**CENTRO UNIVERSITÁRIO BRAZ CUBAS**

**ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

RICHARD RIOS DA SILVA OLIVERA

RONAN BENITIS FERRAZ

**Freeway like game:** Desenvolvimento de aplicação mobile

MOGI DAS CRUZES – SP

2023

RICHARD RIOS DA SILVA OLIVERA

RONAN BENITIS FERRAZ

**Freeway like game:** Desenvolvimento de aplicação mobile

Trabalho avaliativo para a matéria de Sistemas Operacionais apresentado ao professor Claudio Remo Truffa do Centro Universitário Braz Cubas de Mogi das Cruzes, São Paulo, como parte dos requisitos para conclusão da matéria em questão.

Orientador: Prof. Claudio Remo Truffa.

MOGI DAS CRUZES – SP

2023

Dedico este trabalho ao Professor Claudio Remo Truffa. Sua orientação em Sistemas Operacionais, que nos inspira a buscar a excelência. Sua paciência e clareza são inestimáveis.

**AGRADECIMENTOS**

Agradecemos ao Professor Claudio Remo Truffa por sua orientação e apoio durante a realização deste trabalho. Sua sabedoria e paixão pelo ensino foram uma fonte constante de inspiração.

Aos colegas de classe, pela colaboração, pelas discussões produtivas e pelo ambiente de aprendizado cooperativo.

Por fim, ao Centro Universitário Braz Cubas por proporcionar um ambiente acadêmico estimulante e recursos valiosos que foram fundamentais para a realização deste trabalho.

**PROJETO**

# JUSTIFICATIVA

Decidiu-se seguir com o desenvolvimento de uma aplicação mobile de um jogo ao estilo do clássicoFrogger do Atari, esse tipo de jogo é divertido, desafiador e nostálgico para muitos usuários. Além disso, uma aplicação mobile permite que o jogo seja acessado em qualquer lugar e a qualquer momento, aumentando o engajamento e a fidelidade dos jogadores. Para desenvolver a aplicação, foi utilizado linguagem de programação Javascript, que é uma das mais populares e versáteis do mercado, permitindo criar interfaces interativas e responsivas. O projeto do curso da Alura foi usado como guia, pois oferece um passo a passo detalhado e didático de como criar um jogo do zero, desde a lógica até a estilização.

# OBJETIVO

Desenvolver um jogo divertido e desafiador que possa ser jogado em diferentes dispositivos móveis, como smartphones e tablets, aproveitando as funcionalidades de cada plataforma. O jogo deve ser fácil de aprender e jogar, mas também oferecer níveis de dificuldade crescente e recompensas para os jogadores mais habilidosos. O jogo deve ter um design atraente e colorido, que remeta ao estilo retrô do Frogger original, mas com elementos modernos e criativos. O código deve ser bem estruturado e documentado, seguindo as boas práticas de programação e os padrões da indústria, para facilitar a manutenção, a atualização e a colaboração entre desenvolvedores de diferentes níveis de experiência.

**FASEAMENTO DO DESENVOLVIMENTO DO SOFTWARE**

# CÓDIGO

O QUE É PS5. JS?

P5.js é uma biblioteca JavaScript que facilita a codificação criativa, baseada no ambiente de codificação *Processing*. Ela é projetada para tornar a programação visualmente atraente e acessível, especialmente para artistas, designers, educadores e iniciantes. Com p5.js, você pode criar arte interativa, jogos, visualizações de dados e animações diretamente na página do navegador, e ela oferece bibliotecas adicionais para interagir com outros objetos HTML5, como texto, entrada, vídeo, webcam e som. É uma ferramenta poderosa para quem deseja explorar a programação em um contexto visual e interativo.

**PARTE 1: ATOR**

Personagem controlado pelo jogador que move na tela e evita colisões.

Variáveis iniciais:

yAtor e xAtor são as coordenadas do ator no jogo.

velocidadeAtor é a velocidade de movimento do ator.

dimensaoAtor é o tamanho do ator.

colisao é um booleano que indica se ocorreu uma colisão.

pontos é a pontuação do jogador.

