UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO SUPERIOR DE ENGENHARIA DE COMPUTAÇÃO

GUIDO MARGONAR MOREIRA

DESENVOLVIMENTO DE UM SERIOUS GAME PARA USO NA REABILITAÇÃO DO MOVIMENTO DE PRONAÇÃO E SUPINAÇÃO DO ANTEBRAÇO

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO

APUCARANA
2024

GUIDO MARGONAR MOREIRA

DESENVOLVIMENTO DE UM SERIOUS GAME PARA USO NA REABILITAÇÃO DO MOVIMENTO DE PRONAÇÃO E SUPINAÇÃO DO ANTEBRAÇO

Development of a Serious Game for use in the rehabilitation of pronation and supination movements in the forearm

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Engenharia de Computação, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná – Câmpus Apucarana, como requisito parcial para obtenção do título de Engenheiro de Computação .

Orientador: Prof. Dr. Fabio Irigon Pereira
Coorientador: Prof. Dr. Daniel Prado De Campos

APUCARANA
2024

GUIDO MARGONAR MOREIRA

DESENVOLVIMENTO DE UM SERIOUS GAME PARA USO NA REABILITAÇÃO DO MOVIMENTO DE PRONAÇÃO E SUPINAÇÃO DO ANTEBRAÇO

Trabalho de Conclusão do Curso de Gradução apresentado como requisito para obtenção do título do Bacharel em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 17/Junho/2024

Prof. Dr. Fabio Irigon Pereira
Doutor em Microeletrônica
Universidade Tecnológica Federal do Paraná

Prof. Dr. Daniel Prado De Campos
Doutor em Engenharia Biomédica
Universidade Tecnológica Federal do Paraná

ARAPONGAS
2024

RESUMO

Tendo em vista a necessidade de ferramentas para motivar pacientes na realização de exercícios de reabilitação repetitivos e monótonos, porém necessários na reabilitação de atividades funcionais, este trabalho de conclusão de curso tem por objetivo o desenvolvimento de um *Serious Game* voltado a facilitar a reabilitação dos movimentos de supinação e pronação do antebraço. Para tal foi realizada a escolha de um *framework* para desenvolvimento de jogos, a coleta dos biossinais relevantes para os dois movimentos e a comunicação constante entre o jogo e o sensor. Por fim foi feita a construção de um suporte para tornar o controle do jogo confortável para as necessidades físicas dos pacientes, com o fim de futuramente permitir a realização de testes na prática e coletar o *feedback* com os profissionais da área, visando garantir que a experiência do jogo seja funcional e imersiva para os jogadores.

Palavras-chave: Processamento de sinais. Tecnologia de reabilitação. Jogos para computador.

ABSTRACT

Considering the need for tools to motivate patients in carrying out repetitive and monotonous rehabilitation exercises, that are necessary in the rehabilitation of functional activities, this course Final Paper aims to develop a Serious Game designed to encourage the performance of supination and pronation movements of the forearm. For this it was necessary to choose a framework for game development, to obtain the relevant biosignals for both movements and to establish a constant communication between the game and the sensor. Finally, a support was built to make controlling the game comfortable for the physical needs of patients, with the aim of allowing tests to be carried out in practice in the future and collecting feedback from professionals in the field, to ensure the experience is functional and immersive for the players.

Keywords: Signal processing. Rehabilitation technology. Computer Games.

LISTA DE ABREVIATURAS E SIGLAS

SIGLAS

AVC	Acidente Vascular Cerebral
CSV	*comma-separated value*
JSON	*JavaScript Object Notation*

SUMÁRIO

1 INTRODUÇÃO

O acelerado avanço tecnológico das últimas décadas tem por consequências, tanto a evolução dos aparelhos, quanto o barateamento dos modelos antigos o que tem permitido a universalização do acesso. Especialmente dos celulares. Dos quais se estimam cerca de 1,2 por habitante no Brasil (MEIRELLES, 2023). Apocompanhando essa evolução e acesso quase universais a *hardwares*, os *softwares* puderam florescer de maneira exponencial atendendo as necessidades das mais diversas áreas: desde matemática, trabalhos científicos, saúde, comunicação e em especial no lazer. Seja por meio das redes sociais, serviços de *streaming* e jogos que foram incorporados ao dia a dia de milhões de brasileiros.

Os jogos eletrônicos, ou *vídeo games*, são experiências visuais interativas onde qualquer assunto pode ser abordado por meio da proposta de desafios que um ou mais jogadores precisam superar para obter progresso. Criando uma ponte entre os próprios jogos de tabuleiro e os efeitos especiais do cinema.

No passado grandes pensadores, como Platão, já consideravam a possibilidades de jogos lúdicos incentivarem os jovens a aprenderem comportamentos benéficos (PLATAO, 1999), justamente pela a natureza envolvente dos jogos (atualmente agravada nos ambientes eletrônicos). Com uma ideia semelhante em 1970, Clark C. Abt criou o conceito de *Serious Game*, ou jogo sério (ABT, 1987). O qual consiste justamente na criação de um jogo que contenha um propósito que vai além da mera diversão por si própria, visando levar o jogador a alcançar algum bem externo ao aprendizado ou prática de uma atividade.

Alguns anos mais tarde em 2002 Nick Pelling forjou o termo *Gamification* (PELLING, 2009), que trata justamente dessa inclusão de aspectos de jogos em atividades de aprendizado ou trabalho. Alguns exemplos de aplicações desses conceitos no passado foram: a distribuição de recompensas para clientes recorrentes em lojas e as recompensas entregues a escoteiros por aprender coisas novas (BULLOCK, 2023).

A principal vantagem de se utilizar esse artifício é justamente o fato de tornar mais cativante uma atividade que poderia ser maçante, monótona e repetitiva. Com isso em mente já é comum que profissionais na área da saúde usem ou criem jogos eletrônicos que exijam a realização de movimentos específicos pelo paciente. Ajudando tanto na recuperação quanto prevenção da perda de movimentos, sejam elas causadas por fraturas ou perdas por doenças neurológicas como: Acidente Vascular Cerebral (AVC), lesões na medula espinhal, Esclerose Múltipla, Atrofia Muscular Espinhal, entre outros.

Entre os diversos exercícios para recuperação, destacam-se dois movimentos em específico para recuperar a mobilidade do antebraço, denominados pronação e supinação[1]. Essênciais para atividades como segurar objetos, escrever e digitar. Mas normalmente não são o foco principal nos jogos que usam a movimentação do corpo do corpo ou das mãos em específico. Como

[1] Movimentos de rotação do antebraço estendido.

os feitos para controles Kinect™[2], Nintendo Wii™[3] e com sensores da *leap motion*™[4]. Algumas tentativas de se abordar esse movimento em específico para *Serious Games* já foram realizadas em outros trabalhos, como (MASMOUDI *et al.*, 2023),(P CARRATAL-TEJADA M, 2019) e (FERREIRA; MENEZES, 2020), porém os jogos apresentados nestes projetos não são tão acessíveis devido ao uso de sensores mais caros, incluindo alguns dentre os mencionados anteriormente. Tendo isso em mente, este trabalho objetiva criar um jogo e um suporte que foque exclusivamente no incentivo de supinação e pronação, além de fornecer fácil acesso, fácil instalação e baixo custo.

Para o desenvolvimento desse jogo foi necessário, realizar a coleta dos biossinais, dados observáveis produzidos por seres vivos, que permitam mensurar a rotação do antebraço do jogador. Os dados são então enviados ao jogo para controlar as ações do personagem. Esse envio deve ser rápido o bastante para evitar um *delay* que atrapalhe a experiência do usuário. Os desafios propostos deverão incentivar a prática de diferentes combinações dos movimentos, através de pontuações e recompensas, que motivem o jogador.

Outro aspecto importante é que o jogo tenha seus parâmetros ajustáveis para as necessidades de cada paciente, seja trabalhando com diferentes limites nos ângulos do antebraço, diferentes tempos de reação, entre outros. Assim, a aplicação será acessível para pacientes com maior ou menor grau de debilitação. Para tal foi importante a escolha de ferramentas adequadas, que automatizem etapas da criação dos jogos sem abrir mão dessa personalização na configuração.

Por fim, visando o uso desse jogo para acompanhamento do paciente por fisioterapeutas é importante armazenar relatórios contendo os desempenhos dos jogadores. Permitindo que se observe e analise se ocorre progresso, declínio ou estabilização das capacidades do paciente durante as sessões de jogo com configurações variadas. Isso além de permitir avaliar a situação do paciente, ajudará também na observação de se o jogo realmente possui um impacto positivo no tratamento de recuperação.

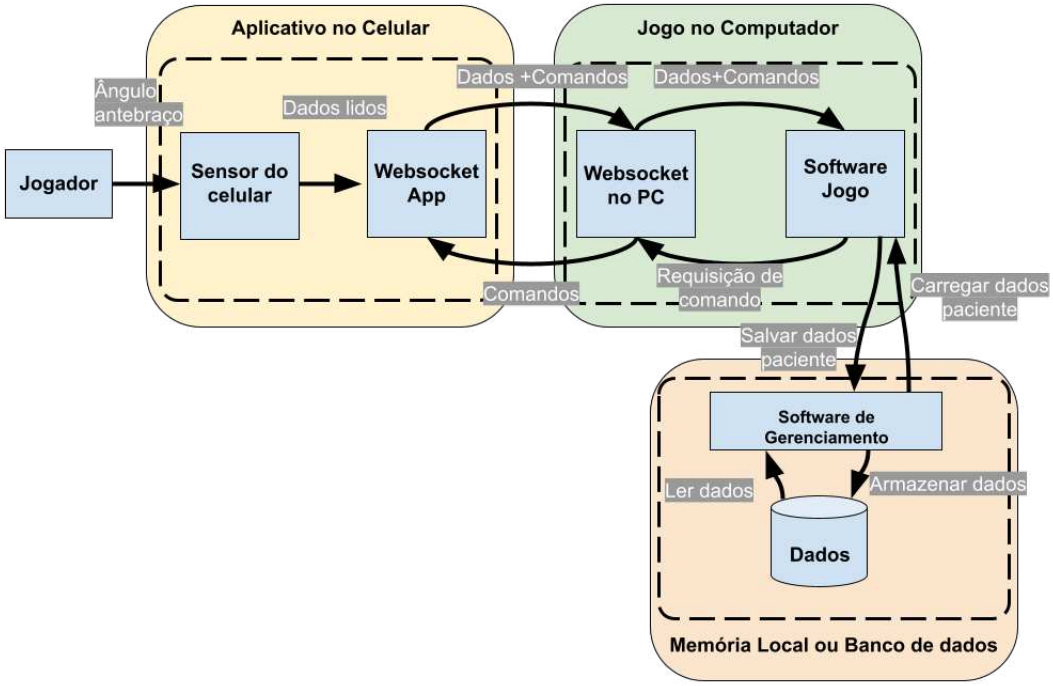
Em resumo, o jogo consistirá em um sensor, coletando o ângulo de rotação do antebraço do paciente, e enviando as medidas por meio de conexão sem fio para o jogo sendo executado em algum computador, que deverá responder, com pouco delay, aos movimentos realizados que permitirão a marcação dos pontos. E ao fim da sessão, irá armazenar o desempenho do jogador, permitindo análise dos fisioterapeutas como ilustrado em mais detalhes na Figura 1.

[2] Aparelho capaz de detectar pose do jogador para usar nos jogos introduzido no Xbox360 da empresa Xbox.

[3] Console que usa a rotação do controle como elemento em certos jogos criado pela empresa Nintendo.

[4] Sensores desenvolvidos pela empresa Ultraleap que escaneiam as mãos permitindo mais interação em experiências imersivas como VR (também conhecido como realidade virtual, se trata de experiências em algum suporte com uma tela fica preso a cabeça do jogador e os movimentos são traduzidos para dentro do jogo).

Figura 1 – Fluxograma Jogo Completo



Fonte: Autoria própria.

1.1 OBJETIVOS

Este projeto tem por objetivo a criação de um jogo para auxiliar em exercícios para recuperação dos movimentos de pronação e supinação do antebraço.

1.1.1 Objetivo Geral

Desenvolver um *Serious Game* para incentivar a realização de exercícios de reabilitação para os movimentos de pronação e supinação do antebraço, que por meio da coleta e armazenamento dos biossinais relevantes, possa gerar relatórios para auxiliar na avaliação da evolução dos pacientes por profissionais da área.

1.1.2 Objetivos Específicos

- Desenvolver aplicativo para realizar coleta dos biossinais do ângulo de rotação do antebraço, tanto esquerdo quanto direito por meio do giroscópio do celular.
- Desenvolver sistema de comunicação sem fio que envie os dados do sensor para o jogo.
- Criar um jogo que use os biossinais mencionados para marcar pontos baseados no desempenho do movimento, incentivando a realização de repetições, ângulos específicos e velocidades variadas.

