PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE JOGOS UTILIZANDO INTERFACE TANGÍVEL

RICARDO ALVES RODRIGUES

GOIÂNIA
2025

RICARDO ALVES RODRIGUES

DESENVOLVIMENTO DE JOGOS UTILIZANDO INTERFACE TANGÍVEL

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade de Goiás, como parte dos requisitos para obtenção do título de Bacharel em Engenharia da Computação.
Prof. Orientador: Dr. Talles Marcelo G. de A. Barbosa

GOIÂNIA
2025

RICARDO ALVES RODRIGUES

DESENVOLVIMENTO DE JOGOS UTILIZANDO INTERFACE TANGÍVEL

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia da Computação, em:_____/_____/_____

_____

Prof. Orientador: Dr. Talles Marcelo G de A Barbosa

AGRADECIMENTOS

Agradeço profundamente à minha mãe, cuja fé constante em mim e cujo papel iluminador em minha trajetória foram decisivos. Sua força e seu amor moldaram quem eu sou e sustentam cada conquista que alcancei.

Ao meu pai, deixo minha gratidão mais sincera. Obrigado por sua presença contínua, pelos esforços feitos em meu favor e por acreditar no meu potencial.

Ao meu irmão, meu companheiro de sempre e grande amigo, registro meu agradecimento especial. Seu apoio inabalável e sua parceria foram fundamentais para enfrentar dificuldades e celebrar vitórias.

À minha esposa, deixo meu mais profundo agradecimento. Obrigado por ser minha companheira em todos os momentos, por seu amor incondicional e por acreditar em mim mesmo quando eu duvidei de mim. Sua compreensão, seu apoio diário e sua presença afetuosa foram essenciais para que eu chegasse até aqui. Você é minha inspiração e minha força, e sou eternamente grato por compartilhar esta jornada ao seu lado.

Ao meu orientador, Dr. Talles Barbosa, manifesto minha gratidão pela dedicação e pela paciência. Seus conselhos, sua experiência e seu incentivo tiveram papel essencial na conclusão do meu TCC.

Aos professores que fizeram parte da minha formação durante o curso, meu muito obrigado. Vocês não só transmitiram conhecimento, como também inspiraram e contribuíram diretamente para o profissional que me tornei.

Aos meus colegas de faculdade, agradeço por cada momento vivido, pelas conversas que ampliaram meu aprendizado e pela união que deixou essa caminhada acadêmica mais leve e significativa.

RESUMO

Este trabalho apresenta o desenvolvimento de jogos interativos para o sistema 3D SandPlay, uma plataforma de interface tangível que projeta elementos virtuais sobre uma caixa de areia. Interface tangível é um tipo de interface na qual as pessoas interagem com informações digitais através do ambiente físico, onde objetos físicos são acoplados computacionalmente às informações digitais subjacentes (Ishii; Ullmer, 1997). O objetivo foi desenvolver dois novos jogos: o Jogo de Sobrevivência e o Jogo da Alimentação. Os jogos foram desenvolvidos utilizando o framework *openFrameworks* com integração do sensor Kinect para captura de profundidade e reconstrução 3D do terreno. Algoritmos de direcionamento de comportamento (*steering behaviors*) foram adaptados para criar movimentos realistas de perseguição, evasão e coleta para os agentes virtuais em um ambiente dinâmico. O trabalho resultou em um protótipo funcional que demonstra a viabilidade técnica de integrar mecânicas de jogo complexas a interfaces tangíveis, estabelecendo base para futuras aplicações. Além disso, confirma-se que sistemas de projeção mapeada podem ser estendidos predominantemente via *software*, com relevância educacional, terapêutica e tecnológica, oferecendo um modelo sustentável e replicável para instituições com recursos limitados.

Palavras-chave: Interface Tangível. Realidade Aumentada. 3D SandPlay. Jogos Interativos. *Steering Behaviors*. Projeção Mapeada. Kinect. *openFrameworks*. Interação Física-Digital. Educação e Terapia.

ABSTRACT

This work presents the development of interactive games for the 3D SandPlay system, a tangible interface platform that projects virtual elements onto a sandbox. A tangible interface is a type of interface in which people interact with digital information through the physical environment, where physical objects are computationally coupled to underlying digital information (Ishii; Ullmer, 1997). The objective was to develop two new games: the Survival Game and the Feeding Game. The games were developed using the openFrameworks framework with integration of the Kinect sensor for depth capture and 3D terrain reconstruction. Steering behavior algorithms were adapted to create realistic pursuit, evasion, and collection movements for virtual agents in a dynamic environment. The work resulted in a functional prototype that demonstrates the technical feasibility of integrating complex game mechanics into tangible interfaces, establishing a foundation for future applications. Furthermore, it confirms that projection-mapped systems can be extended predominantly through software, with educational, therapeutic, and technological relevance, offering a sustainable and replicable model for institutions with limited resources.

Keywords: Tangible Interface. Augmented Reality. 3D SandPlay. Interactive Games. Steering Behaviors. Projection Mapping. Kinect. openFrameworks. Physical-Digital Interaction. Education and Therapy.

LISTA DE FIGURAS

LISTA DE SIGLAS

AR – Augmented Reality (Realidade Aumentada)

RA – Realidade Aumentada

TUI – Tangible User Interface (Interface Tangível de Usuário)

MIT – Massachusetts Institute of Technology

UC Davis – University of California, Davis

UEL – Universidade Estadual de Londrina

PC – Personal Computer

CPU – Central Processing Unit

GPU – Graphics Processing Unit

IDE – Integrated Development Environment

SDK – Software Development Kit

USB – Universal Serial Bus

FPS – Frames Per Second

API – Application Programming Interface

Kinect SDK – Kinect Software Development Kit

OF – openFrameworks

GPL – General Public License

CPU – Central Processing Unit

RAM – Random Access Memory

libusb – Biblioteca de comunicação USB

libusbK – Driver/lib para comunicação USB

libusb-win32 – Driver USB para Windows

libfreenect – Biblioteca de acesso ao Kinect

OpenCV – Open Source Computer Vision Library

SUMÁRIO

1 INTRODUÇÃO

Interfaces tangíveis representam uma categoria especial de interfaces de usuário em que as pessoas interagem com informações digitais através do ambiente físico, onde objetos físicos são acoplados computacionalmente às informações digitais subjacentes, incorporando mecanismos para controle interativo e sendo perceptualmente acoplados às representações digitais mediadas ativamente (Science Direct, 2025; Wikipedia, 2025). Este paradigma, inicialmente concebido como "Graspable User Interface" e posteriormente desenvolvido por Hiroshi Ishii e seu grupo de pesquisa no MIT Media Lab, visa dar forma física à informação digital, permitindo que bits se tornem diretamente manipuláveis e perceptíveis através do princípio do "Tangible Bits" (Ishii; Ullmer, 1997; Shaer; Hornecker, 2010).

Neste contexto, o sistema 3D SandPlay surge como uma plataforma de interface tangível que projeta elementos virtuais sobre uma caixa de areia. O sistema utiliza um sensor de profundidade (Kinect) e um projetor para criar uma interface onde a manipulação direta da areia é capturada e traduzida em projeções visuais que respondem em tempo real às mudanças topográficas (Dourado et al., 2019). Este trabalho propõe desenvolver e implementar dois novos modos de jogo interativos para o 3D SandPlay utilizando o framework *openFrameworks* e sensor Kinect, explorando conceitos de comportamentos de direcionamento para criar experiências lúdicas baseadas na manipulação física da areia.

1.1 Problema e Desafios

O 3D SandPlay, desenvolvido por Dourado et al. (2019) na Pontifícia Universidade Católica de Goiás, foi inicialmente projetado como uma interface tangível que pode ser utilizada como reforçador para o desenvolvimento intelectual e da coordenação motora, com foco especial em pessoas com Síndrome de Down (Dourado et al., 2019).
O projeto parte da hipótese exploratória de que o uso do 3D SandPlay pode facilitar a compreensão de conceitos abstratos e, simultaneamente, promover a integração sensorial neste público (Dourado et al., 2019).

Desta forma, o principal problema do reuso desse ambiente, são as integrações de versões e sistemas. Pois é preciso utilizar versões especificas de ambiente de desenvolvimento (Visual Studio) e bibliotecas de integração (*openFrameworks*).

Como desafio deste trabalho, busca-se expandir as capacidades do 3D SandPlay através do desenvolvimento de jogos interativos que aproveitem a natureza tangível da interface, proporcionando experiências significativas que conectem o mundo físico e virtual de forma

intuitiva. Especificamente, respondendo a seguinte questão: Como reutilizar o 3D Sandplay, para criação de jogos tangíveis?

1.2 Hipótese

A hipótese deste trabalho é mostrar que é possível desenvolver jogos para o 3D SandPlay, que utilizem a manipulação do relevo da areia como mecanismo principal de interação, criando experiências envolventes que demonstram a viabilidade técnica desta abordagem. Propõe-se que estes jogos possam servir como base para futuras aplicações educacionais e terapêuticas, mas esta validação específica foge do escopo técnico deste trabalho. Esta hipótese será testada através da implementação de dois jogos específicos: o Jogo de Sobrevivência, que exigirá a criação de barreiras físicas na areia para proteger peixes virtuais de tubarões predadores; e o Jogo da Alimentação, que desafiará os jogadores a criar canais na areia para guiar peixes até alimentos virtuais. A adaptação de algoritmos de comportamentos de direcionamento para agentes virtuais em ambiente dinâmico é fundamental para a realização desta hipótese, implementando comportamentos realistas de perseguição, evasão e coleta.

1.3 Justificativa

A justificativa para este trabalho reside em aspectos de custo, pois reutilizar a plataforma existente (*hardware* e *software*) permite economizar recursos financeiros e tempo de desenvolvimento. Além disso, a criação de diferentes jogos possibilita a exploração de múltiplos contextos de uso, desde entretenimento até atividades específicas de reabilitação, aproveitando o estímulo tátil ampliado pela percepção tridimensional (Adaptado de Kreylos, 2014).

2 REVISÃO DA LITERATURA

2.1 Interfaces Tangíveis e suas Características

Interfaces tangíveis de usuário (*Tangible User Interfaces* - TUIs) foram inicialmente concebidas como "*Graspable User Interface*" e posteriormente desenvolvidas por Hiroshi Ishii e seu grupo de pesquisa no MIT Media Lab. O conceito visa dar forma física à informação digital, permitindo que bits se tornem diretamente manipuláveis e perceptíveis através do princípio do "*Tangible Bits*" (Ishii; Ullmer, 1997).

Segundo Shaer e Hornecker (2010), interfaces tangíveis proporcionam uma forma de interação mais natural e intuitiva, especialmente para populações que podem ter dificuldade com interfaces tradicionais baseadas em telas e teclados. Esta característica faz das TUIs uma opção promissora para aplicações educacionais e terapêuticas, onde a interação física pode facilitar a compreensão de conceitos abstratos.

2.2 Sistemas de Caixa de Areia Interativa

O *Augmented Reality Sandbox*, originalmente desenvolvido pela Universidade da Califórnia em Davis (UC Davis), representa um marco significativo na área de interfaces tangíveis. Este sistema combina um sensor de profundidade (como Kinect), um projetor e um computador para criar uma interface interativa onde as elevações e depressões moldadas na areia são visualizadas através de cores e linhas de contorno projetadas (Kreylos, 2014).

No Brasil, o projeto 3D SandPlay, desenvolvido por Dourado et al. (2019), adapta este conceito para aplicações terapêuticas, especificamente como uma interface tangível para o desenvolvimento intelectual e da coordenação motora de pessoas com Síndrome de Down. O sistema foi desenvolvido na Pontifícia Universidade Católica de Goiás utilizando o *framework openFrameworks* e *o toolkit ofxKinectProjectorToolkit*, com uma estrutura física otimizada de 45cm de comprimento, 33cm de largura e 7cm de altura, montada em uma estrutura metálica de 90cm de altura (Josefoberdan, 2022).

2.3 Algoritmos de Direcionamento de Comportamento

Algoritmos de direcionamento de comportamento (*steering behaviors*) são técnicas que controlam a movimentação de agentes autônomos em ambientes virtuais, permitindo que estes agentes respondam de forma realista a estímulos do ambiente. Craig Reynolds, pioneiro nesta área, descreveu como estes algoritmos podem simular comportamentos complexos como formação de cardumes, evasão de obstáculos e perseguição de alvos (Adaptado de Reynolds, 1999).

Estes algoritmos são particularmente relevantes para este trabalho, pois permitem que os agentes virtuais (peixes e tubarões) respondam dinamicamente às mudanças no ambiente físico (a topografia da areia), criando uma experiência de interação mais imersiva e realista. A adaptação destes algoritmos para ambientes dinâmicos, onde o terreno muda continuamente, representa um desafio significativo de engenharia de software.

2.4 Gamificação em Interfaces Tangíveis

Gamificação refere-se à aplicação de elementos e princípios de design de jogos em contextos não-lúdicos para aumentar o engajamento e a motivação (Deterding et al., 2011). Quando aplicada a interfaces tangíveis, a gamificação pode potencializar os benefícios da interação física, tornando as experiências mais envolventes e significativas.

No contexto educacional e terapêutico, jogos que utilizam interfaces tangíveis podem proporcionar um ambiente de aprendizado mais inclusivo e acessível. Estudos demonstram que a combinação de estímulos visuais, táteis e sonoros pode aumentar a atenção e facilitar o aprendizado, especialmente para populações com necessidades especiais (Rossit, 2003; Lima et al., 2017).

3 METODOLOGIA

3.1 Metodologia de Desenvolvimento

O processo de desenvolvimento deste trabalho seguiu uma abordagem *bottom-up* incremental, verificando e adaptando os recursos disponíveis para a especificação das aplicações e casos de uso, a *posteriori*. Esta metodologia foi escolhida por permitir uma maior flexibilidade na integração com o sistema 3D SandPlay existente, aproveitando sua base funcional enquanto se adicionavam novas capacidades.