Funções:

movimentaAtor(): Esta função move o ator com base nas teclas pressionadas pelo usuário (cima, baixo, esquerda, direita). A função mantemDentroTela() é chamada para garantir que o ator não saia dos limites da tela.

mostraAtor(): Esta função exibe o ator na tela.

verificaColisao(): Esta função verifica se o ator colidiu com algum dos carros. Se houve uma colisão, o som de colisão é tocado, o ator é retornado à posição inicial e a pontuação é reduzida se for maior que zero.

retornaAtorPontoInicial(): Esta função retorna o ator à posição inicial.

mostraPontos(): Esta função exibe a pontuação atual do jogador na tela.

marcaPontos(): Esta função aumenta a pontuação do jogador se o ator passar de uma certa linha (yAtor < 15), toca um som e retorna o ator à posição inicial.

pontosMaiorQueZero(): Esta função parece estar incompleta, pois não há código dentro dela.

mantemDentroTela(): Esta função garante que o ator não saia dos limites da tela.

**PARTE 2: CARROS**

Obstáculos que o jogador deve evitar enquanto tenta atravessar a estrada para chegar ao outro lado em segurança.

Variáveis iniciais:

xCarros e yCarros são listas que contêm as coordenadas x e y de cada carro no jogo.

velocidadeCarros é uma lista que contém a velocidade de cada carro.

comprimentoCarro e alturaCarro são as dimensões dos carros.

Funções:

mostraCarro(): Esta função exibe cada carro na tela. Ela percorre a lista imagemCarros e usa a função image() para desenhar cada carro na tela nas coordenadas especificadas.

movimentaCarro(): Esta função move cada carro para a esquerda na tela (diminuindo a coordenada x) com base em sua velocidade.

voltaPosicaoInicialDoCarro(): Esta função verifica se cada carro passou completamente pela tela (se a coordenada x é menor que -80). Se sim, o carro é retornado à posição inicial (x = 600).

passouTodaATela(x): Esta função retorna verdadeiro se o carro passou completamente pela tela (se x < -80).

**PARTE 3: IMAGENS E SONS**

Código responsável por carregar as imagens e sons que serão usados no jogo. Explicação das partes principais:

Variáveis:

imagemEstrada, imagemAtor, imagemCarro1, imagemCarro2, imagemCarro3 são variáveis que armazenarão as imagens da estrada, do ator e dos carros, respectivamente.

somTrilha, somColisao, somPonto são variáveis que armazenarão os sons da trilha sonora do jogo, da colisão e do ponto, respectivamente.

Função preload():

Esta função é usada para carregar recursos (como imagens e sons) antes do início do jogo. Isso é feito para evitar atrasos durante o jogo enquanto os recursos são carregados.

As imagens são carregadas usando a função loadImage(), que recebe o caminho para o arquivo de imagem como argumento.

Os sons são carregados usando a função loadSound(), que recebe o caminho para o arquivo de som como argumento.

A variável imagemCarros é uma lista que contém as imagens dos carros. Ela é preenchida com as imagens dos carros carregadas anteriormente.

**PARTE 4: COLISÃO**

Este trecho do código JavaScript é uma biblioteca de detecção de colisão chamada p5.collide2D. Ela fornece várias funções para detectar colisões entre diferentes formas geométricas em um espaço 2D. Partes principais:

Variáveis:

\_collideDebug é uma variável booleana usada para ativar ou desativar o modo de depuração.