- Criar painel de configurações, no jogo, para ajustar parâmetros que precisam ser adaptados às necessidades do paciente, tais como: velocidade dos obstáculos no jogo, *delay* entre os obstáculos, tamanhos de cada obstáculo (para pacientes que tem dificuldade em manter um ângulo preciso), ângulos de máximo e mínimo alcançados pelo antebraço.
- Armazenar dados, gerar relatório de desempenho e disponibilizá-los para análise futura, mostrando informações relevantes como tempo, ângulo do braço do paciente, resultado esperado pelo jogo, etc.
- Construção de suporte para fixação do braço do paciente, visto que pela dificuldade do movimento os pacientes tendem a compensar a força do movimento no tronco e ombro invés do antebraço.
- Criação de suporte móvel para acoplamento de celular com a mão do paciente, permitindo a realização do movimento.
- Criação de um protótipo acessível financeiramente.

2 TRABALHOS RELACIONADOS

Existem inúmeros trabalhos que já trataram do tema de *Serious Games* para supinação e pronação do antebraço, dentre os quais pode-se destacar o de Mostefa Masmoudi(MASMOUDI *et al.*, 2023). Os autores desenvolveram um sistema de reconhecimento facial para ajudar a identificar a situação emocional dos pacientes aos jogar *Serious Games*. Com esse sistema de avaliação pronto, eles desenvolveram um total de 5 jogos, cujo objetivo principal é ajudar as pessoas que sofreram de AVC. Cada um dos jogos é focado em um exercício de reabilitação diferente. Além disso, foi desenvolvida uma interface que, por meio de um banco de dados, salva as configurações de cada usuário. O que permite, por exemplo, mudar o modelo no jogo entre homem e mulher. O único jogo deste trabalho cujo foco é o movimento de pronação e supinação é o terceiro. Para coleta dos biossinais eles utilizaram o sensor *Leap Motion*, que consegue identificar a posição das duas mãos do jogador. No jogo, o paciente deve pegar um recipiente com uma bola colorida e colocá-lo em outro recipiente da mesma cor da bola, simulando um cenário comum como ao despejar o líquido de uma garrafa em um copo.

De maneira semelhante, o trabalho de Carretal-Tejada (P CARRATAL-TEJADA M, 2019) também utiliza o sensor de *Leap Motion*, mas com foco em jogos para recuperação de pacientes com Mal de Parkinson. Para esse fim eles utilizaram a *Game Engine* Unity3D. Nos jogos desenvolvidos alguns faziam uso de uma mão de cada vez, enquanto outros envolviam as duas simultaneamente. Vale destacarmos o jogos denominado FG que tem por objetivo o incentivo a movimentos de supinação e pronação. Nele o paciente segura uma bandeja virtual em cada mão, cada uma contendo um cubo, ele então precisa virar a mão com a palma pra baixo para o cubo cair no chão. Neste projeto em específico os jogos são altamente customizáveis de acordo com as capacidades de cada paciente, sendo possível configurar o número de cubos e realizar ajustes nas posições da mão. Os dados relativos a cada paciente são armazenados em arquivos CSV permitindo ao jogo carregar as preferências dos jogadores recorrentes. Assim, para avaliar a efetividade de cada *Serious Game*, esse projeto utilizou os métodos *The Box and Blocks Test* (BBT), *The Purdue Pegboard Test* (PPT) e um questionário de satisfação do cliente (CSQ-8).

Por fim, o projeto de Bruno Ferreira(FERREIRA; MENEZES, 2020). É focado em *Serious Games* para recuperação de pacientes que sofreram AVC. O trabalho revisa diferentes métodos para tratamento do paciente, bem como métodos de Gamificação voltados para reabilitaçã. É feita a apresentação de jogos voltados ao movimento de supinação e pronação. Para o jogo deste seu trabalho foi utilizado o controle VR do playstation. Os jogos desenvolvidos por ele não tinham foco no movimento de pronação e supinação, porém um diferencial em relação aos mencionados anteriormente foi a criação de uma ”Área do Paciente”, onde o desempenho dos pacientes durante o jogo está disponível para o profissional responsável, com todos os dados armazenados em um banco de dados.

Além dos já mencionados, muitos outros já trataram da criação de *Serious Games* para

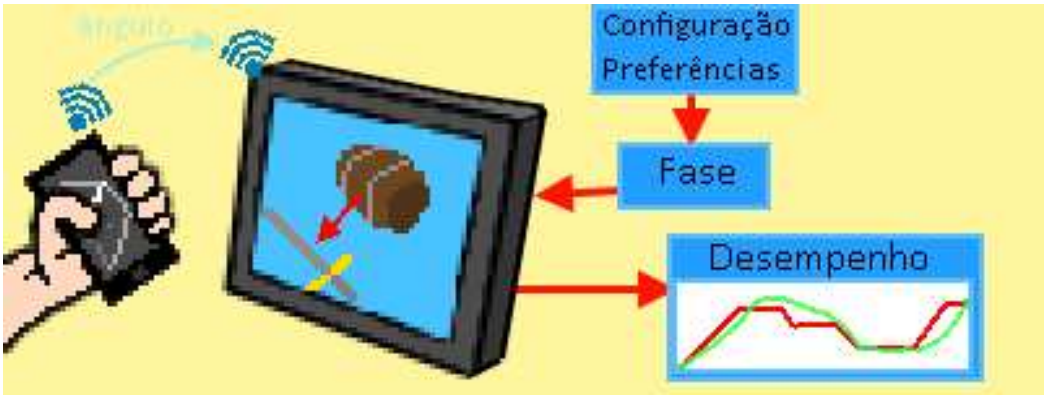
diversas áreas da saúde, esses três no entanto se destacaram em especial pelo tratamento do movimento de supinação. Seja propondo soluções diretas que é o caso do primeiro e do segundo, ou de forma mais teórica no caso do terceiro. Este projeto busca seguir a direção dos dois primeiros, porém focando na criação de uma experiência imersiva bem completa para esses movimento em específico, além da criação de um suporte que permita a fixação do braço para favorecer este movimento. Vale destacar que este projeto também buscou implementar a criação de relatórios com os desempenhos dos pacientes de maneira semelhante a “’Àrea do Paciente’.

3 MATERIAIS E MÉTODOS

3.1 PROPOSTA DE JOGO

Neste projeto foi desenvolvido um *Serious Game* onde um sensor irá detectar os movimentos de supinação e pronação do jogador, que serão enviados para o jogo e controlarão o ângulo de uma espada virtual. Ao mesmo tempo objetos saindo de uma carroça virão na direção da tela em diferentes posições, para que o paciente ajuste a posição da mão aos ângulos variados, alinhando a espada com o objeto que será cortado para marcar pontos, como ilustrado na Figura 2.

Figura 2 – Ilustração da proposta de *Serious Game*



Fonte: Autoria própria.

O jogo poderá ter objetos em ângulos aleatórios ou utilizar valores predefinidos, permitindo que sequências de movimentos do antebraço sejam exercitadas, como alternância ou fixação em ângulos específicos. Também serão fornecidos diferentes modos de jogo e parâmetros configuráveis para adaptar a dificuldade do jogo. Para o profissional acompanhando também será disponibilizado um relatório com o desempenho do paciente e um editor de fases para customizar a experiência.

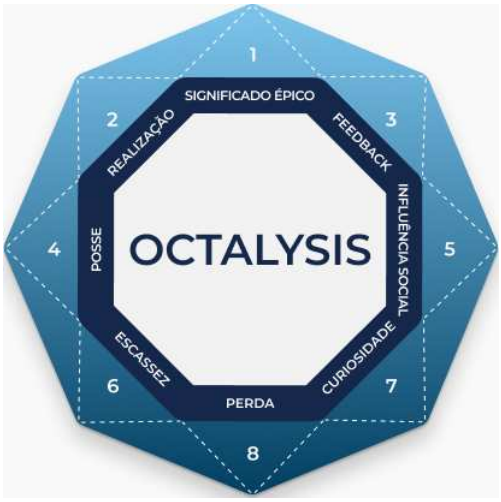
3.2 METODOLOGIAS PARA DESENVOLVIMENTO DE JOGOS

Devido a grande abrangência no uso de jogos, foram criados os mais variados métodos para seu desenvolvimento. Em se tratando de *Serious Games* podemos destacar a divisão entre Gamificação Estrutural e de Conteúdo. A primeira abordagem consiste na inserção de elementos de jogos à atividade que se deseja gamificar, como por exemplo, pelo método PBL, *Points*, *Badges* e *Leaderboards* (pontos, distintivos e placares). Que fornece recompensas ao jogadores por realizar as ações requisitadas, como os já mencionados escoteiros (BULLOCK, 2023). A segunda abordagem, a de conteúdo, insere o objeto da gamificação (no caso deste trabalho, o movimento de pronação e supinação) em um ambiente inovador e divertido, fora do contexto

real do objetivo. Assim, a atividade desejada é incentivada e aprendida sem a repetitividade e monotonia da situação real.

Outro modelo bem conhecido na área é o *Octalysis Framework*, proposto em 2010 por Yu-Kai Chou MARTINEZ (2021), considerado um dos Gurus da Gamificação. Neste método existem 8 classificações para determinar as motivações de uma pessoa para jogar um jogo: Significado Épico e Chamado; Desenvolvimento e Realização; Empoderamento da Criatividade e *Feedback*; Propriedade e Posse; Influência Social e Afinidade; Escassez e Impaciência; Imprevisibilidade e Curiosidade; Perda e Prevenção. O segundo, quarto e sexto são associados ao "lado esquerdo do cérebro" (conceito meramente didático visto que a diferenciação dos lados do cérebro foi desmistificada) focando na parte mais lógica e objetiva enquanto o terceiro, quinto e sétimo focam no "lado direito" envolvendo o lado mais emotivo e criativo. Por fim, a ilustração 3 mostra as 8 classes onde são denominados positivos os métodos da metade de cima e negativos os métodos da metade de baixo.

Figura 3 – *Framework Octalysis* para Gamificação



Fonte: Adaptada de MARTINEZ (2021).

Levando em conta que o público alvo deste projeto é um grupo de pacientes pertencentes a um nicho mais específico, a abordagem desenvolvida pelo trabalho (HEIDMANN, 2015) pode ser mais adequada. O primeiro passo desse método envolve uma clara comunicação entre todas as partes envolvidas (no caso deste projeto: desenvolvedores, fisioterapeuta e pacientes), seguida pela identificação do problema que se busca resolver, para isso são feitas as seguintes perguntas: sobre o que é o jogo? Para quem? Onde será usado? Quando será jogado? Como será jogado? E porque um jogo?

Após as perguntas serem respondidas, o método também chama atenção para algumas possíveis falhas ao longo do processo, passando em seguida para a fase conceitual onde o jogo é planejado até enfim chegar na fase de produção do programa. O design começa com um documento conceitual curto seguido de um documento de design mais detalhado, e por fim é criado um documento de produção considerando os custos e tempos de produção.

Nesse método também é ressaltada a importância de se adotar algum ciclo de design durante o desenvolvimento do jogo, de maneira que o *feedback* dos jogadores seja constantemente considerado para cada nova iteração do programa.

Assim, visto que a metodologia dessa abordagem é mais alinhada com os objetivos deste trabalho, foi decidido que ela seria utilizada de base para o desenvolvimento deste projeto. Sendo a princípio serão prioridades a descoberta das necessidades do jogador, e as mecânicas a serem usadas dentro do jogo.

Os *video games* podem ter gráficos desenvolvidos em 2D, 3D ou uma combinação dos dois. Porém o mais importante é a implementação das mecânicas e do controle, que funcione sem um cabo para evitar restrições desnecessárias ao paciente. Em resumo o que já está demonstrado pelo esquema da Figura 2.

Com tudo isso em mente também foi importante que o jogo fosse amplamente customizável. Todas as mecânicas devem operar se adaptando as configurações de: intervalos de ângulos máximos e mínimos, a velocidade dos obstáculos, o intervalo de tempo em que aparecem na tela, entre outros. Permitindo que o usuário/fisioterapeuta sejam capazes de modificar de acordo com suas necessidades e preferências. Para esse fim, foi implementado um sistema de calibração que utiliza o próprio sensor. Bem como um painel de configuração que permite a alteração de todos os parâmetros relevantes.

Além do mais, por se tratar de um jogo com propósitos de reabilitação foi preferível que a pontuação do jogador seja independente da dificuldade configurada, para não desestimular os pacientes ao compararem seus pontos com outros jogadores que já estiverem em melhor condição física. Em contrapartida, oferecer recompensas baseadas na melhoria do desempenho pessoal possa ser um bom incentivo para os pacientes continuarem jogando.