A abordagem *bottom-up* foi fundamental devido às restrições técnicas do projeto original, que exigiam compatibilidade completa com versões específicas de bibliotecas e drivers. Em vez de reestruturar o sistema existente, optou-se por construir sobre a infraestrutura já funcional, garantindo estabilidade e reduzindo o risco de introduzir falhas no núcleo do sistema de projeção mapeada.

O desenvolvimento modular dos jogos e classes permitiu uma organização clara do código e facilitou a manutenção. Cada componente (Jogo de Sobrevivência, Jogo da Alimentação, sistemas de comportamento de agentes) foi implementado como um módulo independente com interfaces bem definidas, possibilitando testes isolados e integração progressiva.

A implementação de sistemas e integração com APIs existentes seguiu princípio fundamental de não modificar as bibliotecas originais do 3D SandPlay. Esta decisão estratégica preservou a integridade do sistema original e permitiu que as novas funcionalidades fossem adicionadas através de extensões e classes derivadas, mantendo a possibilidade de atualizações futuras sem perder as customizações implementadas.

3.2 Tecnologias Utilizadas

A implementação dos jogos para o 3D SandPlay exigiu uma configuração técnica específica devido às características do projeto original e às limitações de compatibilidade entre as tecnologias. Esta seção detalha todas as tecnologias utilizadas, justificando a escolha de cada uma e explicando sua importância para o desenvolvimento do sistema.

3.2.1 Dependências de Versões de Software

O desenvolvimento deste trabalho exigiu o uso de versões específicas de software devido à natureza do projeto 3D SandPlay original e às restrições de compatibilidade entre componentes, como, Visual Studio 2015, openFrameworks, Kinect SDK V1 1 e Zadig.

Visual Studio 2015, foi necessário manter o uso do Visual Studio 2015 como ambiente de desenvolvimento integrado (IDE) por várias razões críticas. Primeiro, o addon ofxKinect apresenta incompatibilidades sérias com versões mais recentes do Visual Studio (2017 e 2019), exigindo modificações extensivas de código fonte original para funcionar adequadamente. Segundo, a arquitetura de build do openFrameworks 0.98 foi otimizada especificamente para o Visual Studio 2015, com configurações de projeto e templates que não funcionam corretamente em versões posteriores. Terceiro, a estrutura de solução (.sln) e os arquivos de projeto (.vcxproj) do 3D SandPlay original foram configurados para o Visual Studio 2015, e a migração para versões mais recentes quebraria a estrutura de pastas e dependências do projeto. Esta escolha, embora aparentemente retrógrada, permitiu que o foco do desenvolvimento permanecesse nas mecânicas de jogo em vez de ser desviado para resolver problemas de compatibilidade de plataforma.

O *openFrameworks* é *um toolkit open source* em C++ para codificação criativa (*creative coding*), descrito como uma "ferramenta de código aberto para programação criativa". A versão 0.98 foi selecionada especificamente por sua compatibilidade com o Visual Studio 2015 e com as bibliotecas essenciais do projeto. Versões mais recentes do *openFrameworks* introduziram mudanças significativas na API e na estrutura de diretórios que quebrariam a compatibilidade com o código base do 3D SandPlay. A versão 0.98 oferece suporte para o sensor Kinect v1 através do *addon ofxKinect*, além de possuir uma comunidade ativa de usuários que mantém compatibilidade com sistemas legados. Além disso, esta versão apresenta um equilíbrio ideal entre estabilidade e recursos avançados, com bibliotecas otimizadas para processamento de vídeo em tempo real e renderização OpenGL acelerada por *hardware*.

O Kinect SDK V1 1.0 foi escolhido devido à sua compatibilidade com o hardware Kinect v1 (para Xbox 360) e sua integração com o *openFrameworks* 0.98. Versões mais recentes do SDK foram desenvolvidas exclusivamente para o Kinect v2 (para Xbox One) e não oferecem suporte para o hardware utilizado neste projeto. O Kinect SDK V1 1.0 fornece uma interface de programação estável e bem documentada para acesso aos dados de profundidade. Além disso, este SDK foi amplamente testado em ambientes Windows 10 e apresenta melhor performance em sistemas com configurações mais modestas, o que é essencial para garantir uma taxa de quadros estável durante a execução dos jogos.

O Zadig é uma aplicação Windows que instala drivers USB genéricos, como WinUSB, libusb-win32/libusb0.sys ou libusbK (Aeko Consulting, 2025). Foi necessário utilizá-lo, porque os drivers padrão do Windows para o sensor Kinect v1 não permitem acesso completo às

funcionalidades do dispositivo através do *openFrameworks*. Os drivers oficiais da Microsoft bloqueiam o acesso direto ao Kinect, limitando a comunicação apenas às APIs do Kinect SDK. O Zadig permite instalar drivers genéricos (especificamente libusb-win32) que possibilitam comunicação direta com o dispositivo no nível de protocolo USB, contornando estas limitações e permitindo que bibliotecas como *libfreenect* acessem dados brutos do sensor. Esta atenção especial à instalação dos drivers é crítica porque qualquer erro neste passo tornaria impossível a comunicação entre o sensor e o software, inviabilizando todo o sistema. O Anexo A, referência os fabricantes para baixar e instalar esses *softwares*.

3.2.2 Dependências de Bibliotecas

As bibliotecas são elementos fundamentais que viabilizam toda a arquitetura do sistema 3D SandPlay e dos novos jogos desenvolvidos, as principais utilizadas foram, *openFrameworks (ofxKinect, ofxOpenCv e ofxXmlSettings)*, l*ibfreenect e libusb-win32.*

ofxKinect (MIT License), esta biblioteca é essencial para o funcionamento do sistema. Ela encapsula toda a complexidade da comunicação com o sensor *Kinect*, traduzindo hardware complexo em uma interface simples para o *openFrameworks*. Sem esta biblioteca, seria necessário programar diretamente com a API do *Kinect SDK*, onde acarretaria custo de tempo, somente para capturar um quadro de profundidade. Além disso, o *ofxKinect* abstrai a complexidade dos drivers e proporciona integração nativa com outras bibliotecas do *openFrameworks*, como *ofxOpenCv*. Sua licença MIT permite modificação e distribuição comercial, o que foi crucial para a adaptação do sistema às necessidades específicas dos jogos.

ofxOpenCv, esta biblioteca é crítica para o processamento de visão computacional necessário para detectar as variações topográficas na areia. Ela transforma os dados brutos do *Kinect* em informações utilizáveis, como contornos, áreas planas e gradientes de elevação. Sem o *ofxOpenCv*, seria impossível detectar com precisão as características do terreno que formam a base das mecânicas de interação dos jogos.

ofxXmlSettings, esta biblioteca é fundamental para a usabilidade e manutenção do sistema. Permite a persistência de configurações críticas, como a calibração do *Kinect* e projetor, configurações dos jogos (dificuldade, tempo de jogo, quantidade de agentes), parâmetros dos algoritmos de *steering behaviors* e definições de níveis e progressão.

A capacidade de salvar e carregar estas configurações sem recompilar o código é essencial para testes rápidos e ajustes finos durante o desenvolvimento.

A biblioteca *libfreenect*, licenciada sob a Apache License 2.0, constitui a camada fundamental de hardware que permite acesso de baixo nível aos dados provenientes do sensor Kinect. Diferentemente do Kinect SDK, que oferece uma interface de alto nível com funcionalidades mais restritas, a *libfreenect* proporciona um controle granular sobre diversos parâmetros do dispositivo. Isso inclui a capacidade de ajustar a exposição e o ganho da câmera de profundidade, selecionar modos específicos de resolução e taxa de quadros, controlar ativamente o emissor de infravermelho (IR) para otimizar o desempenho em diferentes condições de iluminação, e acessar diretamente os dados brutos do sensor antes de qualquer processamento interno. Essa flexibilidade, aliada à sua independência de plataforma, é essencial para garantir a aquisição de dados de alta qualidade em uma variedade de ambientes operacionais, assegurando a consistência e a confiabilidade do sistema independentemente das condições externas.

Já a biblioteca *libusb-win32*, licenciada sob a GPL v3, atua como a base da conectividade física entre o computador e o dispositivo Kinect. Ela permite a comunicação direta com o sensor no nível do protocolo USB, superando as limitações dos drivers genéricos do Windows em termos de estabilidade e desempenho. Essa robustez é particularmente crítica em aplicações que exigem sessões prolongadas de captura contínua, como é comum em sistemas interativos baseados em realidade aumentada. A compatibilidade da libusb-win32 com hardware legado, especialmente o Kinect v1, também é um fator determinante para a manutenção de sistemas existentes que dependem dessa geração do sensor. A estabilidade fornecida por essa biblioteca evita desconexões inesperadas ou perda de dados durante o uso, aspectos que, se não controlados, poderiam comprometer seriamente a experiência do usuário e a integridade dos dados coletados.

3.2.3 Configuração do Hardware

A configuração do *hardware* foi realizada de forma rigorosa para assegurar o alinhamento preciso entre o sensor Kinect v1, o projetor e o sistema computacional. Essa etapa envolveu a conexão física dos dispositivos, a instalação de drivers de baixo nível via Zadig (utilizando *libusb-win32*) e a instalação complementar do Kinect for Windows SDK v1.0, garantindo tanto compatibilidade quanto estabilidade no ambiente de desenvolvimento. A calibração final foi executada por meio da interface do 3D SandPlay, alinhando a projeção ao mapa de profundidade capturado pelo Kinect. O procedimento completo, incluindo instruções passo a passo e requisitos específicos está detalhado no Apêndice A.

3.2.4 Calibração do sistema utilizando a interface do 3D SandPlay

Antes de iniciar o processo de calibração do sistema 3DSandPlay, é fundamental garantir que a areia esteja devidamente nivelada, pois essa etapa assegura leituras corretas de profundidade. Após essa verificação inicial, na aba de configurações (*Settings*) e selecione a opção *CALIBRATION* e em seguida *MANUALLY DEFINE SAND REGION*, na qual o usuário utiliza o mouse para delimitar manualmente a área correspondente à caixa de areia, informando ao sistema a região exata de interação. Em seguida, ao clicar em ***AUTOMATICALLY CALIBRATE KINECT & PROJECTOR***, inicia-se a calibração automática entre o sensor Kinect e o projetor, processo que ocorre em duas etapas distintas.

A primeira etapa é realizada diretamente sobre a superfície da areia, permitindo o ajuste inicial de leitura e projeção, e, após sua conclusão, o sistema solicita que a caixa de areia seja completamente coberta, (com tampa de papelão, madeira, etc.).

A segunda etapa, responsável pelo refinamento do alinhamento entre captura e projeção, fazendo que a captura de relevos, como bancos de areias e leitos de água sejam identificado pelo Kinect e ao confirmar essa ação pressionando “OK”, o procedimento final de calibração é iniciado. Finalizado estes procedimentos é possível verificar o sucesso da calibração por meio das mensagens exibidas na janela de *Status*, bem como confirmar o correto funcionamento do sistema pela visualização do mapa colorido com linhas de contorno projetadas sobre a areia.

A Figura 1 apresenta o fluxo deste procedimento de calibração para o 3D Sandplay.