Funções:

collideDebug(debugMode): Esta função é usada para ativar ou desativar o modo de depuração.

collideRectRect(x, y, w, h, x2, y2, w2, h2): Esta função verifica se dois retângulos estão colidindo.

collideRectRectVector(p1, sz, p2, sz2): Esta é uma versão da função collideRectRect que aceita objetos p5.Vector em vez de coordenadas x e y.

collideRectCircle(rx, ry, rw, rh, cx, cy, diameter): Esta função verifica se um retângulo e um círculo estão colidindo.

collideRectCircleVector(r, sz, c, diameter): Esta é uma versão da função collideRectCircle que aceita objetos p5.Vector.

collideCircleCircle(x, y, d, x2, y2, d2): Esta função verifica se dois círculos estão colidindo.

collideCircleCircleVector(p1, d, p2, d2): Esta é uma versão da função collideCircleCircle que aceita objetos p5.Vector.

collidePointCircle(x, y, cx, cy, d): Esta função verifica se um ponto e um círculo estão colidindo.

collidePointCircleVector(p, c, d): Esta é uma versão da função collidePointCircle que aceita objetos p5.Vector.

collidePointEllipse(x, y, cx, cy, dx, dy): Esta função verifica se um ponto e uma elipse estão colidindo.

collidePointEllipseVector(p, c, d): Esta é uma versão da função collidePointEllipse que aceita objetos p5.Vector.

collidePointRect(pointX, pointY, x, y, xW, yW): Esta função verifica se um ponto e um retângulo estão colidindo.

collidePointRectVector(point, p1, sz): Esta é uma versão da função collidePointRect que aceita objetos p5.Vector.

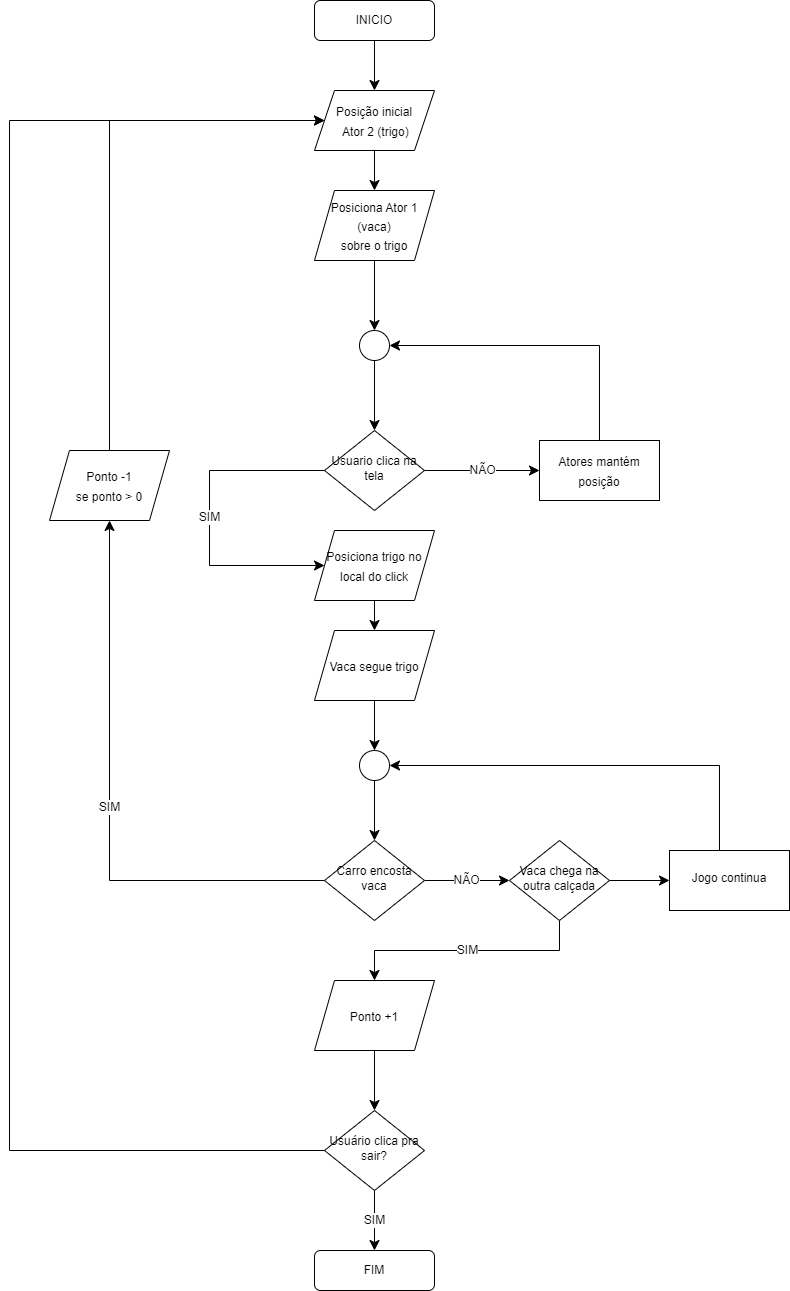
collidePointLine(px, py, x1, y1, x2, y2, buffer): Esta função verifica se um ponto e uma linha estão colidindo. A variável buffer é usada para adicionar uma pequena zona de buffer que dará colisão.

