Universidade Federal Rural do Rio de Janeiro

Curso Ciência da Computação

Nova Iguaçu

01/12/2024

Mariana Ferreira Ferrarez Ribeiro Mat.: 20230001113

Pedro H. Guedes Mat.: 20230000279

Jogo de Física

Disciplina: TM407 - FÍSICA PARA CIÊNCIA DA COMPUTAÇÃO

Professor: Frederico Alan de Oliveira.

Desenvolvemos este jogo com fins educativos para a matéria de Física 2024.2. Durante o projeto, proposto pelo nosso professor Frederico Alan de Oliveira, buscamos aplicar conceitos importantes de resistores e forças eletromagnéticas de forma interativa. Foi uma experiência de grande aprendizado, onde pudemos consolidar nossos conhecimentos e habilidades em programação, além de contribuir para o ensino de Física de maneira lúdica. Como alunos de Ciência da Computação, esse projeto nos permitiu integrar teoria e prática de forma significativa.

Classe Sala

A classe Sala é responsável pela renderização da sala do jogo e por gerenciar os desafios disponíveis. Ela controla a interface gráfica, a ativação dos desafios e a interação do jogador.

Atributos:

- PImage background: Imagem de fundo principal da sala.
- PImage[] backgroundDesafios: Array de imagens de fundo para cada desafio.
- Desafio[] desafios: Array contendo os desafios disponíveis na sala.
- PVector[] pontosDeAtivacao: Array de vetores que indicam os pontos na tela onde os desafios podem ser ativados. Cada PVector contém as coordenadas (x, y) e a tolerância de clique (z).
- String[] perguntas: Array de strings que contém as dicas (perguntas) para ajudar o jogador a encontrar os desafios.
- int desafioAtual: Índice do desafio que está ativo ou que será o próximo a ser ativado.
- boolean[] desafiosIniciados: Array que indica se cada desafio foi ou não iniciado.
- boolean fundoDesafioAtivo: Flag que indica se o fundo especial de um desafio está ativo ou não.

Construtor:

Sala(PImage background, PImage[] backgroundDesafios, Desafio[] desafios, PVector[] pontosDeAtivacao, String[] perguntas)

- Inicializa a sala com um fundo principal, fundos específicos para os desafios, os desafios, os pontos de ativação e as perguntas (dicas).
- Também inicializa o array desafiosIniciados para marcar o progresso dos desafios.

Métodos:

- void render(): Responsável por desenhar o fundo da sala e, se um desafio estiver ativo, desenhar o fundo do desafio. Caso contrário, exibe a dica para o próximo desafio. Se o desafio estiver em andamento, chama o método render() do desafio atual.

- void drawTextBox(String texto, float x, float y, float w, float h): Desenha uma caixa com texto em um determinado local da tela. Utilizada para exibir dicas e mensagens.
- boolean pontoDeAtivacaoClicado(float x, float y, PVector ponto): Verifica se um ponto de ativação foi clicado com base nas coordenadas do clique e na tolerância definida no vetor ponto.
- boolean todosDesafiosCompletos(): Verifica se todos os desafios da sala foram concluídos, retornando true se todos estiverem completos.
- void handleMousePressed(float x, float y): Trata os eventos de clique do mouse. Se um ponto de ativação é clicado, ativa o desafio correspondente. Também encaminha o clique para o método mousePressed() do desafio atual, caso ele já esteja iniciado.
- void handleKeyPressed(char key): Trata os eventos de teclas pressionadas. Se o desafio atual está em andamento, chama o método keyPressed() do desafio correspondente. Caso o desafio seja completado via interação com o teclado, avança para o próximo desafio, se houver.
- int getDesafioAtual(): Retorna o índice do desafio atual.

Classe Desafio

Classe abstrata que define a estrutura básica para um desafio. Cada desafio deve implementar os métodos para renderizar, processar cliques e interagir com o teclado.

Atributos:

- boolean completo: Indica se o desafio foi completado.

Métodos abstratos:

- abstract void render(): Método responsável por desenhar o desafio na tela.
- abstract void mousePressed(): Método para processar cliques do mouse no contexto do desafio.
- abstract void keyPressed(char key): Método para processar interações com o teclado.

Métodos concretos:

- boolean estaCompleto(): Retorna true se o desafio foi completado.

Classe ControladorDeFase

Controla a fase atual do jogo, permitindo transições entre fases e a renderização adequada da fase em andamento.

Atributos:

- Fase faseAtual: A fase que está ativa no momento.

Construtor:

ControladorDeFase(Fase faseInicial)

- Inicializa o controlador com uma fase inicial.

Métodos:

- void setFase(Fase novaFase): Altera a fase atual do jogo.
- void render(): Chama o método render() da fase atual.
- void handleMousePressed(float x, float y): Encaminha o evento de clique do mouse para a fase atual.
- void handleKeyPressed(char key): Encaminha o evento de tecla pressionada para a fase atual.