Figura 1 – Sistema de Calibração do 3D Sandplay

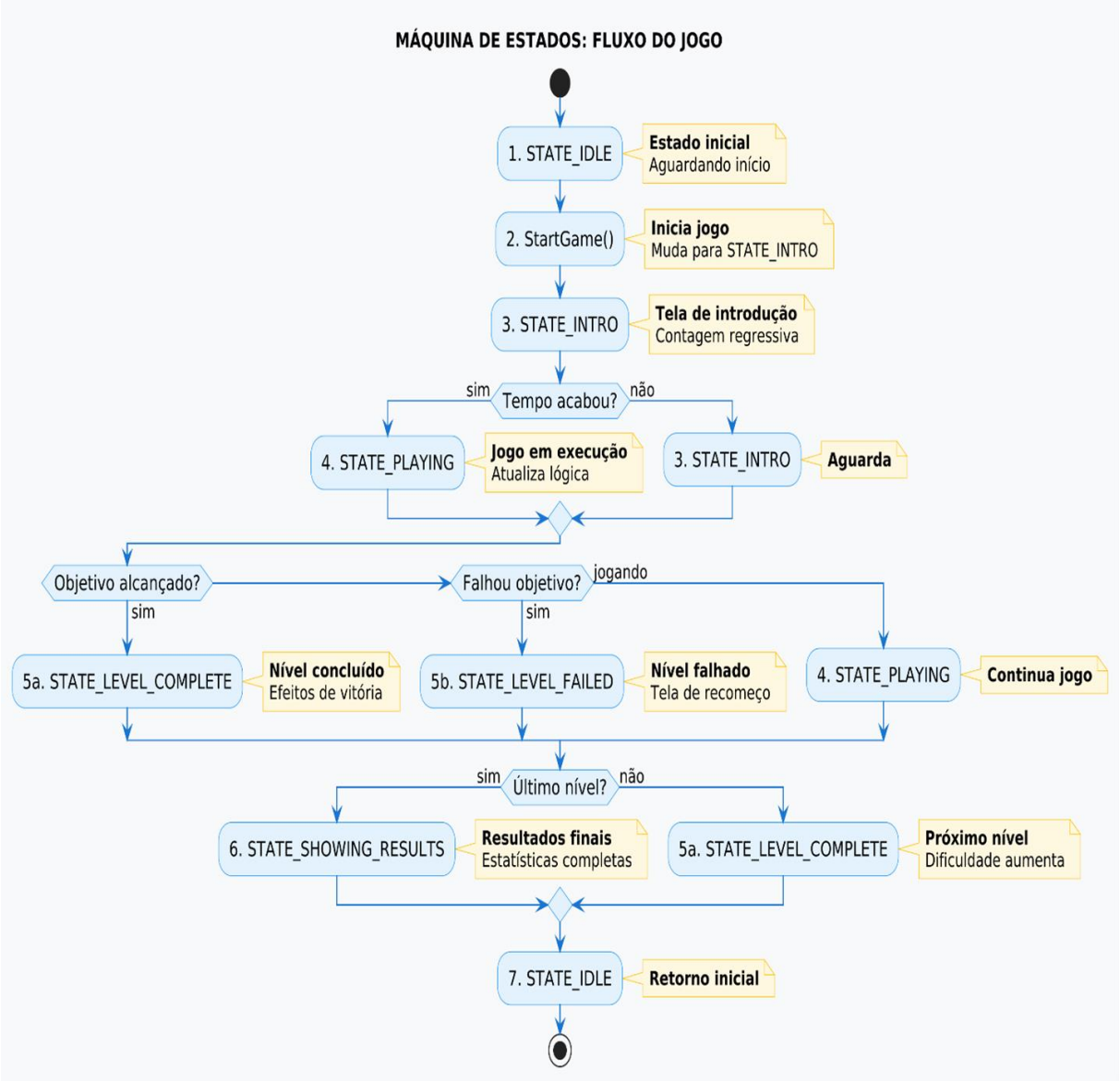
Fonte: Autoria própria

3.3 Apresentação dos Protótipos

3.3.1 Máquinas de Estados

O sistema foi projetado utilizando uma máquina de estados para gerenciar o fluxo dos jogos através de estados discretos (*IDLE*, *INTRO*, *PLAYING*, *LEVEL COMPLETE*, *LEVEL FAILED e RESULTS*.). Cada estado define comportamento específico e transições ocorrem quando condições são atendidas (tempo, objetivos). Esta abordagem centraliza o controle, evita lógica complexa com IF's aninhados e mantém o código organizado e extensível. A classe base *CGameController* orquestra as transições enquanto subclasses implementam lógica específica para cada jogo. A Figura 2 apresenta essa configuração da máquina de estados.

Figura 2 – Máquina de Estados

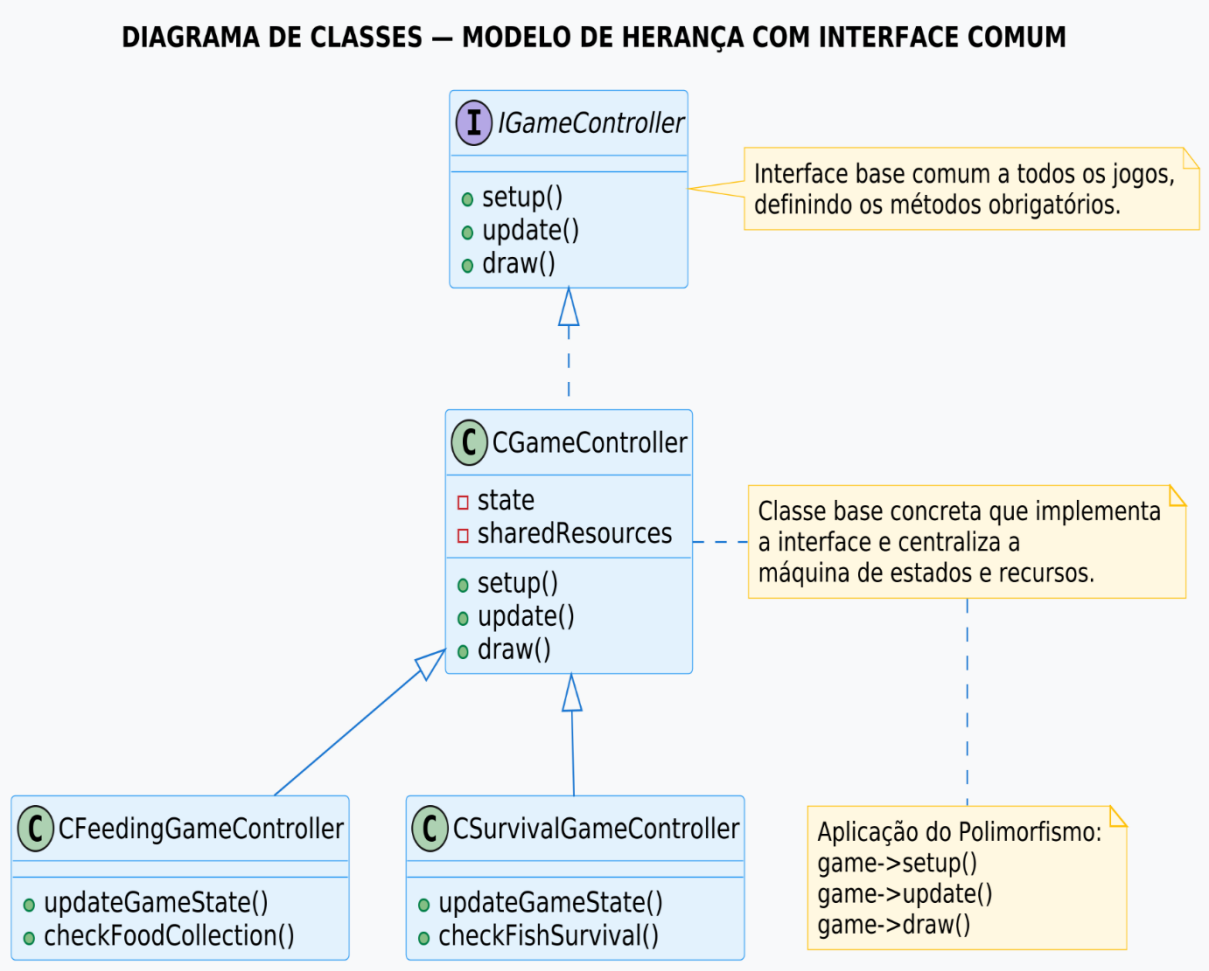


Fonte: Autoria própria

3.3.2 Estrutura de Classes com Interface e Polimorfismo

A estrutura de classes do sistema adota um padrão de herança com polimorfismo, no qual a interface comum *IGameController* atua como um contrato de comunicação entre o sistema e os jogos, definindo os métodos essenciais (*setup(), update() e draw()*) que devem ser implementados por qualquer modo de jogo. A classe *CGameController* implementa essa interface e concentra as funcionalidades base compartilhadas, como a máquina de estados e o gerenciamento de recursos. As classes *CFeedingGameController* e *CSurvivalGameController* especializam a lógica de cada modo de jogo por meio de herança, mantendo a mesma interface para permitir uso polimórfico, de forma que o sistema possa controlar diferentes jogos sem conhecer suas implementações internas. Essa organização facilita a extensão do sistema com novos jogos, preservando consistência estrutural e reutilização de código, conforme apresentado na Figura 3.

Figura 3 – Estrutura de Classes com Interface e Polimorfismo



Fonte: Autoria própria

3.3.3 Atividades Dos Jogos

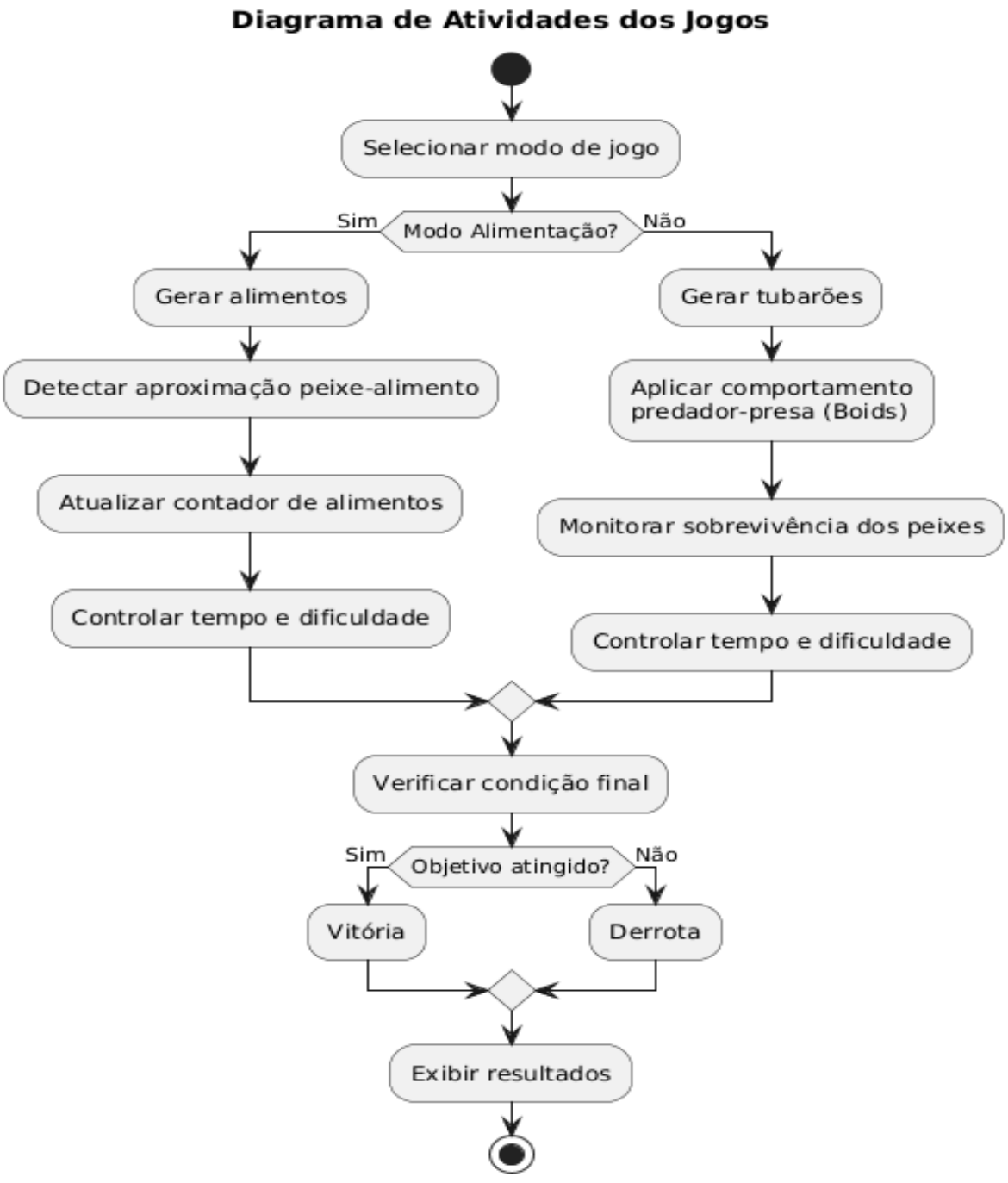
O diagrama de atividades dos jogos descreve o fluxo de execução e os comportamentos associados a cada modo implementado. Após a seleção do modo de jogo, o sistema executa um conjunto de atividades específicas conforme o objetivo definido.

No Jogo da Alimentação, as atividades envolvem a geração de alimentos no ambiente, a verificação da aproximação entre peixe e alimento para possibilitar a coleta e a contagem progressiva dos itens coletados, sob controle de um tempo limite e aumento gradual de dificuldade entre as fases.

No Jogo da Sobrevivência, o fluxo de atividades inclui a geração controlada de tubarões, a simulação do comportamento predador-presa baseada no algoritmo *Boids*, o monitoramento contínuo da sobrevivência dos peixes e a aplicação de um limite de tempo, com dificuldade progressivamente elevada.

A Figura 4 apresenta o fluxo dessas atividades, evidenciando comportamentos distintos para cada modo de jogo a partir de uma estrutura de execução comum.

Figura 4 – Atividades Dos Jogos



Fonte: Autoria própria

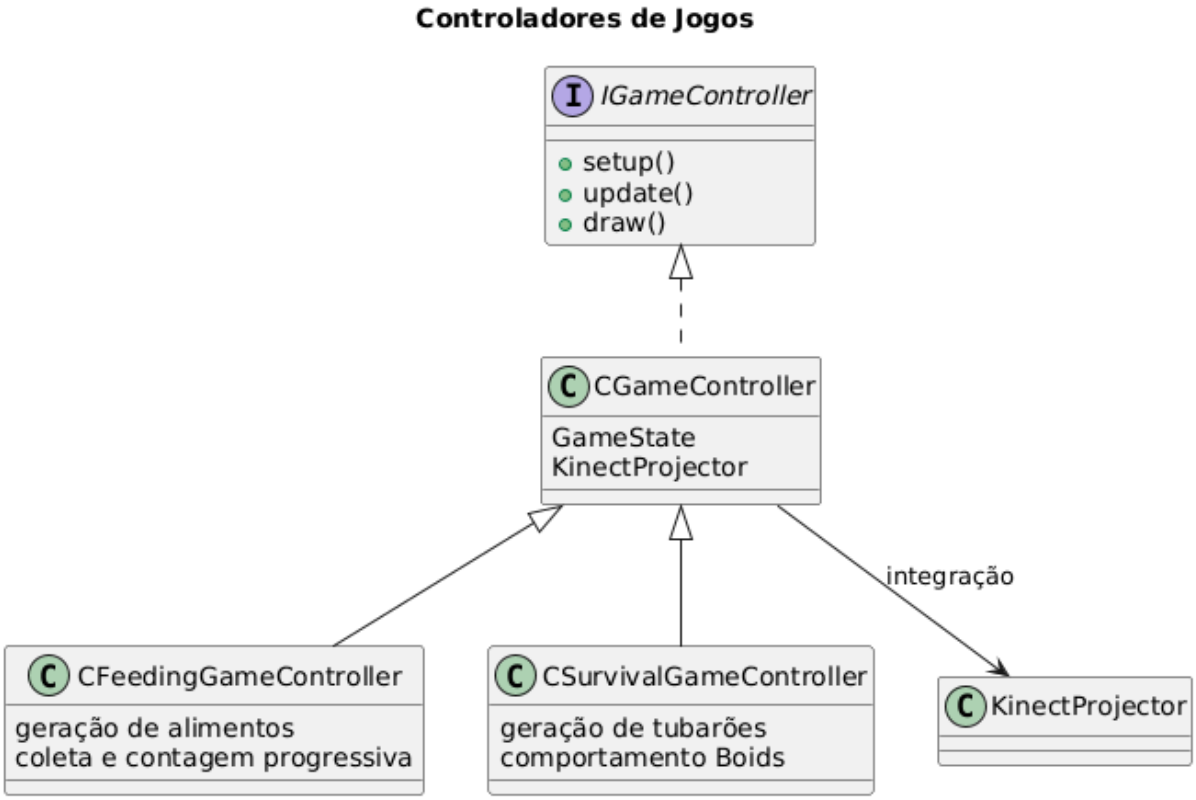
3.3.4 Controladores de Jogos

Os controladores de jogos organiza a lógica da aplicação por meio de uma hierarquia de classes responsável pelo controle do estado do jogo e pela integração com os dispositivos de entrada e saída.