3.3 MOVIMENTO DE SUPINAÇÃO E PRONAÇÃO

Como mencionado anteriormente, o jogo foi desenvolvido com o intuito de incentivar os movimentos de supinação e pronação. Eles consistem na rotação do antebraço em sentidos horário ou anti-horário partindo do meio, tendo assim resultam no deslocamento da palma da mão para baixo e para cima respectivamente, como demonstrado na Figura 4. O movimento é realizado preferêncialmente com o cotovelo dobrado para impedir a compensação por outras partes do corpo, ele pode ser realizado em diferentes velocidades além de apresentar variações no padrão de movimento, como ao manter o antebraço fixo em uma posição ou ao alternar repetidas vezes entre supinação e pronação.

Figura 4 – Ilustração demonstrando movimento de pronação e supinação

Fonte: Autoria própria.

3.4 CONFIGURAÇÕES DE PREFERÊNCIAS PARA O JOGO

Visando permitir que o jogo possa ser adaptado às necessidades de variados pacientes, foi necessário que algumas das mecânicas fossem programadas para funcionar de acordo com os valores das variáveis relevantes. Também foi necessário a criação de um painel, cuja interface permite fácil modificação de cada uma das variáveis relevantes.

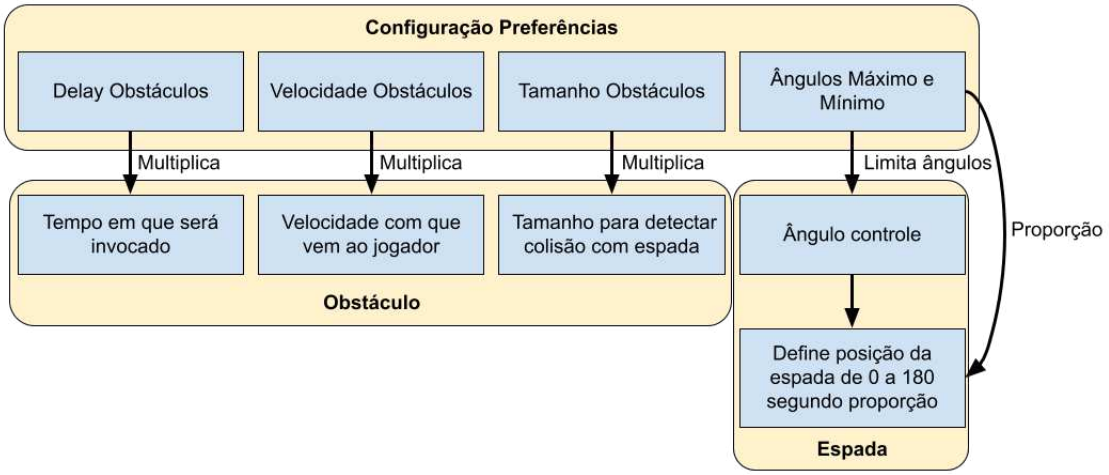
Considerando a natureza do jogo, alguns parâmetros essenciais incluem: os atrasos de tempo para invocar novos obstáculos, pois alguns pacientes podem ter dificuldades com muitos movimentos seguidos; a velocidade com que os obstáculos vão se mover, importante pelo mesmo motivo da anterior, mas também pessoas que possuam tempo de reação mais lento; a precisão dos movimentos (por meio de obstáculos de tamanhos variados), para pessoas que tenham dificuldade em manter o braço fixo nas posições específicas; e os ângulos de máximo e mínimo da espada, permitindo que cada paciente jogue de acordo com os limites de que seu antebraço é capaz. Assim, o menu funciona como ilustrado do Fluxograma 5.

3.5 COLETA DE BIOSSINAIS

Para permitir a interatividade com o *Serious Game* desenvolvido, é necessária a coleta dos biossinais (dados observáveis produzidos por seres vivos) relevantes para o movimento de interesse. Assim, para detecção da supinação e pronação, o biossinal mais relevante é o ângulo em que se encontra o antebraço do paciente.

Para obtenção desse valor existem inúmeros métodos disponíveis, como: com um suporte móvel acoplado a um potenciômetro cuja tensão seria lida por um microcontrolador com voltímetro, permitindo o cálculo do ângulo da mão pela leitura de tensão. Uma desvantagem desse método seria a necessidade de se construir as estruturas e circuitos segundo as especifi-

Figura 5 – Fluxograma para menu de configurar preferências



Fonte: Autoria própria.

cações feitas neste trabalho, o que poderia complicar a acessibilidade do projeto para leigos da área; uma outra possibilidade seria o processamento em tempo real, por algoritmos de inteligência artificial (IA) para detectar a posição da mão. Nesse caso as desvantagens envolvem tanto a modelagem do modelos quanto o treinamento de IA, que precisaria de uma base de imagens robusta para treinamento. Visando assim permitir que a IA consiga detectar todos os diferentes tipos de mão que os pacientes podem ter. Além de fazer isso independente dos cenários de fundo, que podem variar com a iluminação e local onde o jogo estiver sendo jogado.

Uma terceira alternativa seria através do uso de controles já consolidados, como os já mencionados *Leap Motion*, item (a) Figura 6, *Kinect*, item (b) Figura 6 e *Nintendo Wii*, item (c) Figura 6. Entretanto atualmente eles apresentam custo comparável a de um console mais simples (se é que não exigem um console para funcionar). Além disso alguns dos consoles não receberem mais suporte oficial por suas empresas para o desenvolvimento; Assim a alternativa ideal foi a criação de um aplicativo de celular, que realize a leitura dos sensores de giroscópio e acelerômetro. Dessa maneira, se acoplado a mão do paciente (ou simplesmente sendo segurado) permite leitura do ângulo do antebraço. Esse método terminaria por ser mais barato e acessível, visto que o acesso a celulares vem constantemente aumentando ao ponto de ser quase universal (MEIRELLES, 2023),

3.6 SUPORTE

Mesmo que o celular possa ser segurado diretamente, ainda foi interessante a criação de algum tipo de suporte. Que será responsável por manter o celular acoplado na mão do jogador ou fornecendo algum tipo de puxador para ajudar o paciente a firmar sua mão em uma posição mais confortável. Que é importante dado que alguns pacientes possuem dificuldade para manter a preensão da mão sobre objetos mais largos.

Algumas possibilidades incluem acoplar um suporte de celular em uma luva para o pa-

Figura 6 – Exemplos de Controles: (a) Cadastro Leap Motion, (b) Controle Kinect, (c) Controle Nintendo Wii

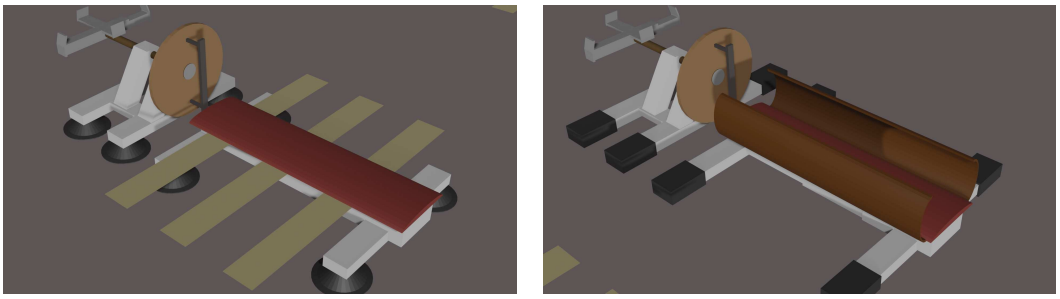
(a) Controle Leap Motion **(b) Controle Kinect** **(c) Controle Nintendo Wii**

Fonte: Adaptadas de Motion (2022), Evan-Amos (2010) e Evan-Amos (2022).

ciente não precisar segurá-lo, ou projetar e construir um suporte de mesa que permita a rotação do celular, de preferência com tamanho ajustável para braços maiores e menores. No caso desse segundo também seria interessante a criação de algum encosto para que o antebraço do paciente não fique no ar ou sobre a mesa, evitando assim que o jogador compense a força do antebraço com outras partes do corpo, como ombros ou tronco. Tendo esses parâmetros em mente foram modelado os rascunhos 3D: o primeiro no item (a) da Figura 7, com ventosas e com velcro para imobilizar o braço; e o segundo no item (b) das Figuras 7, com apoios de borracha e utilizando arcos nas laterais para impedir o braço de ir para fora sem restringir muito a mobilidade do antebraço.

Figura 7 – Rascunhos de Suportes: (a) Rascunho com ventosas e velcro, (b) Rascunho com borrachas e arcos

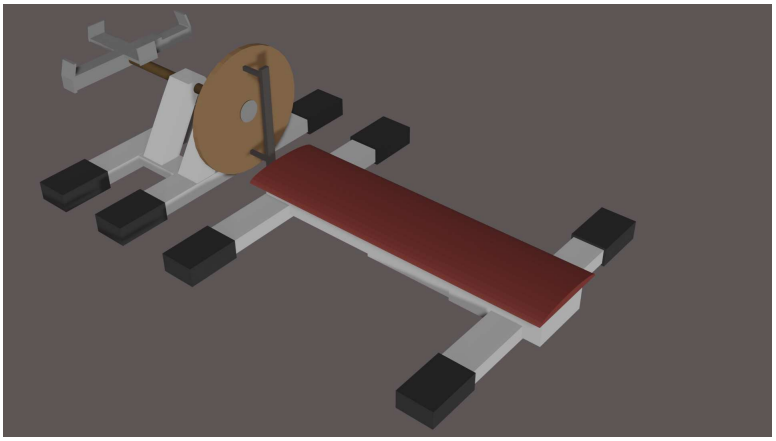
(a) Rascunho 3D para suportes de celular e braço, utilizando ventosas e velcro **(b) Rascunho 3D para suportes de celular e braço, utilizando borrachas e arcos**



Fonte: Autoria própria (2024).

Por sugestão da fisioterapeuta foi decdido que seria usado de base o modelo no item (b) da Figura 7, porém a princípio sem os arcos para restringir o antebraço. Ficando então como na Figura 8. As medidas do suporte foram feitas considerando um braço de cerca de 35cm de comprimento e 8cm de largura, mas com margens de erro para variações nos comprimentos e largura do braço ou da mão. Por fim ficaram definidas as medidas no Anexo A.

Figura 8 – Rascunho 3D para suportes de celular e braço, utilizando borrachas



Fonte: Autoria própria.

3.6.1 Materiais

Os materiais utilizados na construção do suporte foram: madeira para as estruturas principais; uma esponja costurada com tecido para o travesseiro; rolamentos e um parafuso longo de metal para mover o suporte do sensor; PLA (poliácido láctico) para a estrutura de suporte do celular, que foi impresso utilizando o modelo 9 disponível no site (MISTERTECH, 2016); e as estruturas dos suportes para a mão e braço do paciente foram feitas de madeira bem lixada e presa por parafusos.

Figura 9 – Projeto do suporte para celular pronto para impressão 3D

Fonte: Adaptada de mistertech (2016).

3.7 COMUNICAÇÃO SENSOR-JOGO

Outro aspecto importante a se considerar, é o sistema de comunicação entre o sensor com o jogo em tempo real. Existem várias opções para enviar esses dados lidos pelo aplicativo do celular, sejam elas uma conexão física por cabo ou a distância. O sistema cabeado teria a desvantagem de possíveis restrições nos movimentos do paciente, além de estragar mais fácil e poder se embaraçar com outros cabos.

Alternativamente, dentre os sistemas de comunicação a distância um bem conhecido é o *Bluetooth*, um sistema de comunicação de curto alcance (normalmente até 10 metros) comumente usado em fones e caixas de som[1]. Mas visto que nem todo os computadores possuem *Bluetooth* integrado ele poderia implicar em um custo adicional, necessitando da compra de um *Bluetooth* por USB (Universal Serial Bus, um padrão de comunicação por cabos comumente usado na criação de periféricos para o computador), além de possivelmente precisar de configurações dos drivers do computador, para que ele consiga utilizar o aparelho.

Com esses fatores em mente o meio de comunicação escolhido foi pela rede local através da internet. Por esse caminho existem inúmeros métodos, sendo que o escolhido para este projeto foi o dos *WebSockets*. Os quais consiste em um protocolo de comunicação com canais bidirecionais (que permitem tanto envio como recebimento de dados) em uma única conexão TCP, eles atualmente utilizam a API (Interface de programação aplicada) da W3C (a principal organização que desenvolve os padrões para serem utilizados na *web*) sendo compatível com HTTP (protocolo de transferência de hipertexto).

A lógica básica do sistema pode ser observada na Figura 10, onde o cliente realiza um request de *handshake* (ou aperto de mãos) para que o servidor estabeleça uma conexão. Com a conexão aberta, o *WebSocket* permite tanto que o controle (como cliente) envie dados para o jogo (como servidor), quanto ao jogo enviar dados para o controle. Bastando que um deles peça ou quebre o contato para a conexão se desfazer. Para permitir a fácil extração dos dados eles foram enviados como uma string seguindo os padrões de arquivos JSON, um formato para transparência de dados que permite fácil transferência de dicionários e matrizes em um formato de fácil leitura.

