID	Title	State	Author	Assignee	Labels	Time Spent
1	Build Remote Assistance UI	Closed	Gonçalo Santos		priority::high,type::feat	79200
2	Draw AR Notations in a plane	Closed	Gonçalo Santos		priority::high,type::feat	86400
3	Build REST API Library	Closed	Tiago Oliveira		type::feat	36000
4	Implement Websocket Client	Closed	Marco Santos	Tiago Oliveira	type::feat	36000
5	Implement WebRTC streaming	Closed	Tiago Oliveira			0
6	Integrate mobile App with the server services	Closed	Gonçalo Santos	Tiago Oliveira	priority::high,type::feat	0
7	Use the same methods as the WebUI	Closed	Marco Santos	Tiago Oliveira	priority::high,type::improvement	0
8	Integrate with google depth API	Closed	Tiago Oliveira	Tiago Oliveira	priority::medium,type::feat, workflow::in review	0
9	Add data/file transfer support	Closed	Gonçalo Santos	Tiago Oliveira	priority::high,type::feat,workflow::in review	0
10	Arrow and Planes enable by default	Closed	Gonçalo Santos	Tiago Oliveira	priority::medium,type::feat	0
11	Picture in Picture support	Open	Tiago Oliveira		workflow::blocked	0
12	Add support to freeze image	Closed	Gonçalo Santos	Tiago Oliveira	priority::high,type::feat,workflow::in review	0
13	Fix frame with draws from the previous freeze	Closed	Gonçalo Santos	Tiago Oliveira	priority::high,type::bug	0
14	Notify assistant when the app is in background	Closed	Gonçalo Santos	Tiago Oliveira	priority::high,type::feat,workflow::in review	0
15	Remove development settings button	Closed	Marco Santos	Tiago Oliveira	type::bug	0

16	RuntimeException: Unable to destroy activity {com.glartek.remoteassistance/com.glartek.remoteassistance.MainActivit...	Closed	Marco Santos		workflow::in review	0
17	abort	Closed	Marco Santos		workflow::in review	0
18	New design journey	Closed	Tiago Oliveira	Tiago Oliveira	priority::high,type::feat,workflow::in review	0
19	MVP turn on flash light	Open	Gonçalo Santos	Tiago Oliveira	priority::low,type::feat,workflow::blocked	0
20	MVP chat	Open	Gonçalo Santos	Tiago Oliveira	priority::low,type::feat,workflow::in development	0
21	Bad video transmission when freeze mode enabled	Closed	Marco Santos		workflow::in review	0
22	Refactor	Closed	Tiago Oliveira	Tiago Oliveira	workflow::in review	0
23	Bug(delete): AR not deleting objects	Closed	Tiago Oliveira	Tiago Oliveira		0
24	new app icon	Closed	Tiago Oliveira	Tiago Oliveira	workflow::in development	0
25	add additional fields to user Agent	Closed	Tiago Oliveira	Tiago Oliveira	workflow::in review	0
26	bug(draw): no drawing selected	Closed	Tiago Oliveira	Tiago Oliveira	workflow::in review	0
27	Ui patch	Closed	Tiago Oliveira	Tiago Oliveira	workflow::in review	0
28	slow disconnection detection	Closed	Tiago Oliveira	Tiago Oliveira	workflow::in development	0
29	bug(connection): on reconecting wifi	Open	Tiago Oliveira	Tiago Oliveira	workflow::in review	0
30	fix(ui): colors	Closed	Tiago Oliveira	Tiago Oliveira		0
31	MVP(stream): use agora services	Open	Tiago Oliveira	Tiago Oliveira		0
32	MVP(stream): use live switch services	Closed	Tiago Oliveira	Tiago Oliveira		0

33	Support new annotation touch mode	Closed	Gonçalo Santos	Tiago Oliveira	priority::medium,type::feat	0
34	mvp - add hololens video relay to agora	Closed	Tiago Oliveira	Tiago Oliveira		0
35	RuntimeException: java.lang.reflect.InvocationTargetException	Closed	Marco Santos		workflow::in review	0
36	isolate media transmition responsability	Closed	Tiago Oliveira	Tiago Oliveira	workflow::in review	0
37	Audio settings are lost when remote reconnects	Closed	Marco Santos	Tiago Oliveira	workflow::in review	0
38	IllegalStateException: Fragment ServerAssistanceFragment{ea0e6c4} (337d8ba1-56ea-4244-ad2b-2560d3fc8cc2)} not attached t...	Closed	Marco Santos		workflow::in review	0
39	NullPointerException: null cannot be cast to non-null type android.graphics.drawable.RotateDrawable	Closed	Marco Santos		workflow::in review	0
40	mvp-ctrl-z draw	Open	Tiago Oliveira		type::feat,workflow::in development	0
41	Force numeric keyboard on session ID	Closed	Gonçalo Santos	Tiago Oliveira	priority::medium,type::improvement,workflow::approved	0
42	Session ID splited in 3 digits	Closed	Gonçalo Santos	Tiago Oliveira	priority::high,type::improvement,workflow::in review	0
43	Draw / Arrow not drawing from remote	Closed	Marco Santos	Tiago Oliveira	type::bug,workflow::in development	0
44	change arrow	Closed	Tiago Oliveira		workflow::in review	0
45	change line	Open	Tiago Oliveira	Tiago Oliveira	workflow::in review	0
46	self registration	Closed	Tiago Oliveira	Tiago Oliveira		0
47	Agora.io migration	Closed	Tiago Oliveira			0
48	Line revamp	Closed	Tiago Oliveira	Tiago Oliveira	workflow::in review	0
49	fix volume control	Closed	Tiago	Tiago	workflow::in review	0

			Oliveira	Oliveira		
50	agora.io data channels	Closed	Tiago Oliveira	Tiago Oliveira	workflow::in development	0

Anexo B Análise competitiva

Redacted name	Freeze mode	Recording	Zoom	File transfer	Translation	simple dev	HQ shot	AR	AI	Web App	MR
Factus	X	X	X	X	X	X	X	X			
Senhanced								X	X	X	
Situationer	X	X	X	X		X			X		X
Happeningset		X		X		X		X			
Enhancedring		X		X				X			
Realismia		X				X			X		
AugmentUs		X		X				X		X	
TruthUp	X			X							
Refact		X		X		X		X	X		
UniPratical	X					X		X			
Aria		X				X	X	X	X		
Refact				X		X	X	X			
ThuthMax								X			X
GoFact		X	X	X		X					
Realismium	X			X		X	X	X	X		
NuReality		X				X		X			
ARing		X	X		X	X		X			
RealismPro	X			X		X	X	X			
Total	6	11	4	11	2	13	5	14	6	2	2

Anexo C algoritmo de desenho do corpo da seta em 2D

- body width filled (BWF) = 0
- starting x = clicked x (CX) - body w offset
- starting y(SY) = clicked y (CY) + body v offset (BVO)
- ending y (EY) = CY +BVO - body height
- While the BWF is less than the body width
    - BFW = BFW + 1
    - Draw line starting (CX + BFW, SY) and stopping at (CX + BFW, EY)

Anexo D algoritmo de desenho do ponto da seta em 2D

- current start x (CSX) = CX- arrow offset (AO)
- current end x (CEX) = CX + AO
- current y (CUY) = CY - arrow height
- While CUY is less or equal to CY
    - CUY = CUY + 1
    - Draw line starting (CSX, CUY) and stopping at (CEX, CUY)
    - CSX = CSX + 1
    - CEX = CSX - 1