A classe *CGameController* atua como controlador base, gerenciando o estado do jogo (*GameState*), o ciclo de execução e a comunicação com o módulo *KinectProjector*. A partir dessa classe, *CFeedingGameController* e *CSurvivalGameController* especializam comportamentos específicos para cada modo de jogo. O controlador de Alimentação implementa a lógica de coleta de alimentos com detecção de colisão simples e contadores progressivos, enquanto o controlador de Sobrevivência implementa um sistema predador-presa mais complexo, baseado no algoritmo *Boids* e na geração controlada de tubarões.

Essa organização permite o uso de polimorfismo e a reutilização de código comum, mantendo a separação das responsabilidades de cada modo. A Figura 5 apresenta a organização estrutural desses controladores.

Figura 5 - Controladores de Jogos

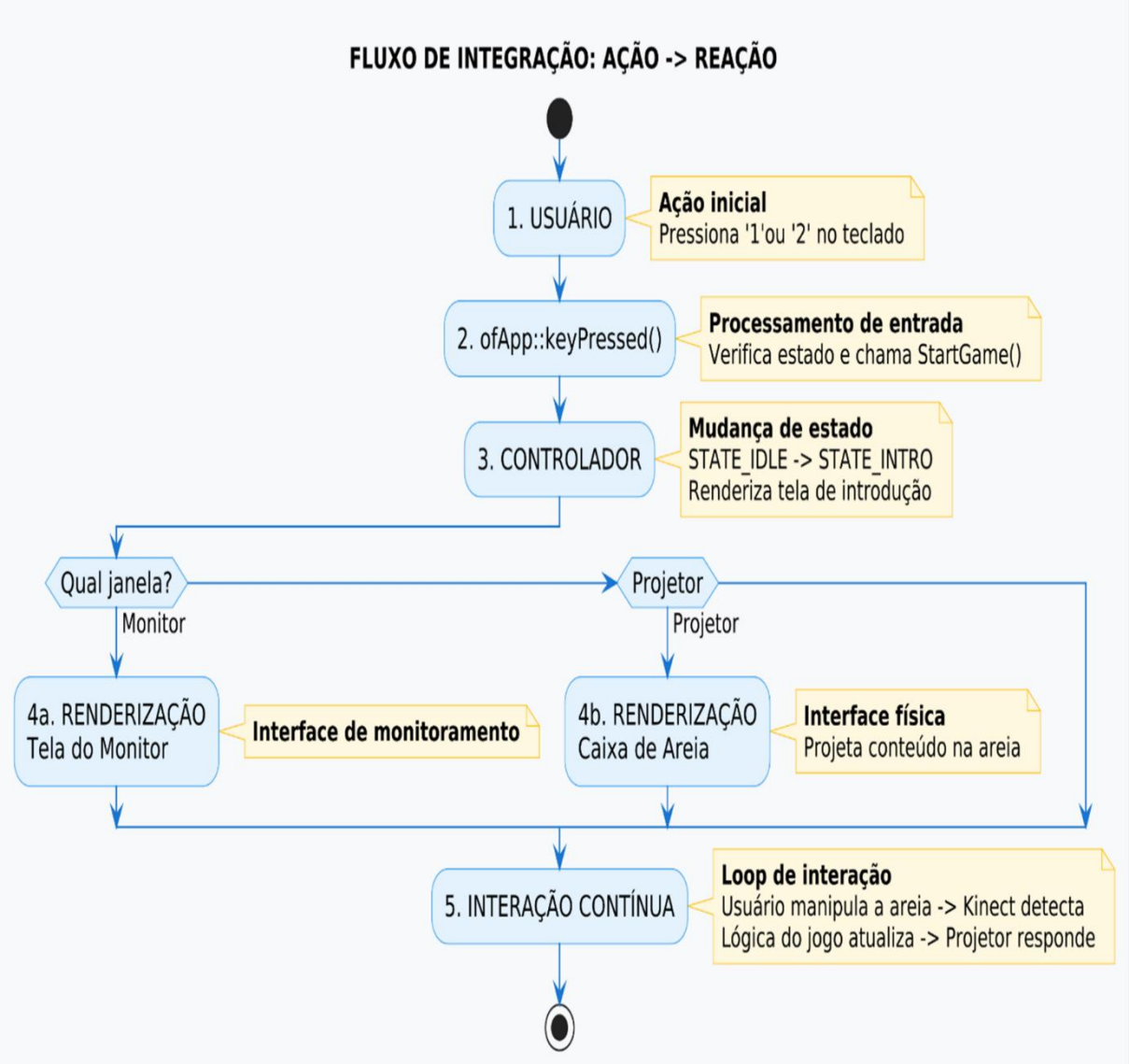


Fonte: Autoria própria

3.3.5 Integração - Ação e Reação

O diagrama de integração mostra o fluxo completo de interação: o usuário pressiona '1' ou '2' para iniciar, *ofApp::keyPressed()* verifica o estado e chama *StartGame()*, o controlador muda para *STATE_INTRO* e renderiza a introdução. O sistema renderiza simultaneamente na tela do monitor (*Main Window*) e na caixa de areia (*Projector Window*). Durante a interação contínua, o usuário modifica a areia, o Kinect detecta as mudanças e atualiza a lógica do jogo em tempo real, com o projetor respondendo imediatamente às ações do usuário. A Figura 6, representa o diagrama de interação do usuário.

Figura 6 - Fluxo de Interação



Fonte: Autoria própria

3.4 Desenvolvimento dos Jogos

O desenvolvimento dos jogos para seguiu arquitetura do software original do 3D SandPlay, seguindo princípios de modularidade e reutilização de código, permitindo a implementação de dois jogos distintos que compartilham a mesma base tecnológica mas possuem mecânicas específicas. Desta forma, foram criados quatro novos arquivos fontes separados e adicionados ao projeto do 3D SandPlay.

Esta seção detalha o desenvolvimento de cada jogo, os sistemas de *steering behaviors* implementados e a estrutura de níveis criada para proporcionar uma progressão desafiadora e educativa.

3.4.1 Jogo da Sobrevivência

O Jogo da Sobrevivência foi implementado como um sistema onde o jogador deve proteger peixes virtuais de tubarões predadores através da manipulação do relevo da areia. A mecânica central envolve a criação de barreiras físicas que os tubarões não conseguem ultrapassar, aproveitando a capacidade do sistema de detectar alterações na topografia da caixa de areia em tempo real.

O desenvolvimento deste jogo incluiu a implementação de componentes essenciais, como geração de predadores, mecanismo de proteção e sistema de pontuação.

Sistema de geração de predadores, controla o aparecimento progressivo de tubarões no ambiente, aumentando a dificuldade conforme o tempo passa. A frequência e localização de aparição dos tubarões são calculadas com base no nível atual e no tempo decorrido, garantindo um desafio crescente mas controlado.

Mecanismo de proteção, permite que os peixes virtuais se escondam atrás das barreiras criadas na areia, com detecção precisa de obstáculos físicos através do processamento do mapa de profundidade.

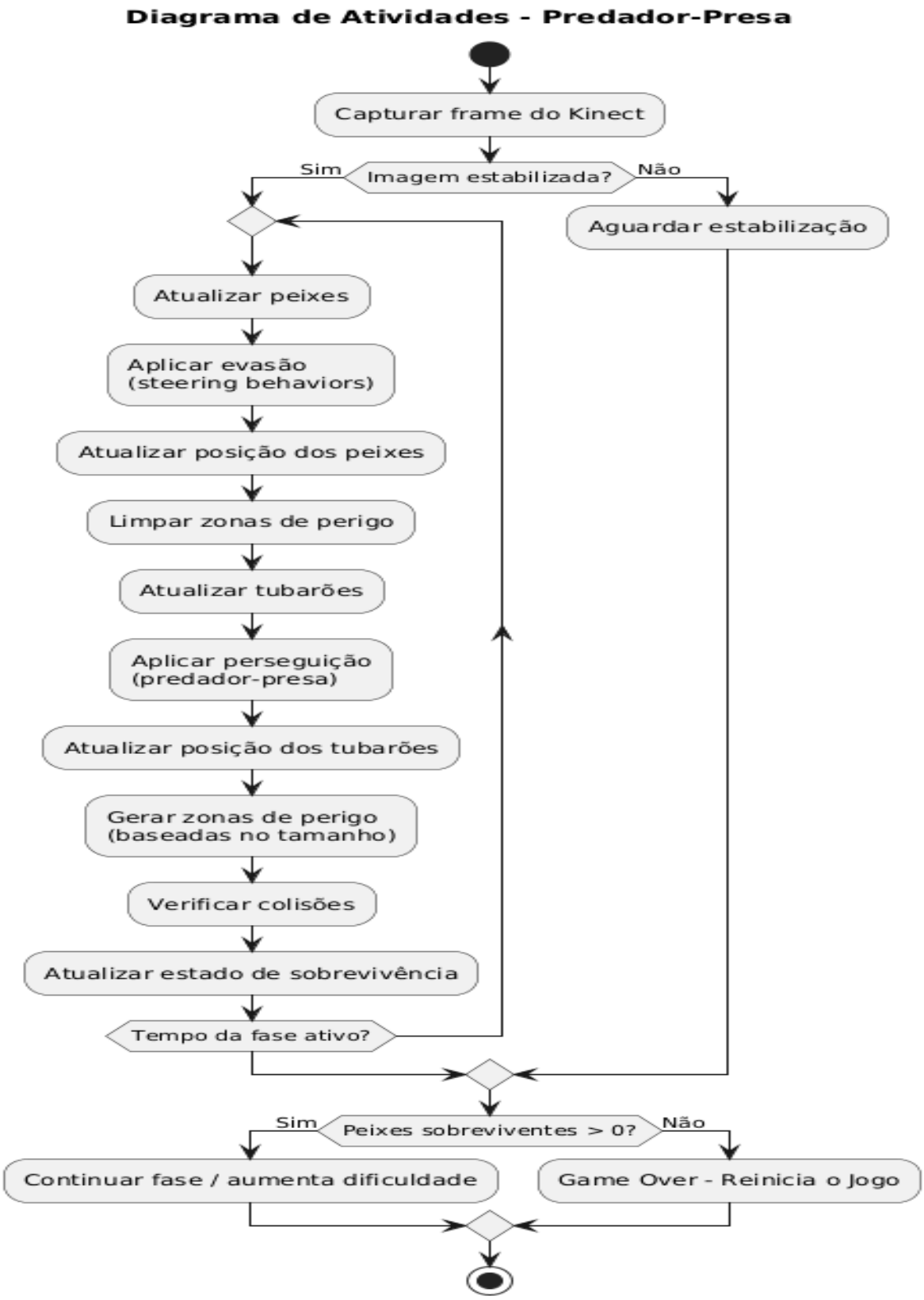
Sistema de pontuação, baseado no tempo de sobrevivência dos peixes e no número de tubarões evitados, com *feedback* visual imediato através da projeção de pontos e progresso na tela da areia.

As implementações referentes ao modo de jogo de sobrevivência, bem como seus respectivos mecanismos, encontram-se disponíveis no repositório GitHub, devidamente referenciado no Apêndice C.

3.4.2 Sistema predador-presa (*steering behaviors*)

O sistema predador-presa foi implementado com base em adaptações dos algoritmos de *steering behaviors* propostos por Reynolds (1999), sendo responsável pela simulação do comportamento dinâmico dos agentes no Jogo da Sobrevivência. Conforme ilustrado na Figura 7, o fluxo de execução inicia com a captura contínua dos dados do Kinect, sendo a atualização dos agentes condicionada à estabilização da imagem. Uma vez atendida essa condição, o sistema entra em um ciclo repetitivo de execução que representa cada quadro do jogo. Nesse ciclo, os peixes têm seu movimento atualizado por meio de comportamentos de evasão, reagindo às zonas de perigo geradas pelos predadores. Em seguida, as zonas de risco são redefinidas e os tubarões são atualizados aplicando comportamentos de perseguição que consideram a posição e a velocidade das presas, resultando na criação de novas áreas de perigo proporcionais ao tamanho dos predadores. Por fim, o sistema verifica a ocorrência de colisões entre tubarões e peixes, atualizando o estado de sobrevivência dos agentes. Esse processo se repete enquanto a fase estiver ativa, sendo encerrado quando o tempo limite é atingido ou não há mais peixes sobreviventes.