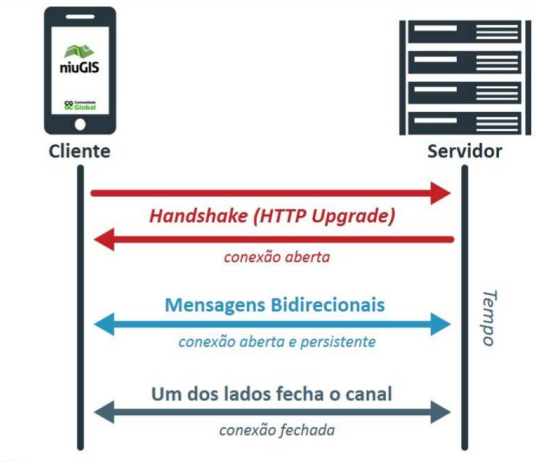
A escolha desse método se deve parcialmente, pelos experimentos realizados na iniciação científica (MOREIRA GUIDO MARGONAR; CAMPOS, 2022). Onde a velocidade de transmissão dos dados por um websocket em um microcontrolador ESP32, se mostrou eficiente contanto que a conexão com internet se mantivesse estável.

3.8 FRAMEWORKS E MOTORES DE JOGO

Com o objetivo de agilizar o processo da criação de jogos eletrônicos já foram desenvolvidas diversas ferramentas no contexto atual. Desde bibliotecas mais simples, com funções para

[1] https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

Figura 10 – Esquema mostrando funcionamento do *WebSocket*



Fonte: Adaptado de Solutions (2022).

facilitar o desenvolvimento de jogos diretamente pelos compiladores de linguagem de programação. Por meio da automatização de diferentes funcionalidades como: criação de interfaces visuais, interação com *inputs* do usuário, cálculo de físicas, etc. Por exemplo o PyGame[2], para o Python e o Libgdx[3], para Java, fornecem funções para abrir janelas, desenhar formas e atualizar a tela do jogo. Como mencionado, também existem bibliotecas focadas no tratamento de físicas como o *pybullet*[4], que integra o motor de física (simulador de física) *bullet* para ser utilizado com o Python.

Outro tipo de *frameworks* mais robustas, são as chamadas *game engines* (Motores de Jogos), que fornecem interfaces visuais completas para automatizar inúmeras etapas no desenvolvimento de jogos, como: o cálculo das físicas, importação de modelos 3D, a renderização de texturas, e muitos outros. Talvez a mais completa na atualidade seja a Unreal[5], que permite fácil criação de gráficos realistas com foco no 3D. Além de não cobrar taxas dos jogos que ainda não alcançaram uma faixa de lucro específica. Porém seu grande número de ferramentas poderosas pode causar problemas na performance tanto do jogo completo, quanto durante o desenvolvimento. Pois ocupa grande espaço na memória e exige uma capacidade de processamento um pouco acima da média para resultados mais satisfatórios.

Mais uma opção bem popular é o Unity[6]. Esse Motor de Jogo já demonstrou sua versatilidade no mercado, tendo sido utilizado para milhares de projetos tanto em 2D quanto 3D. Além de exigir menos capacidade de processamento para o desenvolvimento e possuir uma comunidade gigantesca que fornecem tutoriais e recursos de graça. No entanto, a empresa se envolveu em polêmicas no ano de 2023 ao modificar o sistema de taxação de forma arbitrária, o que gerou uma insegurança geral na comunidade. Alternativamente o Game Maker[7] é outra

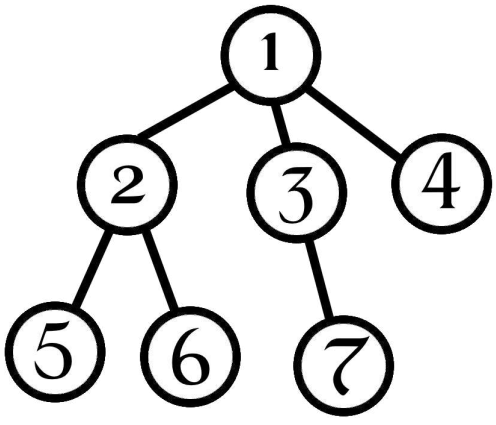
[2] https://www.pygame.org/docs/
[3] https://libgdx.com
[4] https://pybullet.org/wordpress/
[5] https://www.unrealengine.com/pt-BR
[6] https://unity.com/pt
[7] https://gamemaker.io/pt-BR

game engine consolidada no mercado, tendo como principal foco os jogos 2D, contando também com extenso suporte da comunidade, porém não permitindo exportação dos projetos na versão gratuita.

Existem incontáveis motores de jogos além desses, tendo em vista que o escopo deste jogo não precisava necessariamente de gráficos de última geração, e visando minimizar os custos computacionais, a *engine* de escolha para este projeto foi o Godot™[8]. Criado inicialmente por Juan Linietsky e Ariel Manzur, na Argentina, em parceria com a OKAM studios, a Godot é uma *game engine* que vem ganhando espaço no mercado. Um grande diferencial dela é por ser de código aberto, o que significa que seu código fonte está disponível para qualquer um interessado em utilizar ou modificar o programa. Inclusive permitindo a redistribuição do software alterado de forma gratuita ou paga até com outras licenças. Sendo o único requisito uma sinalização de que o projeto foi derivado da licença original[9].

Outro diferencial da Godot, é que todo o fluxo de trabalho na criação de jogos dentro dela, é realizado por meio de um sistema de *nodes* ou "nós" (embora a tradução mais apropriada seria vértice). Visto que cada cena da *engine* segue a estrutura semelhante a uma árvore da teoria dos grafo, no caso da Godot um nó principal é a "raiz" do grafo e todos os vértices "filhos" estão conectados a essa raiz, seja diretamente ou por meio de outros nós "pais", como ilutrado na Figura 11. Sendo que esses nós podem ser criados contendo diversas classes, que representam os diferentes componentes do jogo, como: um personagem, um cenário, a colisão de um objeto, botões, texturas, modelos 3D, entre outros. Dessa maneira os principais pilares do jogo, ficam contidos nessas estruturas. Cada uma contendo funções para seus respectivos papéis, que são herdados das classes de cada nó.

Figura 11 – Exemplo de grafo árvore.



Fonte: Autoria própria.

Ademais, cada nó pode ter um código atribuído a ele. Nesse arquivo o desenvolvedor tem liberdade para criar qualquer lógica de que o jogo necessite no componente selecionado. A

[8] https://godotengine.org
[9] https://godotengine.org/license/

linguagem de programação mais predominantemente usada na Godot é o Gdscript, uma linguagem de alto nível com estrutura bem simples, que herda características do Python, permitindo fácil leitura e utilizando a indentação do código para organização. Nessa linguagem estão presentes todas as funções referentes a cada um dos nós do motor de jogo.

Para permitir a comunicação entre diferentes códigos e nós a Godot disponibiliza os *signals*, ou sinais. Eles podem ser emitidos ao apertar de um botão, colisão de objetos, atualização de um valor ou diretamente por uma função no código. Assim, os vários elementos do jogo podem trabalhar em conjunto e reagir de acordo com os sinais emitidos pelas outras partes.

Quando o jogo estiver concluído ou pronto para testes, a *engine* permite que ele seja exportado como executável ou instalador para algumas das plataformas mais comuns, como: Windows, Linux, Android, entre outras. Por ser capaz de exportar apks, ela foi utilizada tanto na criação do jogo quanto no desenvolvimento do aplicativo para controle. E ainda, como a Godot possui suporte nativo a *WebSockets*, a implementação do sistema de comunicação entre controle e jogo é bem simples, como demonstrado pelo projeto (MRELIPTIK, 2021). Onde um aplicativo de celular controla o movimento e tiro de uma nave em um jogo no computador.

Por fim caso sejam necessárias ferramentas mais avançadas o Motor possui *plugins* desenvolvidos pela comunidade. Entre eles se destaca um chamado godot-python (TOUILLEMAN, 2022) compatível com as versões 3.x da *game engine*. Esse *plugin* permite o uso da linguagem de programação python, criada por Guido Van Rossum em 1991 objetivando fácil uso e aprendizado. Estando atualmente na sua terceira versão ela conta com bibliotecas desenvolvidas pela comunidade para áreas complexas como *data science* e *machine learning*. Os quais podem vir a ser úteis em melhorias futuras do projeto.

3.9 REGISTRO E ANÁLISE DO DESEMPENHO DO PACIENTE

Com o fim de permitir o acompanhamento da evolução do paciente é importante realizar o registro dos ângulos que o paciente conseguiu durante a sessão de jogo para serem comparados com os ângulos considerados ideais. Assim, foi desenvolvido um sistema para registrar um histórico, onde ficam salvos: o tempo em que os eventos aconteceram; os ângulos ideais para vencer os desafios; o ângulo do jogador naquele instante de tempo; os acertos ou erros de cada obstáculo; a pontuação atual; e o tamanho do obstáculo que seria equivalente a uma tolerância a erro.

Para fazer esses registros seria possível utilizar um banco de dados para salvar os dados online, mas como as relações entre os dados são muito pequenas (apenas com o instante de tempo em que aconteceram) o armazenamento pode ser feito através de métodos mais simples. Como por meio da criação de arquivos CSV. Que permitem o armazenamento de valores no formato de tabela, em um arquivo de texto por meio da separação de cada valor por vírgulas. Assim os dados ficaram salvos localmente nos arquivos do computador, facilitando o acesso e manipulação dos arquivos, embora a criação de *backups* para segurança precise ser feita manu-

almente.

Além do desempenho do usuário este projeto também salva as configurações com as preferências de cada paciente na respectiva data em que elas foram utilizadas. Como nesse caso são um volume menor de informações os dados foram registrados em arquivos do formato JSON. Neles os dados podem ser salvos em dicionários que facilitam a leitura do que cada informação significa. Dessa maneira, o jogo poderia facilmente carregar as preferências do usuário selecionado, ao mesmo tempo permitir que o profissional responsável avalie o progresso do paciente ao observar conforme os ajustes nas configurações foram mudando.

3.10 ASSETS

Assim que as mecânicas ficaram prontas, parcial ou totalmente, também foi necessária a criação de uma identidade visual para o jogo, através da inclusão dos chamados *Assets*. Que consistem em objetos 3D, texturas, artes personalizadas, efeitos sonoros, músicas, entre outros. Os quais ajudam a aumentar a imersão e o divertimento para o jogador. Também foi importante garantir que a interface seja intuitiva, profissional e bonita, para otimizar a experiência do usuário. Esses diferentes aspectos podem ser divididos em três áreas principais: *assets* 2D, *assets* 3D e Efeitos sonoros.

3.10.1 Criação Assets 2D

Os *assets* 2D envolvem a criação da interface para os menus, como: os botões e a escolha das fontes; além das artes para elementos 2D (chamados *Sprites*); e texturas para objetos 3D. Os programas que podem vir a ser utilizados nessa área a princípio seriam os de livre uso como: Paint.net[10], Gimp[11], InkScape[12] e Canva[13]. Em especial o Paint.net por possuir uma das interfaces mais intuitivas e ser o mais familiar ao desenvolvedor.

3.10.2 Criação Assets 3D

Para criação de jogos mais imersivos, contendo noções de distância e profundidade, costuma ser necessário o uso dos objetos 3D. Normalmente conjuntos de vértices, arestas e faces, combinados com texturas 2D para representar os elementos de jogos e animações. Embora a Godot permita o uso de volumes simples como cubos e cilindros é importante que sejam colocados modelos 3D customizados para representar o cenário, personagens e inimigos, para incentivar o engajamento com o jogo.

[10] https://www.getpaint.net/download.html
[11] https://www.gimp.org
[12] https://inkscape.org/pt-br/
[13] https://www.canva.com

Outro fator importante a considerar é garantir que os modelos não possuam polígonos em excesso, para não pesar no processamento do computador, fazendo com que o jogo trave. Portanto é preferível que os *assets* sejam do tipo *Low Poly*, que focam em criar modelos estilizados mantendo um pequeno número de polígonos. Ou também por meio de texturas 2D aplicadas a planos 3D para simular objetos detalhados. Para esse fim dois programas grátis se destacam: o *Blender*[14], programa *open source* com ferramentas para diversas áreas da arte digital mas com foco maior nas áreas que envolvem manipulação de objetos 3D, como modelagem, animação e renderização 3D; e o magicaVoxel[15], programa focado na criação e renderização de modelos 3D feitos de voxels (cubos) permitindo fácil criação de modelos estilizados.

3.10.3 Criação Efeitos Sonoros

Por fim para a criação e edição de efeitos sonoros se destacam softwares de edição de áudio como o Audacity[16], que permite aplicar efeitos sobre gravações feitas com microfone ou editando arquivos de áudios. As músicas poderiam ser feitas com auxílio de instrumentos físicos, digitais ou pela edição de *assets* gratuitos, que serão tratados a seguir.