Classe Fase

Classe abstrata que define a estrutura básica de uma fase. Cada fase deve implementar a renderização e o tratamento de eventos.

Atributos:

- ControladorDeFase controlador: Controlador responsável por gerenciar a fase.

Construtor:

Fase(ControladorDeFase controlador)

- Inicializa a fase com um controlador.

Métodos abstratos:

- abstract void render(): Desenha a fase na tela.
- abstract void handleMousePressed(float x, float y): Processa cliques do mouse na fase.
- abstract void handleKeyPressed(char key): Processa teclas pressionadas na fase.

Classe GerenciadorDeSala

Gerencia as salas do jogo e as transições entre elas.

Atributos:

- ArrayList<Sala> salas: Lista de salas no jogo.
- int salaAtual: Índice da sala atual.

Métodos:

- void addSala(Sala sala): Adiciona uma sala à lista de salas.
- Sala getSalaAtual(): Retorna a sala atual.
- void render(): Renderiza a sala atual. Se todos os desafios da sala forem completados, exibe uma mensagem de transição para a próxima sala.
- void handleMousePressed(float x, float y): Encaminha o evento de clique para a sala atual.
- void handleKeyPressed(char key): Processa teclas pressionadas na sala atual e, se necessário, avança para a próxima sala.

Classe DesafioDaVerdade

Este desafio apresenta uma questão relacionada ao conceito de refração da luz, onde o jogador deve inserir o ângulo de refração correto para completar o desafio. A interface mostra a fórmula da refração e permite ao jogador inserir a resposta.

- **Atributos:**
    - pergunta: A string com a questão sobre ângulo de refração e índices de refração.
    - resposta: A resposta correta do desafio (ex: "33,83").
    - desafioConcluido: Booleano que indica se o desafio foi concluído.
    - faseJogo: Referência à fase do jogo associada a este desafio.
    - respostaS: A letra misteriosa obtida ao completar o desafio.
    - tentativa: Armazena a tentativa de resposta do jogador.
- **Métodos:**
    - render(): Exibe a pergunta, a fórmula de refração, e o campo para o jogador inserir a resposta. Se a resposta estiver correta, uma mensagem indicando a letra misteriosa é exibida.
    - mousePressed(): Verifica se o jogador clicou para avançar após completar o desafio.
    - keyPressed(char key): Lida com a entrada de teclado, permitindo o jogador inserir a resposta e remover caracteres com o BACKSPACE.

Classe DesafioResistor

Este desafio envolve a montagem de um circuito com cinco resistores, onde o jogador alterna entre configurações em série e paralelo até alcançar uma resistência total específica.

- **Atributos:**
    - resistores: Array que armazena os 5 resistores.
    - resistenciaTotal: Armazena a resistência total do circuito.
    - resistenciaAlvo: A resistência alvo que o jogador precisa alcançar para vencer (ex: 10 ohms).
    - venceu: Indica se o jogador venceu o desafio.
    - valorFixo: Valor fixo dos resistores (ex: 50 ohms).
    - faseJogo: Referência à fase do jogo associada a este desafio.
- **Métodos:**
    - calcularResistencia(): Calcula a resistência total do circuito, considerando resistores em série e paralelo.
    - render(): Desenha os resistores e exibe suas configurações, além de mostrar a resistência total e alvo. Quando a resistência alvo é atingida, o jogador recebe uma letra misteriosa.
    - mousePressed(): Alterna entre as configurações série/paralelo ao clicar nos resistores.

Classe DesafioFE

Neste desafio, o jogador deve calcular a distância correta entre duas cargas para que a força elétrica atinja um valor dado. A fórmula da força elétrica é exibida na tela, e o jogador insere a distância correta.

- **Atributos:**
    - resposta: Resposta correta para o desafio (ex: "200").
    - pergunta: Pergunta que envolve o cálculo da força elétrica com base na distância entre duas cargas.
    - tentativa: Armazena a tentativa de resposta do jogador.
    - posX: Posição X do objeto no cenário.
    - imaX, imaY: Variáveis para auxiliar alinhamento da fórmula.
    - desafioCompleto: Indica se o desafio foi completado.
    - mostrarLetra: Indica se a letra misteriosa deve ser exibida.
    - faseJogo: Referência à fase do jogo associada a este desafio.
- **Métodos:**
    - render(): Desenha os objetos e a fórmula da força elétrica na tela, além de exibir a tentativa do jogador. Se a resposta estiver correta, a letra misteriosa é mostrada.
    - keyPressed(char key): Permite a entrada de caracteres para a tentativa do jogador, incluindo apagar com BACKSPACE.
    - mousePressed(): Verifica se o desafio foi completado e avança o jogo.