Figura 7 - Diagrama de Atividades - Predador-Presa



Fonte: Autoria própria

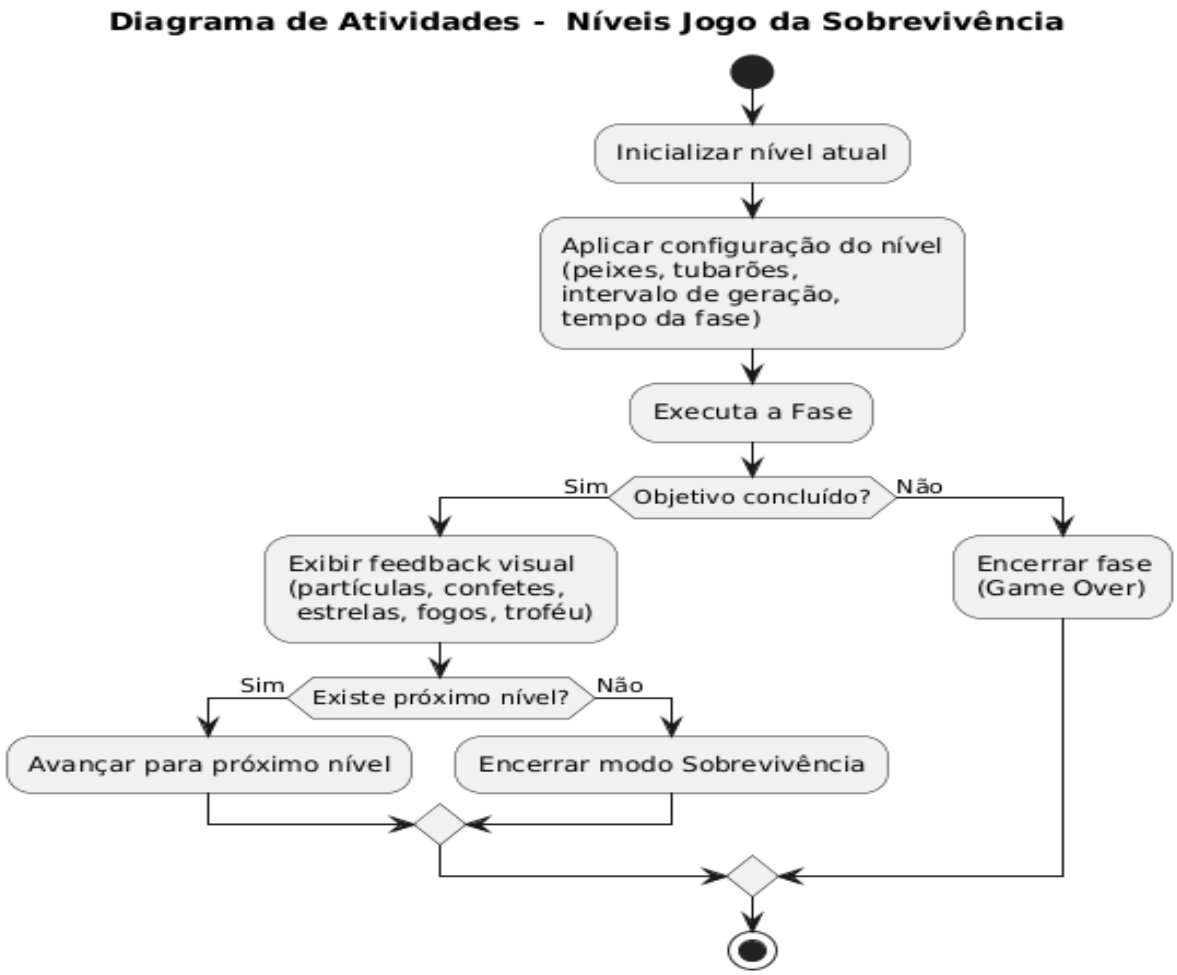
3.4.3 Níveis do Jogo da Sobrevivência

Os níveis do Jogo da Sobrevivência implementam uma progressão gradual de dificuldade composta por três níveis distintos, definidos por parâmetros que controlam a quantidade de agentes, o ritmo de surgimento dos predadores e a duração da fase. Conforme demonstrado na Figura 8, cada nível é configurado por meio de um conjunto específico de valores aplicados automaticamente pelo controlador do jogo.

No Nível 1 considerado introdutório, o sistema utiliza cinco peixes e dois tubarões, com geração de novos tubarões a cada 4,5 segundos e duração total de 45 segundos. No Nível 2 a dificuldade é aumentada com oito peixes e quatro tubarões, reduzindo o intervalo de geração para 3,5 segundos e ampliando a duração da fase para 55 segundos. Já no Nível 3 o nível mais desafiador, o sistema opera com doze peixes e oito tubarões, intervalo de geração de 2,5 segundos e duração total de 65 segundos.

A transição entre os níveis ocorre automaticamente após o cumprimento dos objetivos de cada fase, sem intervenção do jogador. Ao final de cada nível, o sistema apresenta feedback visual positivo por meio de efeitos gráficos, como partículas, confetes, estrelas, fogos de artifício e a exibição de um troféu de conclusão, classificado como bronze, prata ou ouro conforme o nível alcançado.

Figura 8 - Níveis Jogo da Sobrevivência



Fonte: Autoria própria

3.4.4 Jogo da Alimentação

O Jogo da Alimentação foi implementado como um sistema onde o jogador deve criar canais na areia para guiar peixes virtuais até alimentos distribuídos pelo ambiente. A mecânica central envolve a modelagem do relevo da areia para formar caminhos que simulam o fluxo de água, aproveitando a capacidade do sistema de detectar alterações na topografia da caixa de areia em tempo real e calcular o movimento dos peixes conforme as correntes criadas.

O desenvolvimento deste jogo incluiu a implementação de componentes essenciais, como geração de recursos alimentares, mecanismo de fluxo de água e sistema de pontuação.

Sistema de geração de recursos alimentares, controla o aparecimento estratégico de alimentos no ambiente, aumentando a complexidade conforme o nível avança. A distribuição e quantidade de alimentos são calculadas com base no nível atual e no progresso do jogador,

garantindo um desafio equilibrado que incentiva a exploração do território e a criação de rotas eficientes.

Mecanismo de fluxo de água, permite que os peixes sigam as correntes criadas pelos canais na areia, com detecção precisa do relevo através do processamento do mapa de profundidade. Os peixes navegam seguindo o gradiente descendente do terreno, simulando o comportamento natural de busca por alimento em corpos d'água, respondendo dinamicamente às modificações feitas pelo jogador durante a partida.

Sistema de pontuação, baseado no número de alimentos coletados no tempo limite estabelecido para cada nível, com feedback visual imediato através de partículas de água, efeitos de coleta e progresso projetados diretamente na superfície da areia. O sistema inclui bônus por eficiência na criação de rotas e penalidades por alimentos não coletados, incentivando o planejamento estratégico e a otimização dos caminhos criados.

As implementações referentes ao modo de jogo de sobrevivência, bem como seus respectivos mecanismos, encontram-se disponíveis no repositório GitHub, devidamente referenciado no Apêndice C.

3.4.5 Coleta de Alimento (*steering behaviors*)

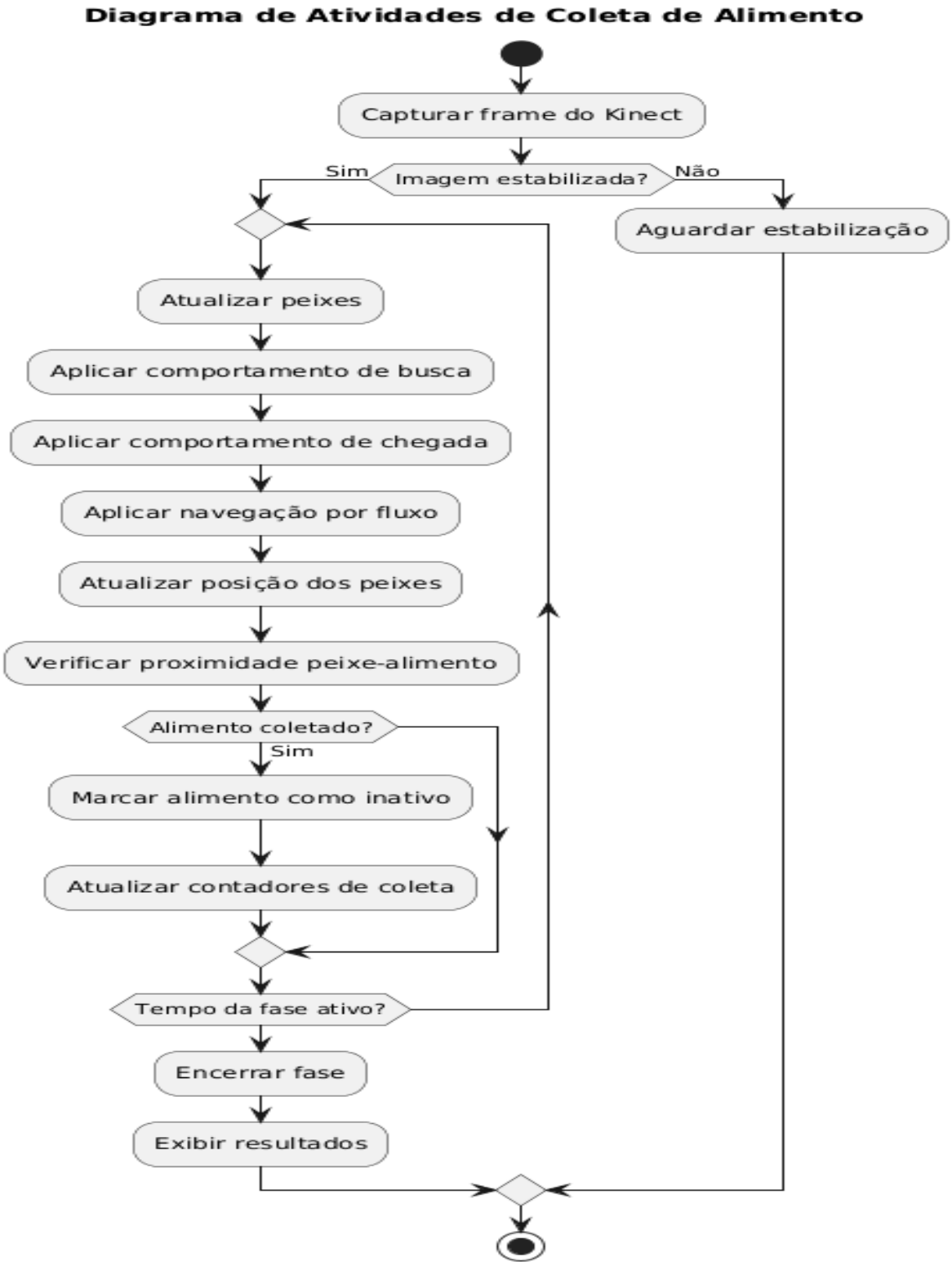
A coleta de alimentos foi implementada a partir de adaptações dos algoritmos de *steering behaviors*, sendo responsável por regular a interação entre os peixes e os recursos alimentares no Modo Alimentação. Esse sistema define como os agentes virtuais respondem às rotas e obstáculos criados pelo usuário no ambiente da caixa areia.

Conforme ilustrado na Figura 9, a cada ciclo de atualização o sistema verifica inicialmente a estabilização da imagem capturada pelo Kinect. Uma vez satisfeita essa condição, os peixes têm seus comportamentos atualizados, aplicando regras de navegação que combinam diferentes estratégias de movimento. Entre os comportamentos utilizados estão o comportamento de busca, no qual os peixes se deslocam em direção aos alimentos mais próximos, aumentando gradualmente a intensidade do movimento à medida que se aproximam do alvo, e o comportamento de chegada, responsável por suavizar a desaceleração dos peixes no momento da aproximação, garantindo uma coleta precisa e visualmente coerente.

Após a atualização do movimento, o sistema verifica a proximidade entre peixes e alimentos, detectando a coleta quando a distância entre eles é inferior a um limite definido. Nesse caso, o alimento é removido do ambiente e os contadores de coleta são atualizados. Esse

processo se repete continuamente enquanto a fase estiver ativa, caracterizando o fluxo apresentado na Figura 9.