Funções:

collidePointArc(px, py, ax, ay, arcRadius, arcHeading, arcAngle, buffer): Esta função verifica se um ponto e um arco estão colidindo. A variável buffer é usada para adicionar uma pequena zona de buffer que dará colisão. Se buffer for indefinido, ele será definido como 0. A função cria vetores para o ponto, a posição do arco e o raio do arco. Ela então verifica se a distância do ponto ao arco é menor ou igual ao raio do arco mais o buffer. Se for, ela calcula o produto escalar e o ângulo entre o vetor raio e o vetor do ponto ao arco. Se o produto escalar for maior que 0 e o ângulo estiver dentro do ângulo do arco, a função retorna verdadeiro. Caso contrário, ela retorna falso.

collidePointArcVector(p1, a, arcRadius, arcHeading, arcAngle, buffer): Esta é uma versão da função collidePointArc que aceita objetos p5.Vector em vez de coordenadas x e y.

**FLUXOGRAMA DO FUNCIONAMENTO DO SOFTWARE**

****

# INSTALAÇÃO E CONFIGURAÇÃO

Não é necessária nenhuma instalação para jogar o jogo. Como é uma aplicação web, você pode jogar diretamente em qualquer navegador moderno, seja em um dispositivo móvel ou desktop. Basta acessar o seguinte link para começar a jogar: <https://editor.p5js.org/RonanFerraz/full/QSjrPD7Wp>Certifique-se de que o seu navegador está atualizado para a melhor experiência de jogo.

# USO

Como Jogar

O jogo é controlado pelo posicionamento de um trigo na tela, que guia a vaca entre os carros em movimento. O objetivo é atravessar a vaca com segurança para o outro lado da estrada, evitando colisões com os carros.

Controles

Posicionamento do Trigo: Toque ou clique na posição desejada na tela para colocar o trigo. A vaca seguirá em direção ao trigo.

Movimento da Vaca: A vaca se move automaticamente em direção ao trigo. Certifique-se de reposicionar o trigo para guiar a vaca com segurança entre os carros.

Dicas

Planeje seus movimentos com antecedência para evitar ficar preso entre os carros.

Use movimentos rápidos e precisos para reposicionar o trigo e manter a vaca em movimento.

# REFERÊNCIAS

Cursos de JavaScript: Estes cursos ofereceram a base para a lógica e o desenvolvimento do nosso projeto. Disponível em: Curso JavaScript: <https://cursos.alura.com.br/course/javascript-listas-lacos,> <https://cursos.alura.com.br/course/pong-javascript>

p5.js: Uma biblioteca JavaScript que facilita a programação criativa. Saiba mais sobre o p5.js em: <https://p5js.org/>

A documentação oficial das funções do p5.js e suas interpretações em JavaScript. Confira em: [https://p5js.org/reference](https://www.youtube.com/watch?v=JPpmUPcmIKs)

Input Touch no p5.js: Um vídeo explicativo sobre como implementar input touch no p5.js. Assista em: <https://www.youtube.com/watch?v=JPpmUPcmIKs>