3.10.4 Assets Grátis

Também é possível, por meio da internet se encontrar *assets* gratuitos. Alguns disponíveis no domínio público (ou seja livres para qualquer uso) e outros com licenças que limitam o uso ou exigem créditos como a *CC BY 4.0 DEED*[17]. Esses *assets* abrangem todas as áreas mencionadas acima. Desde: a interface do usuário, texturas, modelos 3D, *sprites* 2D, efeitos sonoros e até músicas. Assim, contanto que os termos das licenças sejam respeitados, essas ferramentas podem facilitar muito a criação da identidade visual do jogo. Para este projeto os *assets* externos utilizados se encontram no quadro 1, com seus respectivos tipos, elemento relevante para o projeto e link de onde foi adquirido.

3.10.5 Mapa Infinito

Para criar uma sensação de movimento foi utilizada uma técnica de geração procedural onde os pedaços de chão, contendo o cenário, são adicionados à cena por código (entre 3 opções de grama com objetos encima: árvores, arbustos, cogumelos, etc, ou uma ponte passando por um rio), e possuem um sensor de posição preso na extremidade do modelo. Ao chegar em uma certa posição é invocado outro chão nessa posição adjacente ao anterior.

[14] https://www.blender.org
[15] https://ephtracy.github.io
[16] https://www.audacityteam.org/download/
[17] https://creativecommons.org/licenses/by/4.0/deed.pt-br

Tipo	*Asset*	Link
3D	Espada 3D	(ADLER2019, 2020)
3D	Objetos para obstáculos	(DANIEL74, 2014)
3D	Carroça	(DANIEL74, 2012)
2D	*Sprites* medievais para jogo e interface	(FROG, 2022)
2D	Texturas de grama e terra	(POLIIGON, 2016)
2D	Espada 2D para ícone	(SHADE, 2022)
2D	Texturas medievais usadas na interface	(CAINOS, 2021)
2D	Estrelas para mostrar desempenho	(PRINBLES, 2023)
Efeitos Sonoro	Sons de espada, rio e floresta	(TOMMUSIC, 2023)
Efeitos Sonoro	sons para interface	(LEOHPAZ, 2022)
Música	Música de ação	(SHONONOKI, 2020)
Fonte	Fonte normal	(LUXWOLDA, 2024)
Fonte	Fonte medieval	(MILES, 2007)

Tabela 1 – Quadro com *assets* utilizados no desenvolvimento do jogo

Cada peça do cenário ficará se movendo em direção ao jogador e cada um possuí um *timer* próprio para ser removido pouco após sair da vista do jogador. Assim é criada a ilusão de se estar andando por um caminho sem fim, a vantagem desse método está em evitar que o jogo possa chegar nos limites computacionais das variáveis numéricas de posição (por exemplo: caso o jogador se movesse mais longe do que o computador pode calcular por padrão). Além de por criar combinações aleatórias fornece experiências de ambiente única para cada sessão de jogo.

3.11 FASES

Outro aspecto importante foi o desenvolvimento das fases, um conjunto de desafios a ser executado para vencer o jogo ou marcar pontos. Seria possível que o jogo funcionasse simplesmente gerando posições aleatórias para cada obstáculos, porém com o fim de incentivar a realização de padrões de movimento específicos foi mais interessante que ele fosse capaz de rodar algumas fases com valores já predefinidos, carregando os atributos para cada obstáculo que será instanciado.

Para isso podem ser utilizados novamente os arquivos JSON. Neles é possível criar um vetor de dicionários onde cada dicionário representará um obstáculo do jogo com os atributos relevantes: ângulo, a posição em que o obstáculo aparecerá na tela para o paciente atingir; tempo, o segundo em que o obstáculo será invocado; velocidade, um valor que multiplicará a velocidade permitindo criar obstáculos mais rápidos e mais lentos; tamanho, um modificador que multiplica o tamanho do obstáculo permitindo criar obstáculos maiores e menores, exigindo menor e maior precisão; índice do asset, que permitirá a escolha de qual o objeto 3D que vai aparecer para ser atingindo (exemplo: um saco ou um barril); ponto, um modificador que vai multiplicar o ponto que será ganho ao atingir o obstáculo (por padrão será modificador * 10 pontos), tudo isso seria salvo no formato JSON como exemplificado na Figura 12.

Então o jogo carrega todos os arquivos que se encontrem na pasta "Fases", como mostrado no item a) da Figura 13. Então irá exibi-los na tela para uma ser selecionada b), permitindo

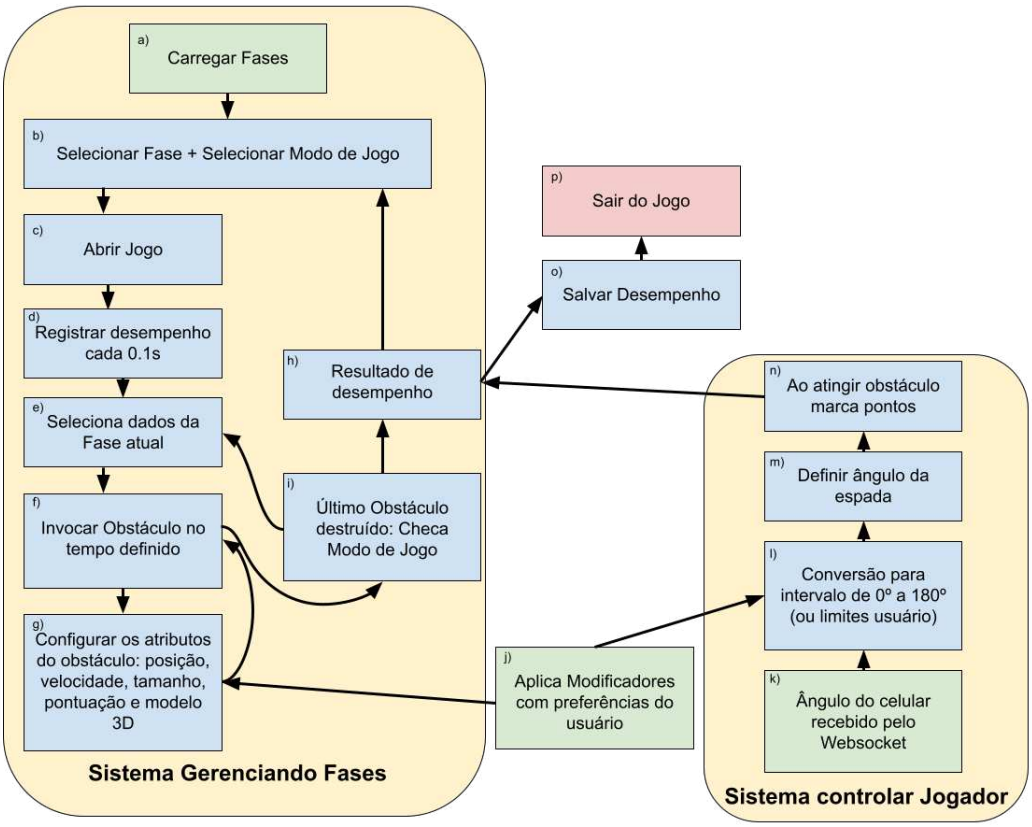
Figura 12 – Exemplo de dados de uma fase em JSON

```
{"desafio":[
  {"angulo": 0,
  "tempo": 1,
  "velocidade":1,
  "tamanho": 1,
  "asset": 0,
  "ponto": 3
  },
  {"angulo": 180,
  "tempo": 2,
  "velocidade":1,
  "tamanho": 1,
  "asset": 0,
  "ponto": 1
  }
]}
```

Fonte: Autoria própria.

ao jogador escolher a fase que preferir com o exercício que vai focar no dia. Além das fases, antes de iniciar o jogo também no item b) é disponibilizada uma opção para ser escolhido o modo de jogo. Alguns exemplos poderiam ser: simplesmente completar uma fase e ver a pontuação, jogar todas as fases em ordem aleatória sem fim, jogar sem fim até errar, ou talvez com posições aleatórias que vão ficando mais difíceis conforme o jogador marca pontos (com velocidades maiores, tamanhos menores e menos delay entre cada obstáculo).

Figura 13 – Fluxograma com Lógica do jogo



Fonte: Autoria própria.

Então o jogo será exibido c), o JSON de cada fase será convertido em um *array* de dicionários, e a fase (ou as fases) a serem utilizadas serão salvas em outro *array* contendo todos essas fases. No passo d), o *timer* responsável por registrar o desempenho do jogador então será iniciado a cada 0,1 s (permitindo a reconstrução da partida jogada com uma taxa de 10 quadros por segundo).

3.12 LÓGICA DO JOGO

Para rodar as fases dentro do jogo uma função é utilizada para iniciar o desafio com a fase escolhida no momento e). Serão coletados o tempo de invocação para cada obstáculo, sendo um índice global definido para o primeiro obstáculo e o desafio atual sendo salvo em uma variável a parte. Em seguida é inicializado um *timer* com o tempo para invocar o primeiro obstáculo.

No final do *timer* será executada uma função responsável por invocar o obstáculo do índice atual f), colocando ele na posição equivalente ao ângulo escolhido e configurando obstáculos segundo os atributos g): tamanho, velocidade, índice do *asset* e modificador de pontos.

Em seguida a função checa se existem mais obstáculos dentro da fase atual, caso existam ela vai realizar o mesmo processo de antes. Mas desta vez o *timer* tem o tempo relativo ao tempo atual voltando a executar f). Caso não hajam outros obstáculos para invocar na fase ele vai checar qual o modo de jogo sendo executado a tomar a ação apropriada i). Que pode ser: o início de outra fase e) ou a exibição da tela com o desempenho do paciente h). Dela é possível continuar jogando voltando para b) ou salvar o desempenho o) e fechar o jogo p).

A fase sendo rodada também utilizará os modificadores contendo as preferências para o jogo configuradas pelo jogador ou profissional acompanhando j). Assim algumas variáveis irão multiplicar atributos que afetam todos os obstáculos do jogo como: o tempo para invocação dos próximos obstáculos, o tamanho deles e a velocidade.

Também é permitido customizar os ângulos de máximo e mínimo, transferindo proporcionalmente o movimento do controle. Permitindo que um jogador que tenha movimento reduzido consiga mover a espada pela tela inteira. Nada disso afeta na pontuação, visto que serve apenas para adaptar a dificuldade do jogo às necessidades do jogador.

3.12.1 Controle Da Espada

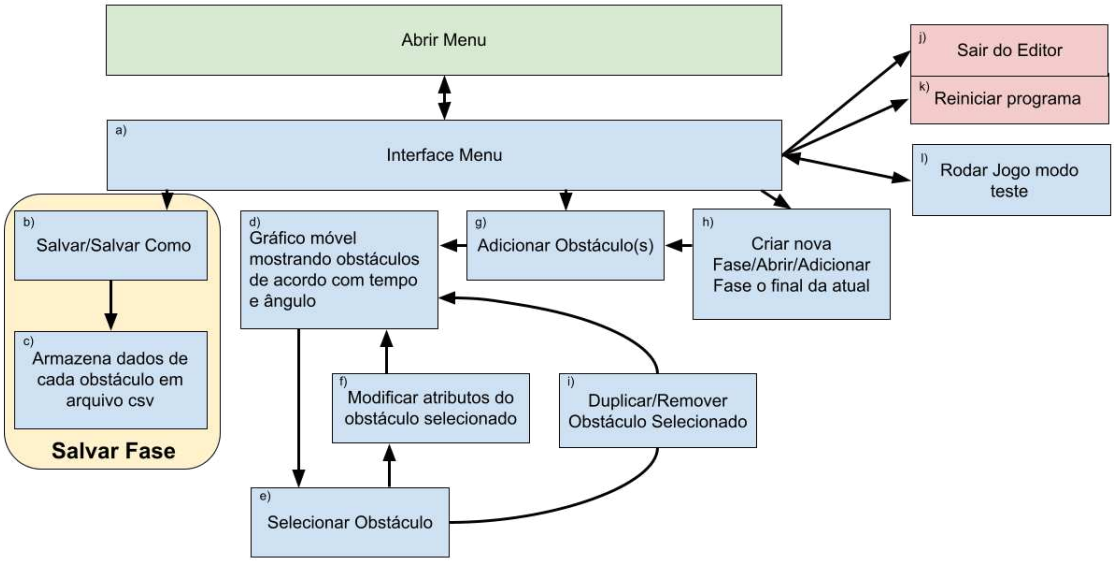
Simultaneamente, o celular estará enviando os dados com o ângulo do antebraço do jogador k), a medida recebida varia por volta de -10 a 10 equivalente a ângulos indo de -90 a 90 graus, porém esse valor nem sempre é preciso dependendo do estado do sensor. Além de que movimentos bruscos podem levar o sensor a ultrapassá-lo, então antes de ser convertido no ângulo da espada o dado precisa ser colocado normalizado dentro dos limites do hardware.