Figura 9 – Diagrama de Atividades de Coleta de Alimento



Fonte: Autoria própria

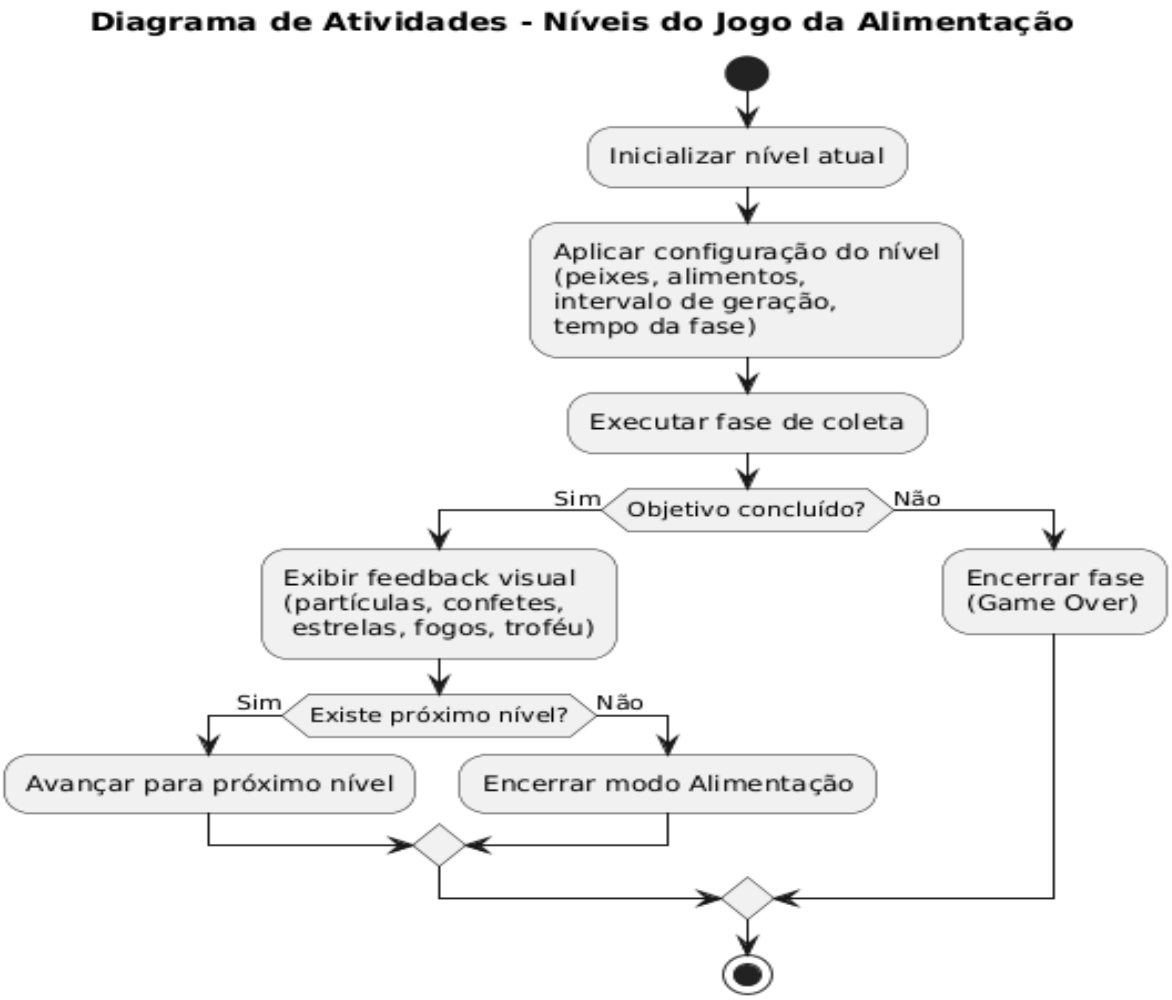
3.4.6 N Jogo da Alimentação

Os níveis do Jogo da Alimentação implementam uma progressão gradual de dificuldade composta por três níveis distintos, definidos por parâmetros que controlam a quantidade de peixes, o número de alimentos disponíveis, o intervalo de geração dos itens e a duração da fase. Esses parâmetros são aplicados automaticamente pelo controlador do jogo, conforme demonstrado na Figura 10.

No Nível 1 considerado introdutório, o sistema utiliza dez peixes e cinco alimentos, com geração de novos alimentos a cada 2,5 segundos e duração total de 60 segundos. No Nível 2 a dificuldade é aumentada com oito peixes e seis alimentos, aumentando o intervalo de geração para 3,5 segundos e diminuindo a duração da fase para 55 segundos. Já no Nível 3 o nível mais desafiador, o sistema opera com seis peixes e oito alimentos, intervalo de geração de 4,5 segundos e duração total de 50 segundos.

A transição entre os níveis ocorre automaticamente após o cumprimento dos objetivos estabelecidos, sem necessidade de intervenção do usuário. Ao final de cada nível, o sistema apresenta feedback visual positivo, incluindo efeitos de partículas, confetes, estrelas, fogos de artifício e a exibição de um troféu de conclusão classificado como bronze, prata ou ouro, de acordo com o nível alcançado. A Figura 10 ilustra o fluxo de execução e progressão desse sistema.

Figura 10 - Níveis do Jogo da Alimentação



Fonte: Autoria própria

4 TESTES E AVALIAÇÃO DE CENÁRIOS

4.1 Configuração do Ambiente de Testes

Os testes foram realizados em um ambiente controlado, com ambientação adequada, com luminosidade mais baixa para garantir a consistência dos resultados. O *hardware* foi configurado conforme as especificações descritas no capítulo anterior, com o Kinect e o projetor montados em suportes estáveis para evitar vibrações que poderiam comprometer as projeções.

A caixa de areia utilizada nos testes possuía dimensões de 45x33x7cm e foi preenchida com areia branca de aquário para garantir boa reflexão da luz e detecção precisa pelo sensor Kinect. A iluminação do ambiente foi controlada para minimizar interferências externas, principalmente durante testes de sensibilidade à luz ambiente.

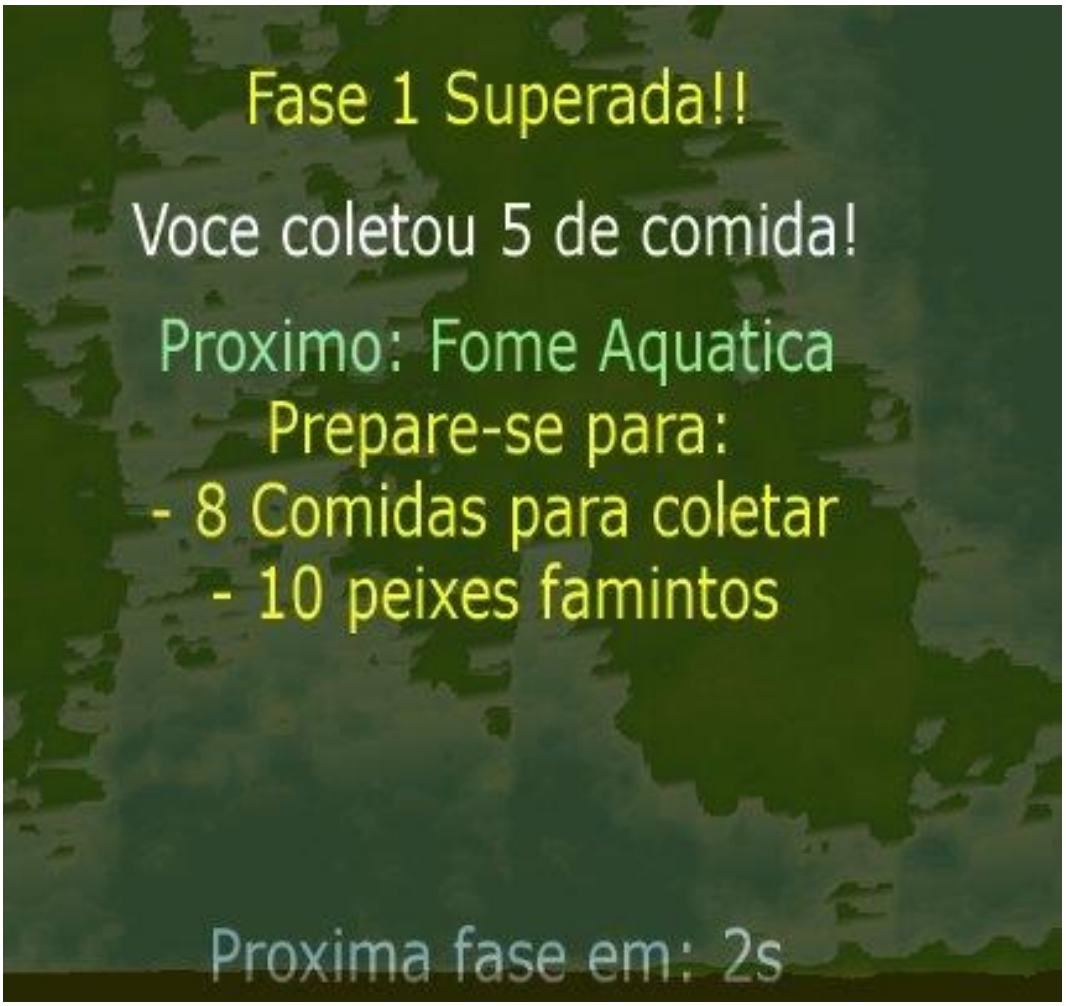
4.2 Procedimento dos Testes

O procedimento de testes seguiu uma estrutura sistemática com cinco etapas principais: calibração inicial para verificação do alinhamento entre o sensor Kinect e o projetor através da interface de calibração automática do sistema; testes funcionais para verificação individual de cada componente do sistema, incluindo detecção de profundidade, projeção mapeada e funcionamento dos algoritmos de comportamento de agentes; e testes integrados para avaliação da interação entre todos os componentes em cenários reais de uso, simulando a interação do usuário com os jogos. Cada fase de teste foi documentada com capturas de tela e vídeos para análise e avaliação dos resultados obtidos. Todo esse procedimento se encontra no Apêndice D.

4.3 Ajustes Finais e Refinamento do Protótipo

Durante o processo de testes, foram identificadas oportunidades de refinamento do protótipo inicial, com ajustes focados em melhorar a experiência do usuário através de melhorias na interface visual. Estas melhorias incluíram o refinamento do retorno visual com a adição de elementos como troféus, estrelas, fogos de artifícios e confetes nas telas de vitória para aumentar a clareza do sucesso nas fases, a otimização de cores e contrastes para melhorar a visibilidade dos elementos na projeção especialmente em condições de iluminação variável, e a implementação de transições suaves entre estados com efeitos visuais que proporcionaram maior fluidez na experiência do usuário durante a interação com o sistema. A Figura 11 apresenta o protótipo inicial de conclusão da primeira fase do Jogo da Alimentação.

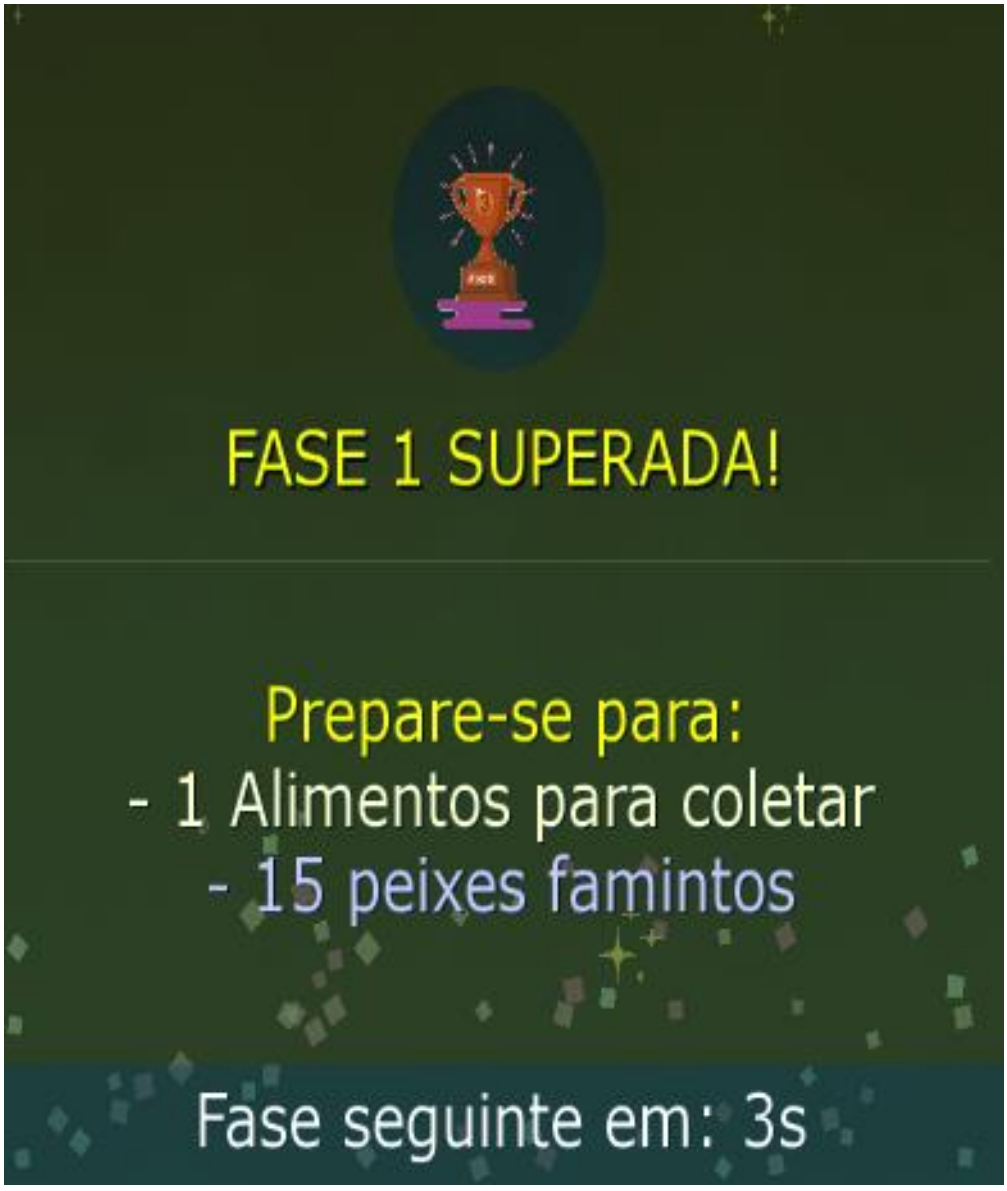
Figura 11 - Protótipo Fase 1 – Jogo da Alimentação



Fonte: Autoria própria

A partir das análises realizadas durante os testes, foi conduzido um processo de refinamento do protótipo com foco na melhoria da experiência visual do usuário. Esse processo incluiu a reestruturação da interface e a incorporação de novos elementos visuais, como troféus, estrelas, e confetes, visando aprimorar a clareza na apresentação dos resultados. A Figura 12 apresenta o refinamento do protótipo da fase 1.