O valor então é convertido a princípio para graus de 0 a 180, ou para os valores configurados nas preferências do usuário j). Então a posição da espada será definida de acordo com esse valor m). A espada terá um sistema de detectar colisão, que incrementa a pontuação quando um obstáculo é atingido pela espada do jogador n). De acordo com o multiplicador de pontos associado ao obstáculo. Que havia sido definido por g). Esses dados então serão registrados no histórico do desempenho h), como um acerto ou erro.

3.13 EDITOR DE FASES

Aproveitando que as fases foram salvas no formato JSON, que permite fácil manipulação dos dados salvos, também foi criada uma aplicação conjunta ao jogo para criação e edição de fases. Nela serão implementadas todas as funções no fluxograma 14. Assim, após o editor de fases ser aberto são carregados todos os botões e opções da interface a). Da parte dos arquivos são: criação de uma nova fase do zero item h); abrir um arquivo de fase já existente h); salvar as mudanças no arquivo aberto ou salvar fase atual como novo arquivo ambos no item b) seguido pelo c); e também permitir que se adicione os osbstáculos de uma fase ao final de outra h).

Figura 14 – Fluxograma com lógica do editor de fases



Fonte: Autoria própria.

Com o arquivo aberto será criada uma interface visual interativa d) onde serão exibidos cada obstáculo contido na fase, como em um plano cartesiano, com o x sendo o ângulo de 0 a 180 e o y sendo o tempo começando no zero e indo até onde a pessoa desejar (talvez tenha um limites superior dependente do computador utilizado). É possível mover esse menu para cima ou para baixo com o *Scroll* do mouse para ver todos os obstáculos.

Na direita inferior do menu foi colocado um botão para permitir a adição de novos obstáculos a fase com valores padrão atribuidos. Para isso será criado um novo dicionário com valores

padrões e tempo igual a zero e ele será adicionado ao *array* da fase. Um sistema semelhante é utilizado nas operações de h).

Para permitir a seleção dos obstáculos, cada representação visual deles é um botão, que guarda na memória o índice de seu obstáculo no vetor de desafios. Assim quando o botão for apertado ele emite um sinal para que o editor carregue os dados do seu índice e). O editor então exibe cada um dos atributos em áreas de texto com legendas (Mostrando todos eles: velocidade, tamanho, asset, tempo, ponto e ângulo).

Os valores são exibidos em nós chamados *LineEdit* (ou linha editável), eles permitem ao usuário editar o valor exibido, além de emitir um sinal quando esse valor é modificado permitindo que ele seja atualizado dentro do respectivo obstáculo no vetor da fase f).

No caso da variável de ângulo também foi configurado um *Slider* indo de 0 a 180 para permitir a alteração do ângulo de forma mais intuitiva. Por fim, para manter a representação gráfica atualizada, sempre que as variáveis tempo ou ângulo são modificadas a posição do botão selecionado é modificada para a posição relativa aos novos valores em d).

Outro botão, implementado no canto inferior esquerdo do editor, foi o de deleção i). Ele basicamente checa se existe algum obstáculo selecionado e deleta o obstáculo na interface gráfica e o dicionário correspondente no *array*. Ele também precisa atualizar nos botões os índices que forem maiores que o do deletado já que o vetor diminuiu.

O terceiro botão, implementado logo abaixo do botão de adicionar obstáculos, foi o de duplicar i), ele primeiro checa se tem algum obstáculo selecionado e então funciona de maneira bem semelhante ao botão de adicionar, porém após criado o novo obstáculo cada um dos atributos é copiado para o novo dicionário e uma nova nova representação visual é colocada na mesma posição do obstáculo utilizado de base d).

Também foram implementados botões para fechar o programa: um que sai do editor e fecha a aplicação j) e outro para reiniciar a aplicação voltando para a tela de conexão do controle k).

Por fim, foi implementado o botão que permite testar a fase do editor dentro do jogo, quando esse botão é apertado ele emite um sinal para iniciar o jogo l). Junto com esse sinal são enviados um vetor com todos os obstáculos da fase e uma variável para configurar o modo de jogo de teste. Assim, o jogo só roda a fase do editor e quando o ultimo obstáculo dela for destruído o jogo será pausado e o editor de fase será aberto preservando os mesmos dados de antes a).

3.14 NAVEGAÇÃO PELAS TELAS

Para a navegação dentro da interface do jogo foram criados uma série de janelas conectadas entre si. No geral a navegação funciona como está exibido na Figura 15. Assim inicialmente é aberta a janela "Conectar controle", onde é exibido o Ip do servidor e o websocket do servidor fica aguardando a conexão. Quando o controle conecta a tela "Jogo ou Editor" é aberta com

dois botões. O botão da direita abre o "Editor de fases", que além da criação e edição das fases, permite voltar, testar o jogo ou fechar a aplicação.

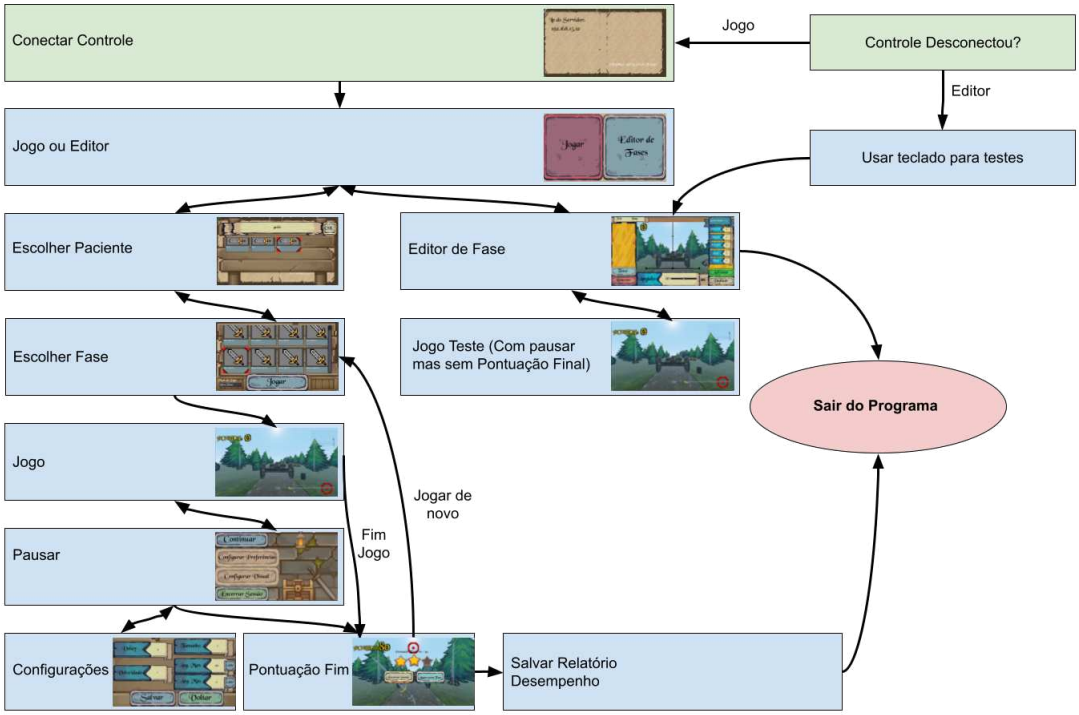
Já o botão da esquerda abre a tela para "Escolher paciente". Ele permite voltar ou seguir em frente quando o nome do paciente for confirmado. Daí ele passa para a tela "Escolher fase" que exibe todas as fases disponíveis na pasta e permite a seleção do modo de jogo (também tem um botão para voltar).

Depois da fase ser selecionada e o botão para jogar ser pressionado a tela "Jogo" é exibida. Então toda a lógica do jogo entra em ação e fica rodando até o fim da fase. Podendo ser interrompido pela tela "Pausar". Ela permite acessar a tela "Configurações" com as preferências do jogador. Além de permitir continuar o jogo ou encerrá-lo.

No caso de o jogo terminar de rodar as fases, ou o botão na tela de pause para encerrar a sessão for pressionado, será exibida a tela "Pontuação Final". Ela permite voltar ao menu de escolher fases para continuar jogando, mas também fornece a opção de salvar os dados com o desempenho e preferências do usuário em CSV e JSON, respectivamente. Para então fechar o jogo.

Além disso, caso a conexão com o controle seja interrompida o jogo irá se reiniciar, ou caso esteja sendo utilizado o editor de fases será ativado o modo controle por teclado. Que permite jogar com as setas direcionais do teclado, para poder continuar a testar a fase sendo modificada.

Figura 15 – Fluxograma com navegação entre telas



Fonte: Autoria própria.

3.15 TESTE DE SATISFAÇÃO

Depois de ter as primeiras versões do jogo pronta, com a aprovação do comitê de ética para o jogo e o suporte do sensor, será importante a coleta do *feedback* dos pacientes e da fisioterapeuta. Para avaliar os pontos positivos e negativos que eles tenham observado. Além de levar em conta as sugestões dessas pessoas que têm um contato mais direto com as necessidades dos pacientes e consequentemente aos ajustes necessários para o projeto. A princípio essa avaliação seria feita por meio de algum questionário, seguindo algum modelo de perguntas como o CSQ-8 utilizado no trabalho (P CARRATAL-TEJADA M, 2019).

Independente do método escolhido, será importante a coleta das opiniões quanto aos fatores tanto objetivos (desafios na vida real, nível de dificuldade das fases e responsividade do controle), quanto subjetivos (como temática, aparência, divertimento, etc). O questionário poderá ser respondido diretamente pelos fisioterapeuta e pacientes, após a sessão do jogo ou com o auxílio de terceiros caso necessário.

4 RESULTADOS E DISCUSSÃO

Com a junção de todas as ferramentas escolhidas, o resultado final se encontra na Figura 16. Onde o celular, preso ao suporte, roda o aplicativo que lê o acelerômetro e envia o ângulo para o *WebSocket* cliente. Que por sua vez envia os dados para o *WebSocket* servidor no computador. O jogo então tem acesso direto a esses dados que são usados para controlar o personagem do jogador (espada), que tenta destruir os obstáculos nas posições carregadas pela fase do arquivo.

Figura 16 – Jogo sendo jogado com o suporte, com o cotovelo mantendo uma flexão de 90 graus

Fonte: Autoria própria.

Em paralelo também é realizado o registro dos dados do paciente e da fase que está sendo executada. O que permitirá a geração de relatórios para análise do desempenho do paciente pelo fisioterapeuta. Todos os códigos que foram criados para o funcionamento do jogo estam disponíveis em um repositório do github (MOREIRA, 2023).

4.1 APLICATIVO

Para o desenvolvimento do aplicativo foi utilizado de base, o aplicativo disponível no repositório (MRELIPTIK, 2021). Assim basta digitar o Ip do servidor na caixa de texto, e apertar no botão "Conectar". Para que o *websocket* cliente tente estabelecer a conexão com o servidor, como mostrado na Figura 17.

Se a conexão for estabelecida com sucesso o aplicativo começa a ler continuamente o dado do acelerômetro no celular, esse dado então é passado para o formato JSON dentro de uma *string* e o *websocket* cliente envia os dados para o servidor toda vez que são lidos novos dados. Enquanto isso a tela do aplicativo exibe o ângulo na interface visual mostrada na Figura

Figura 17 – Janela do aplicativo para conectar ao servidor

Fonte: Autoria própria.

18. Onde a bolinha se move proporcional ao ângulo x ou z, que variam em média de -10 a 10, sendo equivalentes a ângulos de -90 a 90 graus. Porém, como explicado na metodologia, esse valor pode variar de acordo com a situação do sensor. Acima da bolinha são exibidos todos os ângulos lidos pelo sensor: x, y e z.

Figura 18 – Janela do aplicativo para ler e enviar dados ao servidor

Fonte: Autoria própria.

4.2 SUPORTE

Seguindo o modelo no Anexo A, as peças de madeira foram cortadas e perfuradas. Depois os parafusos e porcas foram encaixados, o resultado está na Figura 19. O suporte para celular foi feito por impressão 3D com o modelo vertical (MISTERTECH, 2016), sendo o resultado mostrado na Figura 20. Por fim o encosto o braço foi feito pela costura do tecido com uma esponja na Figura 21.

A junção de todas essas estruturas resultou nos suportes completos, como se podem observar na Figura 22. O suporte para o celular está bem leve na mão permitindo a realização dos movimentos de supinação e pronação de maneira confortável.

Figura 19 – Foto dos suportes madeira

Fonte: Autoria própria.

Figura 20 – Foto do suporte para celular

Fonte: Autoria própria.

Figura 21 – Foto encosto para o braço

Fonte: Autoria própria.