Figura 12 - Refinamento Fase 1 – Jogo da Alimentação



Fonte: Autoria própria

4.4 Avaliação dos Resultados

A avaliação dos resultados foi realizada por meio de testes funcionais e observacionais, considerando o comportamento dos jogos, a resposta dos sistemas interativos e as limitações técnicas identificadas durante o uso do protótipo. Em ambos os jogos, observou-se o funcionamento adequado das funcionalidades implementadas, com progressão gradual de dificuldade ao longo dos níveis e a utilização de efeitos visuais de partículas como mecanismo de feedback, contribuindo para uma comunicação clara e intuitiva das ações e dos resultados ao usuário. No Jogo da Sobrevivência, verificaram-se a geração progressiva de tubarões, a resposta correta dos algoritmos de evasão dos peixes às barreiras criadas na areia e a atualização do sistema de pontuação em tempo real. No Jogo da Alimentação, a geração procedural de alimentos apresentou distribuição coerente no ambiente e o sistema de detecção de colisão funcionou com precisão. Apesar dos resultados positivos, foram identificadas limitações relacionadas à sensibilidade do sistema à iluminação do ambiente, especialmente sob incidência direta de luz solar e iluminação da sala onde foram realizados os testes.

5 CONCLUSÕES

5.1 Considerações finais

Este trabalho teve como objetivo a ampliação das funcionalidades do sistema de interface tangível 3D SandPlay por meio da implementação de algoritmos de *steering behaviors*, adaptados para um ambiente interativo no qual o terreno físico é continuamente modificado pelo usuário. A pesquisa demonstrou a viabilidade técnica da aplicação dos modelos propostos por Reynolds (1999) em um contexto de interação tangível, exigindo o processamento em tempo real das variações topográficas capturadas pelo sensor Kinect.

Ao longo do desenvolvimento, o principal desafio enfrentado consistiu na reutilização de um ecossistema de software legado, o que demandou uma abordagem cuidadosa de engenharia reversa para preservação da integridade do sistema original. A manutenção da compatibilidade com versões específicas de ferramentas (Visual Studio 2015, *openFrameworks* 0.9.8 e Kinect SDK v1.0) e bibliotecas (*ofxKinect*, *libfreenect*), uma vez que essa abordagem possibilitou a incorporação dos algoritmos de *steering behaviors* de busca, evasão e desvio de obstáculos, a criação de dois novos modos de jogo, Sobrevivência e Alimentação e a adição de sistemas de geração procedural de agentes e alimentos, pontuação em tempo real e *feedback* visual, sem comprometer o funcionamento do sistema preexistente.

Como resultado, foi desenvolvido um protótipo funcional no qual agentes virtuais demonstraram comportamentos coerentes e responsivos às alterações físicas realizadas pelo usuário, confirmando que sistemas de projeção mapeada existentes podem ser estendidos predominantemente por meio de *software*, sem a necessidade de substituição integral do *hardware*.

Do ponto de vista prático, a relevância deste trabalho manifesta-se em três dimensões fundamentais. Na dimensão educacional, os jogos desenvolvidos evidenciam o potencial das interfaces tangíveis para a mediação de conceitos abstratos, como fluxo hidrológico, dinâmica de populações e princípios ecológicos, por meio de interações físicas intuitivas que estabelecem uma ponte cognitiva entre o mundo tangível e o abstrato. Na dimensão terapêutica, o sistema oferece um ambiente de estímulo multissensorial capaz de contribuir para o desenvolvimento de habilidades motoras finas e cognitivas em populações com necessidades especiais, em especial pessoas com Síndrome de Down, público-alvo original do 3D SandPlay. Por fim, na dimensão tecnológica, este trabalho estabelece um *framework* de extensibilidade que demonstra

como sistemas de projeção mapeada existentes podem ser evoluídos por meio de intervenções predominantemente em *software*, sem a necessidade de substituição completa do *hardware*, configurando um modelo de desenvolvimento sustentável e replicável para instituições com recursos limitados.

5.2 Sugestões para Trabalhos Futuros

Com base nos resultados obtidos neste trabalho e nas limitações identificadas, propõem-se duas linhas de pesquisa prioritárias para desenvolvimentos futuros:

- Modos Multijogador Cooperativos

Implementação de modos multijogador cooperativos, permitindo interação entre múltiplos usuários. Isto possibilitaria avaliar a interação social entre diferentes usuários para a consecução das tarefas. Tecnicamente, esta implementação exigiria uma reestruturação significativa da arquitetura de software para suportar múltiplas instâncias de agentes virtuais com comportamentos coordenados, compartilhamento de estado do terreno em tempo real e sincronização de eventos entre diferentes pontos de interação.

- Realização de estudos empíricos com usuários

Realização de estudos empíricos com usuários reais para validar os benefícios do sistema através de testes controlados com populações específicas. Esta validação possibilitaria transformar o protótipo em uma ferramenta reconhecidamente eficaz para contextos educacionais e terapêuticos. Tecnicamente, exigiria o desenvolvimento de sistemas de avaliação quantificáveis com métricas objetivas de progresso, mecanismos de adaptação automática de dificuldade e dashboards de monitoramento para análise longitudinal do desenvolvimento de habilidades cognitivas e motoras.

- Implementação de um sistema de vidas ou ponto de salvamento de progresso (checkpoint)

Essa implementação permiti que o usuário tenha um limite de tentativas por nível e pontos de salvamento estratégicos, possibilitando reiniciar o nível atual ou retornar ao anterior, reduzindo frustração e tornando a experiência de jogo mais equilibrada e adaptativa.

6 REFERÊNCIAS

AEKO CONSULTING. ***Zadig: USB driver installation utility***. 2025. Disponível em: https://zadig.akeo.ie/. Acesso em: 6 dez. 2025.

ALVES, Ricardo. ***TCC-Eng-Computação:*** *Código-fonte dos Jogos*. Disponível em: https://github.com/GlT-Ricardo/TCC-Eng-Computa-o. Acesso em: 18 dez. 2025.

DETERDING, S. et al. ***Gamification: toward a definition***. In: Chi 2011 Gamification Workshop Proceedings, 2011.

DOURADO, Gabriel dos Santos et al. ***An ARSandplay System for People with Down Syndrome***. In: 2019 Ieee Mit Undergraduate Research Technology Conference (URTC), Cambridge, MA, EUA, 11-13 out. 2019. Anais [...] Piscataway: IEEE, 2019. p. 1-6. DOI: https://ieeexplore.ieee.org/document/9660441. Disponível em: https://ieeexplore.ieee.org/document/9660441. Acesso em: 15 jun. 2025.

HIROSHI ISHII AND BRYGG ULLMER. 1997. ***Tangible bits: towards seamless interfaces between people, bits and atoms***. In Proceedings of the ACM SIGCHI Conference on Human factors in computing systems (CHI '97). Association for Computing Machinery, New York, NY, USA, 234–241. https://doi.org/10.1145/258549.258715. Disponível em: https://dl.acm.org/doi/10.1145/258549.258715. Acesso em: 15 jun. 2025

JOSEFOBERDAN. ***Projeto 3D SandPlay: especificações técnicas e implementação***. 2022. Disponível em: https://github.com/josefoberdan/Projeto3DSandPlay. Acesso em: 6 dez. 2025.

KOGAN, Gene; WOLF, Thomas; PAULSEN, Rasmus R. ***Magic Sand***. Califórnia, 2018. Disponível em: https://github.com/thomwolf/Magic-Sand. Acesso em: 28 nov. 2025.

KREYLOS, Oliver. Argumented Reality Sandbox (AR Sandbox). Califórnia: University of Californica, Davis, 2014. Disponível em: https://web.cs.ucdavis.edu/~okreylos/ResDev/SARndbox/. Acesso em: 6 dez. 2025

LIMA, D. et al. ***Software with Biofeedback to Assist People with Down Syndrome***. International Journal of Computer Applications, v. 158, n. 5, p. 31-39, 2017. DOI: https://doi.org/10.5120/ijca2017912704.

OPENFRAMEWORKS. ***openFrameworks: creative coding toolkit.*** 2025. Disponível em: https://openframeworks.cc/. Acesso em: 6 dez. 2025.

REYNOLDS, Craig W. **S*teering behaviors for autonomous characters***. In: Game Developers Conference, 1999. p. 763-781. Disponível em: https://www.red3d.com/cwr/papers/1999/gdc99steer.pdf. Acesso em: 6 dez. 2025.

ROSSIT, Rosana Aparecida Salvador. ***Matemática para deficientes mentais: contribuições do paradigma de equivalência de estímulos para o desenvolvimento e avaliação de um currículo.*** 2003. 169 f. Tese (Doutorado em Educação Especial) – Universidade Federal de São Carlos, São Carlos, 2003.

SCIENCE DIRECT. ***Interfaces de usuários tangíveis: definição e conceitos fundamentais***. Disponível em: https://www.sciencedirect.com/topics/computer-science/tangible-user-interfaces. Acesso em: 6 dez. 2025.

SHAER, Orit; HORNECKER, Eva. ***Tangible user interfaces: past, present and future directions.*** Foundations and Trends® in Human–Computer Interaction, v. 3, n. 1-2, p. 1-137, 2010.

WIKIPEDIA. ***Interface tangível de usuário***. 2025. Disponível em: https://pt.wikipedia.org/wiki/Interface_tangível_de_usuário. Acesso em: 6 dez. 2025.

APÊNDICE A – CONFIGURAÇÃO DO HARDWARE

Nesta seção são apresentadas as configurações de hardware descritos no Capítulo 3 Sessão 3.2.3 Configuração do Hardware.

O sensor Kinect e o projetor foram conectados ao computador através de cabos USB e HDMI/VGA, respectivamente. Esta etapa é fundamental para garantir que todos os componentes estejam fisicamente conectados antes da configuração de software.

- Instalação dos drivers do Kinect via Zadig:

1. Download do Zadig: https://zadig.akeo.ie/
2. Execução do aplicativo com privilégios de administrador
3. Seleção de cada componente do Kinect (Xbox Camera, Xbox Audio, Xbox Motor)
4. Escolha do driver libusb-win32 (v1.2.6.0) para cada componente
5. Instalação repetida para cada componente até confirmação de sucesso.

- Instalação do Kinect V1 SDK 1.0:

Download SDK: https://www.microsoft.com/en-us/download/details.aspx?id=28782

Para instalar o SDK:

Certifique-se de que o sensor Kinect não esteja conectado a nenhuma das portas USB do computador.

Se uma versão anterior do SDK do Microsoft Kinect para Windows estiver instalada, você deverá desinstalá-la antes de prosseguir.

Remova quaisquer outros drivers para o sensor Kinect.

Desinstale todos os componentes do *Microsoft Server Speech Platform Runtime* e SDK, incluindo as versões de 86 e 64 bits, além do *Microsoft Server Speech Recognition Language - Kinect Language Pack*.

Caso o Visual Studio estiver aberto, ele deve ser fechado o antes de instalar o SDK e reiniciá-lo após a instalação para que as variáveis de ambiente necessárias para o SDK sejam reconhecidas.

Na pasta onde você baixou o arquivo, clique duas vezes em KinectSDK-v1.0-Setup.exe. Este instalador único funciona tanto em versões de 32 bits quanto de 64 bits do Windows.

Após a instalação bem-sucedida do SDK, certifique-se de que o sensor Kinect esteja conectado a uma fonte de alimentação externa e, em seguida, conecte-o à porta USB do computador. Os drivers serão carregados automaticamente.