4.3 JOGO

O jogo fez uso na maior parte dos já mencionados *Assets* de licença CC0 (os *assets* que possuírem outras licenças estão com elas anexadas nas referências). Desde: espada, árvores, pedras, ponte, obstáculos para o jogador atingir, a carroça, além de outros *assets* utilizados

Figura 22 – Foto suporte completo

Fonte: Autoria própria.

no cenário do jogo. As imagens utilizadas para criar as diferentes interfaces do jogo vieram principalmente de dois conjuntos de texturas (FROG, 2022) e (CAINOS, 2021). Que foram modificados e organizados no paint.net, junto da fonte em estilo medieval (MILES, 2007).

Ao abrir o jogo a primeira interface que aparece é a tela para conexão do controle, onde aparecem os Ips do servidor, Figura 23. Nela é configurado e iniciado o *websocket* servidor que fica aguardando a conexão do controle.

Figura 23 – Tela para conectar Sensor

Fonte: Autoria própria.

Depois da conexão ser estabelecida com o aplicativo de celular, o jogo fica recebendo todos os dados. Que são encaminhados para os códigos responsáveis por rodar o jogo. A tela muda para a janela de seleção entre jogo ou editor de fases, Figura 24. Caso a opção jogo seja escolhida, é aberta a tela para escolha do paciente, Figura 25. Onde é possível digitar o nome do paciente manualmente ou escolher um dos pacientes já registrados nas pastas.

Figura 24 – Tela para escolher entre jogo e editor de fases

Fonte: Autoria própria.

Figura 25 – Tela para escolher paciente que vai jogar

Fonte: Autoria própria.

Tendo sido confirmado qual paciente estará jogando, será aberta a tela de seleção de fases, Figura 26. Nela são exibidas todas as fases salvas na pasta "Fases" no formato JSON. Para selecionar uma basta clicar na fase desejada. Também é fornecida a opção de escolher o modo de jogo dentre as opções disponibilizadas no canto esquerdo dentro do botão de opções (comumente chamado de *dropdown* ou *combobox*). Por fim basta apertar no botão "Jogar".

O jogo então tendo carregado todas as fases, irá passar apenas as fases que serão utilizadas para o vetor "df_todos". De acordo com o modo de jogo que tiver sido selecionado.

Figura 26 – Tela para selecionar fase

Fonte: Autoria própria.

Com as fases carregadas, uma função é chamada para iniciar o jogo. Acompanhada do índice da fase escolhida, do modo de jogo e de uma fase de teste (só relevante para modo de jogo usado pelo Editor de Fase). Nesse momento já estará aberta a tela com toda a interface do jogo rodando como mostra a Figura 27.

Figura 27 – Jogo recém aberto

Fonte: Autoria própria.

O valor recebido do celular será continuamente convertido para o intervalo segundo os parâmetros definidos. O resultado é aplicado à espada do jogo, com uma pequena animação de transição entre os ângulos para que o movimento não fique travado.

Tendo o jogo começado, a função de iniciar desafio define qual a fase atual baseado nas configurações definidas. Então ordena os obstáculos da fase em ordem crescente, baseado no tempo para invocar cada obstáculo. Esses tempos são armazenados em outro vetor para ser usado pelo *timer*(temporizador) de invocar obstáculos.

O *timer* será então inicializado com o tempo do primeiro obstáculo, ao fim do temporizador um obstáculo será invocado para o jogador atingir. Tendo todos os atributos configurados: velocidade, tamanho, índice do *asset* 3D, modificador de pontos e ângulo em que vai aparecer, Figura 28. Então o *timer* será reiniciado com o tempo do próximo obstáculo menos o tempo do obstáculo invocado por último, Figura 29. Isso se repete até todos os obstáculos da fase terem sido invocados na cena.

Figura 28 – Jogo com primeiro obstáculo

Fonte: Autoria própria.

A espada fica constantemente detectando colisões. Caso ela atinja um obstáculo ele será deletado, a pontuação será aumentada na proporção do modificador do obstáculo atingido e uma partícula de explosão aparece onde ele estava antes de sumir, Figura 30. Se a pessoa erra e o obstáculo atravessar o alcance da espada um detector de colisão do fundo do mapa vai detectar esse erro. Deletar o obstáculo que passou e aumentar um contador oculto para saber quantos pontos a pessoa deixou de ganhar.

Quando a fase é concluída uma tela aparece mostrando o desempenho do jogador baseado na razão entre os pontos conseguidos e o total possível. São então atribuídas de 1 a 3 estrelas baseada na porcentagem de acertos, Figura 31.

Também foi implementado um minimapa 2D no canto superior direito da tela, Figura 32. Que permite ter uma noção mais simples de onde estão os obstáculos e qual a localização da espada em relação a posição ideal para atingi-los. Essa ferramenta pode ser ativada ou desativada ao apertar a tecla "m" no teclado.

Figura 29 – Jogo com vários obstáculos

Fonte: Autoria própria.

Figura 30 – Paciente atingiu obstáculo

Fonte: Autoria própria.

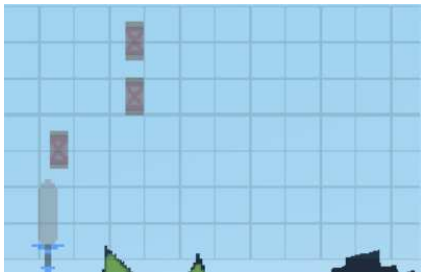
Com o mesmo intuito de facilitar a experiência do jogador, também é exibido um auxílio visual, Figura 33. Ele é uma mira que fica com a posição alinhada ao ângulo em que a espada precisa se encontrar para atingir os próximos obstáculos.

Utilizando os *sprites* e as texturas foram criados 3 modelos 3D de estrada e uma ponte atravessando um rio. Além de 6 conjuntos utilizando 2 planos 3D, para criar uma ilusão de um modelo 3D com as texturas de árvores, pedras, cogumelos, arbustos, etc. Então foi utilizada a lógica mencionada na metodologia para ser criada a ilusão de que o jogador está caminhando em uma estrada infinita, passando por várias partes da floresta e cruzando rios. Mas na verdade o cenário é que está sendo gerado, movido e deletado. Existindo simultaneamente apenas a

Figura 31 – Tela fase concluída

Fonte: Autoria própria.

Figura 32 – Minimapa



Fonte: Autoria própria.

Figura 33 – Auxílio visual

Fonte: Autoria própria.

quantidade de caminhos exibidos na Figura 34.

Com o jogo rodando é possível interromper a fase abrindo o menu de pausa ao apertar a tecla "Esc" do teclado, Figura 35. Nele se encontram as opções de continuar jogando, abrir as configurações de preferências e encerrar a sessão do jogo.

As tela para configurar as preferências do paciente, Figura 36, apresenta 5 variáveis para serem modificadas: delay para invocação, velocidade, tamanho dos obstáculos e ângulos de máximo e mínimo do controle. Esses últimos têm um botão que permite a calibração manual

Figura 34 – Caminho sendo adicionado e removido

Fonte: Autoria própria.

Figura 35 – Tela do menu de pausa

Fonte: Autoria própria.

usando os dados do sensor. Quando os dados forem configurados é possível apertar o botão "Salvar". Para que as informações sejam transferidas para o jogo, sendo aplicadas aos próximos obstáculos que forem invocados. O botão "Voltar" permite retornar ao menu de pausa.

4.4 RELATÓRIO DESEMPENHO

Ao mesmo tempo que o jogo é inicializado, o *timer* de 0.1 segundos começa a executar continuamente. Ao fim de cada contagem a função para registro de histórico é chamada. Ela

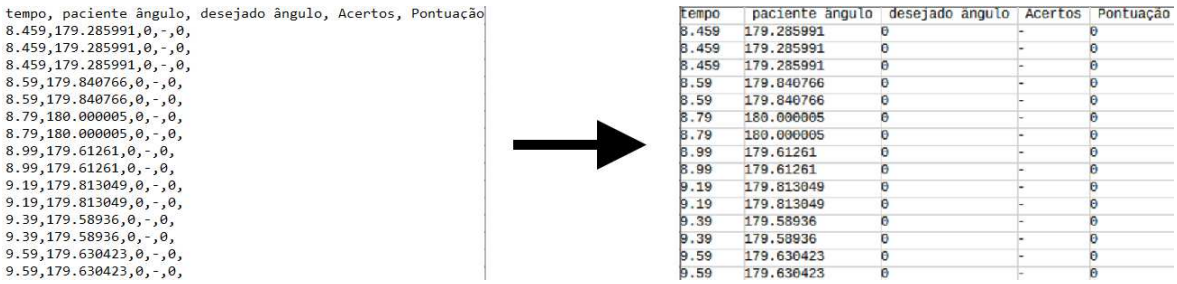
Figura 36 – Tela para configurar preferências do jogador

Fonte: Autoria própria.

adiciona os dados relevantes para a avaliação do desempenho do paciente a um vetor, sendo eles: o tempo atual, o ângulo da mão do paciente, o ângulo ideal para marcar ponto no jogo, se o obstáculo foi atingido ou o paciente errou e a pontuação do paciente no momento. Para garantir precisão, a função de registro de acertos e erros é chamada quando o obstáculo é deletado.

Ao finalizar a sessão de jogo o histórico do vetor será utilizado para a criação de um novo arquivo de CSV, dentro da pasta de desempenho do respectivo paciente que estava jogando. Nele todos os dados do histórico são salvos, Figura 37. Além desses dados de histórico também é atualizado um arquivo JSON onde se encontram as configurações das preferência do paciente. Acompanhada de uma variável com o tempo em que a sessão foi concluída para que o profissional possa comparar as configurações utilizadas em cada dia ou sessão, Figura 38.

Figura 37 – Exemplo de CSV onde o desempenho do paciente está armazenado



```
tempo, paciente ângulo, desejado ângulo, Acertos, Pontuação
8.459,179.285991,0,-,0,
8.459,179.285991,0,-,0,
8.459,179.285991,0,-,0,
8.59,179.840766,0,-,0,
8.59,179.840766,0,-,0,
8.79,180.000005,0,-,0,
8.79,180.000005,0,-,0,
8.99,179.61261,0,-,0,
8.99,179.61261,0,-,0,
9.19,179.813049,0,-,0,
9.19,179.813049,0,-,0,
9.39,179.58936,0,-,0,
9.39,179.58936,0,-,0,
9.59,179.630423,0,-,0,
9.59,179.630423,0,-,0,
```

tempo	paciente ângulo	desejado ângulo	Acertos	Pontuação
8.459	179.285991	0	-	0
8.459	179.285991	0	-	0
8.459	179.285991	0	-	0
8.59	179.840766	0	-	0
8.59	179.840766	0	-	0
8.79	180.000005	0	-	0
8.79	180.000005	0	-	0
8.99	179.61261	0	-	0
8.99	179.61261	0	-	0
9.19	179.813049	0	-	0
9.19	179.813049	0	-	0
9.39	179.58936	0	-	0
9.39	179.58936	0	-	0
9.59	179.630423	0	-	0
9.59	179.630423	0	-	0

Fonte: Autoria própria.

Devido ao formato, os arquivos podem ser facilmente abertos em visualizadores de texto e editores de planilhas. Permitindo que os dados sejam convertidos em gráficos manualmente para fácil análise.

Figura 38 – Exemplo de JSON com preferências do paciente e data


```
{
 "guido": [
  {
   "angulo max": 158.66,
   "angulo min": 64.76,
   "data": "2024-04-10T10:31:47",
   "delay inimigos": 2,
   "tamanho objetos": 1,
   "velocidade objetos": 1
  },
  {
   "data": "2024-04-24T13:26:10",
   "delay inimigos": 2,
   "velocidade objetos": 1,
   "tamanho objetos": 1,
   "angulo max": 180,
   "angulo min": 0
  }
 ]
}
```


Fonte: Autoria própria.

4.5 CRIADOR DE FASES

Por fim, caso se escolha a opção de abrir o editor de fases, Figura 24. Então as etapas de registro de dados do paciente e abertura de fase serão ignoradas. Enquanto a tela do Editor será aberta, Figura 39.

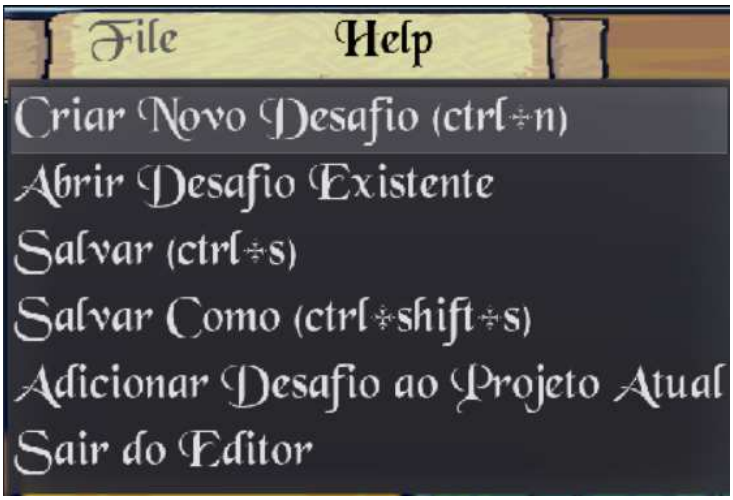
Figura 39 – Tela editor de fases

Fonte: Autoria própria.

No canto superior esquerdo dessa janela se encontram os botões Arquivo e Ajuda, este oferece uma breve explicação do funcionamento do editor e as informações para contato com o desenvolvedor. Enquanto aquele abre as opções para lidar com os arquivos de fases, Figura 40. Sendo elas: "Criar Novo Desafio", apagando as edições atuais (pode ser executado pelo atalho do teclado ctrl+n); "Abrir Desafio Existente", um *popup* para abrir arquivos permite selecionar a fase que se deseja abrir, Figura 41; "Salvar", que vai atualizar os dados do arquivo aberto no

momento (pode ser executado pelo atalho do teclado ctrl+s); "Salvar Como", que vai permitir a criação de um novo arquivo com os dados que estão no editor, por meio de outro *popup* (pode ser executado pelo atalho do teclado ctrl+shift+n); "Adicionar Desafio ao Projeto Atual", que permite que uma fase seja importada e os seus obstáculos sejam colocados no final da fase atual; "Sair do Editor", que fecha a ferramenta.

Figura 40 – Botões para mexer com arquivos



Fonte: Autoria própria.

Figura 41 – *Pop Up* para abrir arquivo

Fonte: Autoria própria.

Quando uma fase é aberta os obstáculos, são salvos no vetor da fase e também tem uma representação visual adicionada ao fundo móvel, Figura 42. Onde a posição no eixo x é relativa ao ângulo (de 0 a 180) do obstáculos, enquanto a posição no eixo y depende do tempo em que o obstáculo será invocado (que precisa ser maior que zero).

Cada obstáculo recebe um índice e podem ser selecionados com o mouse. Então seus respectivos atributos são carregados na interface para serem editados, Figura 43. Quando os

Figura 42 – Interface gráfica de fundo móvel

Fonte: Autoria própria.

valores são editados é realizada uma checagem para ver se respeitam os limites de cada variável e então eles substituem os valores anteriores dentro do vetor de desafios.

Caso o tempo ou ângulo sejam modificado a posição do botão na interface gráfica é atualizada. O ângulo pode ser alterado através da digitação do valor, pelo movimento de um *slider* ou pode ser definido como aleatório por meio de um *checkbutton* (botão que funciona como um interruptor). Este que define o valor do ângulo como -1 (quando o jogo o encontra ele aleatoriza entre 0 e 180).

Figura 43 – Obstáculo selecionado e dados para edição

Fonte: Autoria própria.

Nos cantos inferior direito, Figura 39 também se encontram os botões de adicionar e duplicar. O primeiro cria um novo obstáculo com valores padrões no atributo, enquanto o segundo

duplica os dados do obstáculo que esteja selecionado. Do outro lado no canto inferior esquerdo se encontram os botões de remover, que deleta o obstáculo selecionado (esses três botões fazem as atualizações necessárias, tanto na interface visual, quanto no vetor com os dicionários dos obstáculos da fase).

Por último, logo acima do botão de deleção, está o botão para testar a fase sendo desenvolvida, ele envia um sinal para iniciar o jogo no o modo de "teste" (índice -2). Neste modo, ao invés de mandar o índice de uma fase aberta o programa envia o vetor com a fase sendo desenvolvida. O jogo então executa normalmente utilizando apenas a fase de teste. Quando o último obstáculo da fase é destruído o jogo abre o editor de fases com os mesmos dados de antes.

5 CONCLUSÃO

Assim, a maioria dos objetivos do projeto foram alcançados: o aplicativo de controle lendo o ângulo do antebraço pelo celular; os suportes para celular, mão e braço; o jogo 3D onde uma espada controlada pelos movimentos de supinação e pronação é utilizada para marcar os pontos; A configuração ajustável dos parâmetros de jogo para se adaptar às necessidades do jogador; A geração de relatórios com o desempenho dos jogadores, além de salvar as configurações das preferências; o Editor de Fases que permite modificar e combinar as fases já existentes ou criar novas do zero; tudo utilizando ferramentas acessíveis, bastando ter acesso a um computador e celular na mesma rede para rodar o jogo. Faltando apenas a realização de testes para o ciclo de *feedback* com os pacientes.

Mesmo que o jogo já se encontre em uma situação viável para lançamento ainda seriam interessantes a realização de mais testes, para encontrar *bugs* e outros possíveis problemas. Além de se coletar o *feedback* dos pacientes ao testar o jogo e levar em conta as sugestões ou dificuldades encontradas.

Também seria interessante a implementação de mais conteúdo, para tornar o jogo mais imersivo e evitando que ele fique muito repetitivo. Como por exemplo: novos modos de jogo, como um que tenha um sistemas de vidas que são perdidas ao errar os obstáculos; novos conjuntos de *Assets* para o cenário, jogador e *props* (objetos com que o jogador pode interagir), como com uma estética noturna, passando por outros biomas, cidades ou até futurista com robôs e armas de ficção científica; *Power ups* para maior variedade no ciclo de jogo; um contador de combos incentivando mais ainda a não errar os movimentos; uma aura ao redor dos obstáculos para ser mais fácil de enxergá-los; uma fase ou tela de tutorial que ensine de maneira simples a utilizar o jogo e o editor de fases; e configurações visuais que permitam ajustar as preferências do usuário quanto a aparência do jogo.

Com a conclusão dos objetivos proposto, o desenvolvimento do jogo ainda poderá ser levado adiante, e os conhecimentos adquiridos com a experiência desse desenvolvimento permitem a criação de *Serious Games* para muitas áreas mais.

REFERÊNCIAS

ABT, C.C. **Serious Games**. University Press of America, 1987. ISBN 9780819161482. Disponível em: https://books.google.com.br/books?id=axUs9HA-hF8C. Citado na página 8.

ADLER2019. **Medieval Game Assets**. 2020. Disponível em: https://blendswap.com/blend/24379. Citado na página 27.

BULLOCK, Matt. **Who started gamification? Gamification**. 2023. Disponível em: https://spinify.com/blog/gamification-history/. Citado 2 vezes nas páginas 8 e 14.

CAINOS. **Pixel Art Platformer - Village Props**. 2021. Disponível em: https://cainos.itch.io/pixel-art-platformer-village-props. Citado 2 vezes nas páginas 27 e 37.

DANIEL74. **Medieval Props 2**. 2012. Disponível em: https://blendswap.com/blend/6695. Citado na página 27.

____. **Medieval Containers**. 2014. Disponível em: https://blendswap.com/blend/13795. Citado na página 27.

EVAN-AMOS. **The Microsoft Kinect peripheral for the Xbox 360.** 2010. Disponível em: https://pt.m.wikipedia.org/wiki/Ficheiro:Xbox-360-Kinect-Standalone.png. Citado na página 19.

____. **Wii-Console.png**. 2022. Disponível em: https://pt.wikipedia.org/wiki/Wii#/media/Ficheiro:Wii-Console.png. Citado na página 19.

FERREIRA, Bruno; MENEZES, Paulo. Gamifying motor rehabilitation therapies: Challenges and opportunities of immersive technologies. **Information**, v. 11, n. 2, 2020. ISSN 2078-2489. Disponível em: https://www.mdpi.com/2078-2489/11/2/88. Citado 2 vezes nas páginas 9 e 12.

FROG, Pixel. **Tiny Swords**. 2022. Disponível em: https://pixelfrog-assets.itch.io/tiny-swords. Citado 2 vezes nas páginas 27 e 37.

HEIDMANN, Olivier. How to create a serious game? **EAI Endorsed Transactions on Game-Based Learning**, v. 2, p. 150608, 11 2015. Disponível em: https://www.researchgate.net/publication/283575177_How_to_create_a_serious_game. Citado na página 15.

LEOHPAZ. **RPG Essentials SFX - Free!** 2022. Disponível em: https://leohpaz.itch.io/rpg-essentials-sfx-free. Citado na página 27.

LUXWOLDA. **Dutch Coffee Font**. 2024. Disponível em: https://www.1001fonts.com/dutch-coffee-font.html. Citado na página 27.

MARTINEZ, LEANDRO. **Octalysis: Um framework para gamificação - Catarinas**. 2021. Disponível em: https://catarinasdesign.com.br/octalysis-um-framework-para-gamificacao/. Citado na página 15.

MASMOUDI, Mostefa *et al.* Assessing the effectiveness of virtual reality serious games in post-stroke rehabilitation: A novel evaluation method. **Scientific Figure on ResearchGate**, jun. 2023. Disponível em: https://www.researchgate.net/figure/a-Serious-game-interface-Arcade-b-Pronation-Supination-of-the-forearm_fig5_371995271. Citado 2 vezes nas páginas 9 e 12.

MEIRELLES, FERNANDO S. **Pesquisa do Uso da TI -Tecnologia de Informação-nas Empresas**. 2023. Disponível em: https://eaesp.fgv.br/sites/eaesp.fgv.br/files/u68/pesti-fgvcia-2023_0.pdf. Citado 2 vezes nas páginas 8 e 18.

MILES, Doug. **BlackChancery Font**. 2007. Disponível em: https://www.1001fonts.com/blackchancery-font.html. Citado 2 vezes nas páginas 27 e 37.

MISTERTECH. **Car or wall mount gripping smartphone holder system**. 2016. Trabalho sob licença Creative CommonsAttribution 4.0 International License disponível em: https://creativecommons.org/licenses/by/4.0/. Disponível em: https://www.thingiverse.com/thing:1778271. Citado 2 vezes nas páginas 20 e 35.

MOREIRA, Guido. **TCC**. 2023. Disponível em: https://github.com/guidoMoreira/TCC. Acesso em: 29 mai. 2024. Citado na página 34.

MOREIRA GUIDO MARGONAR; CAMPOS, Daniel Prado de. Envio de dados em tempo real de sensor em microcontrolador para aplicaÇÃo web em python. **Anais do XII Seminário de Extensão e Inovação XXVII Seminário de Iniciação Científica e Tecnológica da UTFPR**, v. 2, p. 7, dez 2022. Disponível em: https://even3.blob.core.windows.net/anais/547645.pdf. Citado na página 21.

MOTION, Leap. **Leap Motion LM-C01-DE Controle de Gesto com Cabo Preto**. 2022. Disponível em: https://www.amazon.com.br/Leap-Motion-LM-C01-Controle-Gesto/dp/B00H45P892. Citado na página 19.

MRELIPTIK. **godot_experiments**. 2021. Disponível em: https://github.com/MrEliptik/godot_experiments/tree/3.x. Citado 2 vezes nas páginas 24 e 34.

P CARRATAL-TEJADA M, Monge-Pereira E Collado-Vázquez S Sánchez-Herrera Baeza P Cuesta-Gómez A Oña-Simbaña ED Jardón-Huete A Molina-Rueda F Balaguer-Bernaldo de Quirós C Miangolarra-Page JC Cano-de la Cuerda R Fernández-González. Leap motion controlled video game-based therapy for upper limb rehabilitation in patients with Parkinson's disease: a feasibility study. **J Neuroeng Rehabil**, nov. 2019. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6836460/. Citado 3 vezes nas páginas 9, 12 e 33.

PELLING, Nick. **The (short) prehistory of "gamification"...** 2009. Disponível em: https://nanodome.wordpress.com/2011/08/09/the-short-prehistory-of-gamification/. Citado na página 8.

PLATAO. **Leis**. [S.l.]: Project Gutenberg, 1999. Citado na página 8.

POLIIGON. **Free Textures**. 2016. Trabalho sob licença disponível em: https://help.poliigon.com/en/articles/8749749-asset-use-licensing. Disponível em: https://www.poliigon.com/textures/free. Citado na página 27.

PRINBLES. **robin**. 2023. Disponível em: https://prinbles.itch.io/robin. Citado na página 27.

SHADE. **16x16 Weapon RPG Icons**. 2022. Disponível em: https://merchant-shade.itch.io/free-16x16-weapon-rpg-icons. Citado na página 27.

SHONONOKI. **Free RPG Music Pack**. 2020. Disponível em: https://shononoki.itch.io/rpg-music-pack-svl. Citado na página 27.

SOLUTIONS, S.A. Novageo. **Protocolo WebSocket: O que é e para que serve**. 2022. Disponível em: https://www.novageo.pt/novageo/novageo/displayArticles?numero=38362&protocolo_websocket_que__para_que_serve. Citado na página 22.

TOMMUSIC. **Free Fantasy 200 SFX Pack**. 2023. Disponível em: https://tommusic.itch.io/free-fantasy-200-sfx-pack. Citado na página 27.

TOUILLEMAN. **godot-python**. 2022. Disponível em: https://github.com/touilleMan/godot-python. Citado na página 24.

ANEXO A – PROJETO DO SUPORTE DE MADEIRA COM MEDIDAS