APÊNDICE B – RESULTADOS DAS FASES

Imagem B1 – Fase 1 – Nível Fácil – Jogo da Sobrevivência

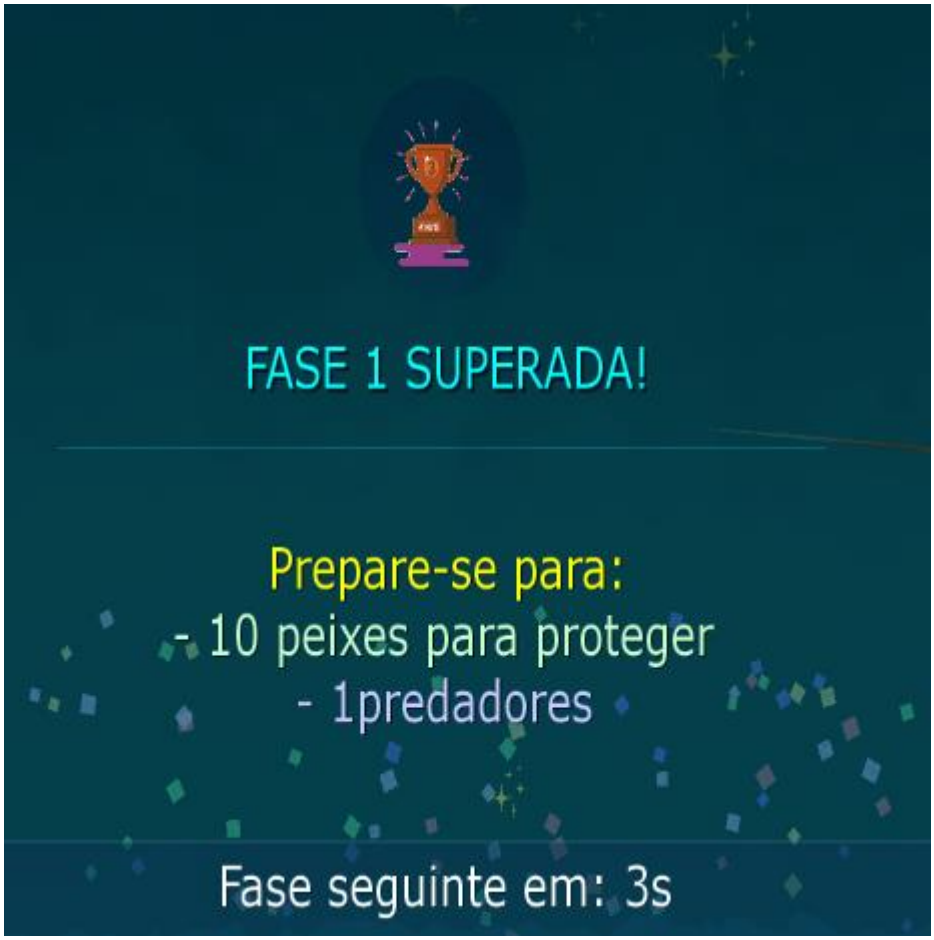


Imagem B2 – Fase 2 – Nível Médio – Jogo da Sobrevivência

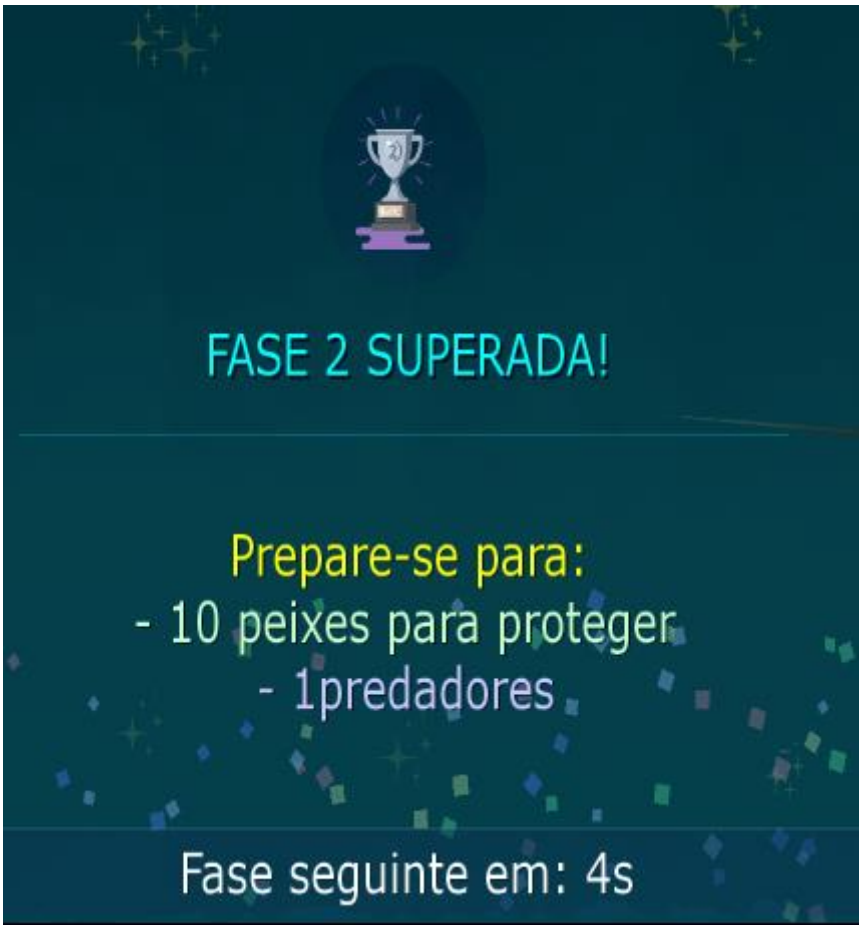


Imagem B3 – Fase 3 – Nível Difícil – Jogo da Sobrevivência

Imagem B4 – Game Over – Jogo da Sobrevivência

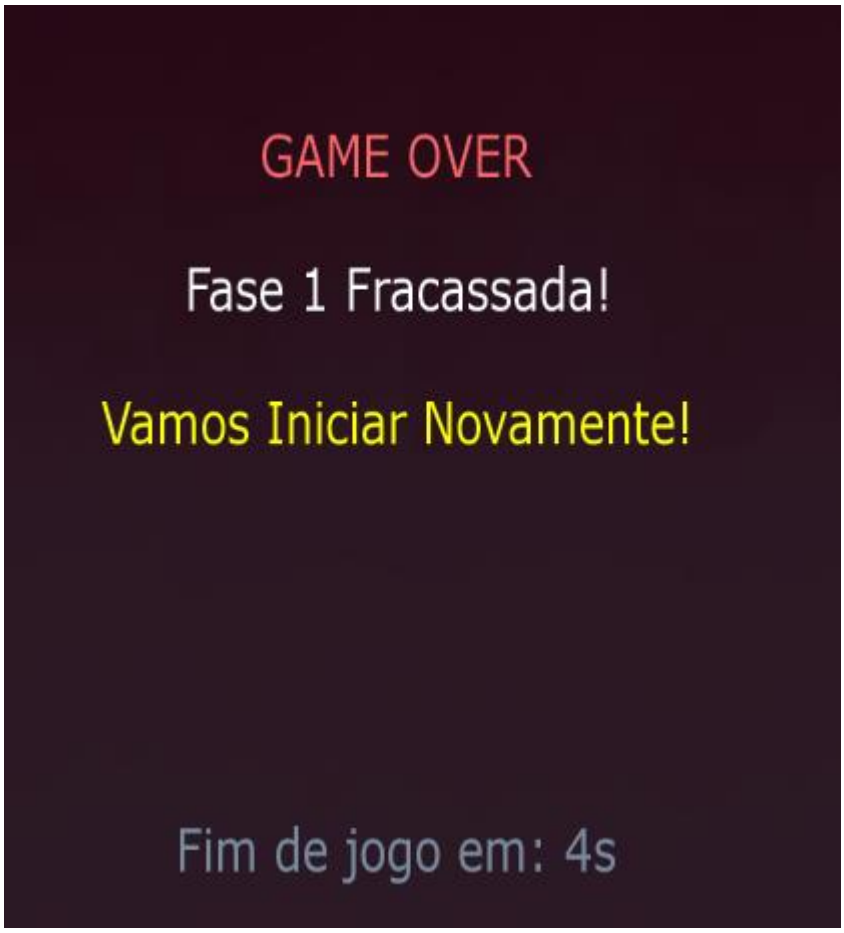


Imagem B5 – Fase 1 – Nível Fácil – Jogo da Alimentação

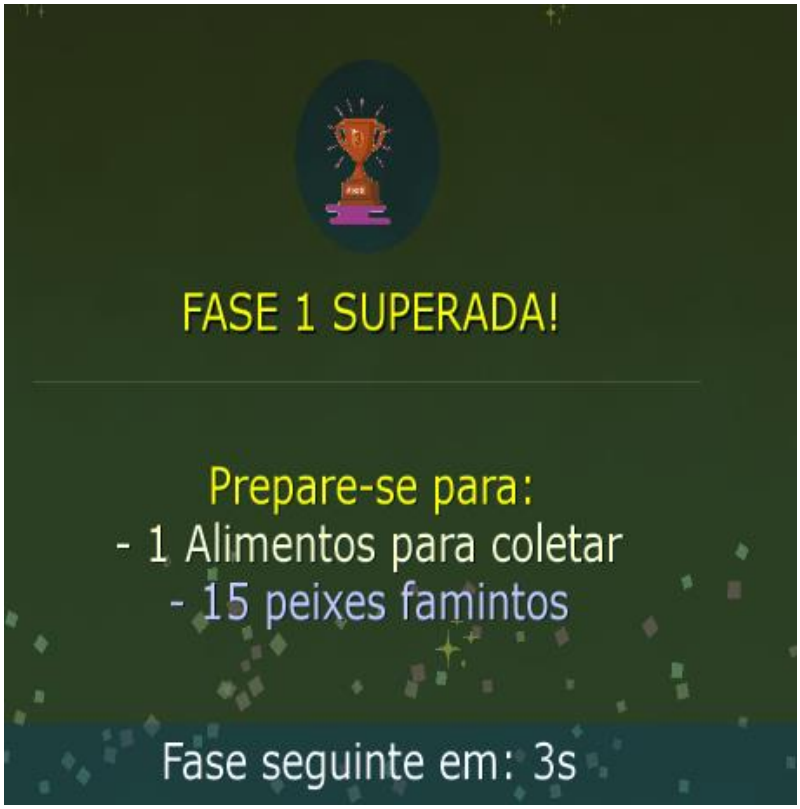


Imagem B6 – Fase 2 – Nível Médio – Jogo da Alimentação

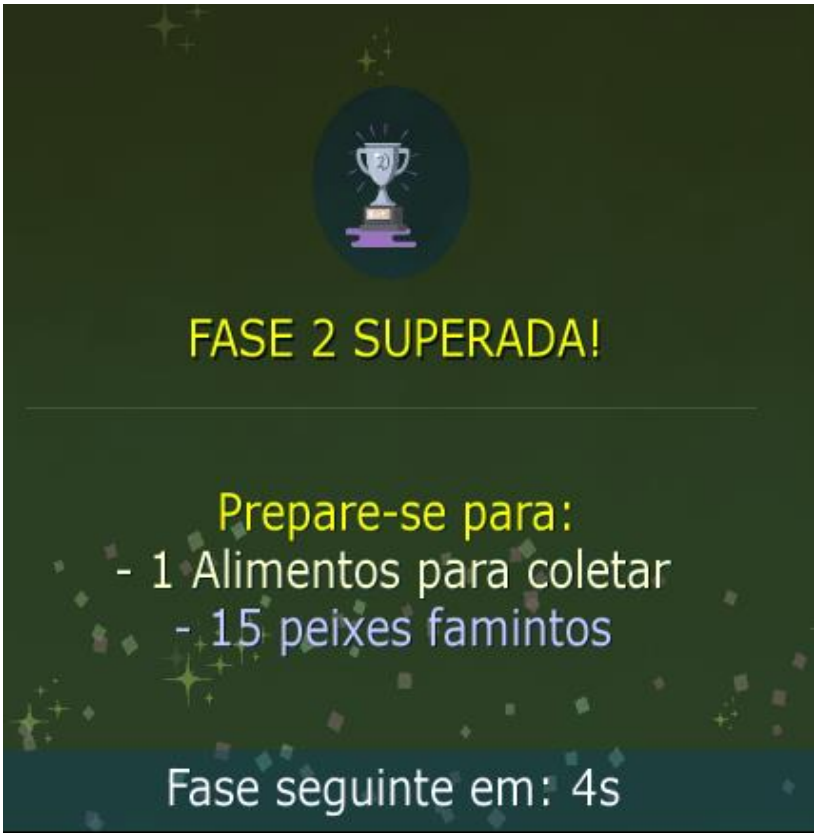
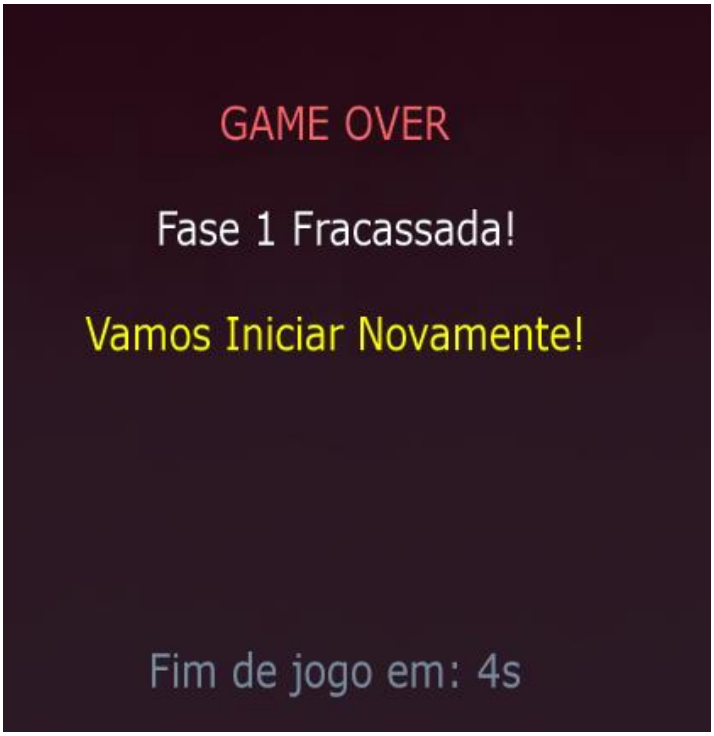


Imagem B7 – Fase 3 – Nível Difícil – Jogo da Alimentação

Imagem B7 – Game Over – Jogo da Alimentação



APÊNDICE C – REPOSITÓRIO DOS JOGOS

O código-fonte completo desenvolvido para este trabalho, assim como toda documentação, está disponível publicamente no repositório GitHub da autora, no seguinte endereço:

https://github.com/GlT-Ricardo/TCC-Eng-Computa-o

ANEXO A – MANUAL DE COMPONENTES

Nesta seção são apresentados todos os manuais dos componentes utilizados no desenvolvimento do protótipo, organizados de acordo com os itens descritos no Capítulo 3 Sessão 3.2.1 Dependências de Versões de Software

- Visual Studio 2015:

MICROSOFT. Pacotes Redistribuíveis do Visual C++ para Visual Studio 2015. Disponível em: https://www.microsoft.com/pt-br/download/details.aspx?id=48145. Acesso em: 17 dez. 2025.

STEFANETTI, Roberto. Visual Studio Community 2015 (Free Visual Studio 2015 with Windows Developer Tools). Dynamics 365 Community Blogs. Disponível em: https://community.dynamics.com/blogs/post/?postid=6cd3fadf-f831-429c-a16e-9f5cfc5790e3. Acesso em: 17 dez. 2025.

- openFrameworks:

OPENFRAMEWORKS. About openFrameworks. Disponível em: https://openframeworks.cc/about/. Acesso em: 18 dez. 2025.

- Kinect SDK v1.0

MICROSOFT. Kinect for Windows SDK v1.0. Disponível em: https://www.microsoft.com/en-us/download/details.aspx?id=28782. Acesso em: 18 dez. 2025.

- Zadig

AEKO CONSULTING. Zadig: USB driver installation utility. 2025. Disponível em: https://zadig.akeo.ie/. Acesso em: 18 dez. 2025.