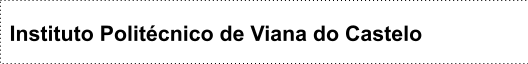
A picture containing company name

Description automatically generated

A picture containing text

Description automatically generatedA picture containing text, sign

Description automatically generated

A picture containing text

Description automatically generated

Uma imagem com texto

Descrição gerada automaticamente

Índice

[Introdução: 3](#_Toc139828534)

[Desenho da mesa: 4](#_Toc139828535)

[Taco de bilhar: 6](#_Toc139828536)

[Bolas de bilhar: 9](#_Toc139828537)

[Disparo da bola branca: 9](#_Toc139828538)

[Colisão entre as bolas: 10](#_Toc139828539)

[Colisão com os cantos da mesa: 14](#_Toc139828540)

[Entrada nos buracos: 15](#_Toc139828541)

[Logica de jogo 16](#_Toc139828542)

[Conclusão 17](#_Toc139828543)

[Referencias 17](#_Toc139828544)

# Introdução:

O trabalho seguinte consiste no desenvolvimento do jogo proposto no “Trabalho 3” : “Simulação do jogo do Pool (Bola 8)”

Para correr o código basta abrir o index.html

Para jogar o jogo deve clicar no botão esquerdo do rato durante o tempo que achar melhor a fim de tentar colocar as bolas dentro dos buracos.

O código encontra-se dividio entre os assets,inputs e o javascript/html do programa.

O Assets.js carrega os sprites do jogo.

No input tem funcionalidades do rato como a localização do mouse e se os botões estão pressionados.

No Game.js temos a inicialização e o mainloop do jogo.

No Canvas.js temos a instância do canvas2d para funcionalidades de desenho do canvas

O Vector.js contem funções que eu utilizei ao longo do código (estas serão explicadas em detalhe durante o relatório)

O color.js contem as cores das bolas

O taco.js e bola.js e gameworld.js contem o código mais imporante para o funcionamento do jogo que será explicado ao longo do relatório

Uma imagem com captura de ecrã, Software de multimédia, design

Descrição gerada automaticamente

# Desenho da mesa:

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

O seguinte código foi retirado do stackoverflow para facilitar a vista de onde estão os pixéis exatos no código. (quando o rato vai para baixo diz a posição do click)

Posicionamento do rato no ecra:

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Ver qual dos botões do rato esta premido 1 para esquerda 2 para meio e 3 para direita:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

O mesmo se aplica para quando largamos o botão:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Para funções de teste que os botões estavam a ser premidos e que o rato estava a ser tracked corretamente corri o seguinte código e atribui a posição do taco ao rato.

Uma imagem com texto, captura de ecrã, Tipo de letra, design

Descrição gerada automaticamente

# Taco de bilhar:

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

A posição do taco será passada com uma variável (position) para este se alinhar com a bola branca, a rotation irá ser definida a apontar para o posicionamento do rato. A origem serve para ter a posição não ser o canto superior esquerda da imagem como é o default no canvas. A força será a força aplicada à bola no evento do disparo. Ondisparo será como o nome indica a função de disparo o dispr servirá para saber se o taco está em estado de disparo.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

O taco é posicionado por trás da bola branca e ligado à mesma

Uma imagem com texto, file, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

No lugar do gato temos o taco e o ponto vermelho é a posição do rato, nos queremos rodar o taco para que fique a apontar para o ponto vermelho. Para isso é preciso descobrir o “angle”, para isso é preciso descobrir o anglo oposto e a adjacente, então no lugar de catx,caty temos a coordenadas do taco e pointerx,pointery as coordenadas do rato. Com estes dados fazemos temos a tangente e arctangente que nos permite descobir então o anglo que temos que dar para a rotação

Uma imagem com texto, Tipo de letra, branco, design

Descrição gerada automaticamenteUma imagem com Tipo de letra, texto, branco, file

Descrição gerada automaticamente

Traduzindo-se assim para o código:

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Uma imagem com captura de ecrã, texto, Software de multimédia

Descrição gerada automaticamente

Reposicionamento do taco para a nova posição da bola quando esta estiver parada. Bolas não estão em movimento e o taco está no estado disparado.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

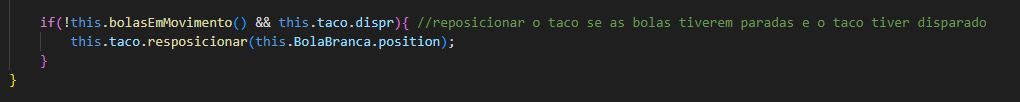
A força vai ser utilizada para dar velocidade à bola e a origem.x faz com que o taco progressivamente va para tràs à medida que seguramos o right click do rato.

Uma imagem com texto, captura de ecrã, software, Sistema operativo

Descrição gerada automaticamente

Desenho + alinhamento a volta da bola branca no draw.

A rotação é definida pela função vista anteriormente, quando a bola se para de movimentar o posicionamento do taco é atualizado para o novo posicionamento da bola, esta função e chamada no gameworld como viste no seguinte excerto de código:



# Bolas de bilhar:

Uma imagem com texto, captura de ecrã, ecrã, software

Descrição gerada automaticamente

Existem 4 tipos de bolas onde varia a cor: branca, preta , amarela e vermelha o posicionamento foi feito de acordo com o jogo real.

# Disparo da bola branca:

Sendo o centro a posição da bola o centro da seguinte circunferência então a velocidade deverá ser um vetor novo com a o cos do angulo como x e sin do angulo como y a fim de se dirigir na direção do ponteiro do rato. A força que aplicamos no taco será multiplicada para aumentar a velocidade em que a bola se move na direção.

Uma imagem com file, diagrama, círculo, Gráfico

Descrição gerada automaticamente

Uma imagem com captura de ecrã, texto

Descrição gerada automaticamente

Depois de determinar a velocidade adicionamos as coordenadas à bola com a seguinte função

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Utilizando uma multiplicação de 0.98 no x e no y (varx\*x,vary\*y para criar uma ilusão de fricção e abrandar a bola. O delta é uma variável global para controlar quanto é que quero fazer update em cada iteração, no fundo muda a velocidade do jogo.

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

# Colisão entre as bolas:

Uma imagem com texto, captura de ecrã, ecrã, software

Descrição gerada automaticamenteNo Gameworld criei uma função que fica encarregado pelas colisões onde é iterado por todas as bolas do array e verificado inicialmente a colisão pelos cantos da mesa, depois iterar outra vez pelas bolas para colisão entre duas bolas e iterar pelos buracos para colisão entre as bolas e os buracos

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Este método de colisão entre bolas chama-se ellastic collision, que pode ser defenida pelos seguintes passos:

1-encontrar o vetor normal subtraindo a posição de uma das bolas pela outra como visto na proxima equação: e no código anterior como a const N

Uma imagem com Tipo de letra, texto, branco, escrita à mão

Descrição gerada automaticamenteUma imagem com diagrama, círculo, design

Descrição gerada automaticamente

A distância é utilizada para verificar se existe ou não colisão entre duas bolas.

De seguida calculamos o unit vector do nosso n que é feito pela seguinte equação:

Uma imagem com texto, Tipo de letra, captura de ecrã, branco

Descrição gerada automaticamente

Em terceiro voltamos á próxima figura para descobrir a tangente da superfície das bolas no ponto de colisão:

Uma imagem com diagrama, círculo, design

Descrição gerada automaticamente

Este pode ser calculado com o seguinte uma vez que já temos o vetor un:

Uma imagem com Tipo de letra, texto, branco, tipografia

Descrição gerada automaticamente

De seguida calculamos os vetores de velocidade da tangente e normal com as seguintes equações:

Uma imagem com texto, Tipo de letra, branco, tipografia

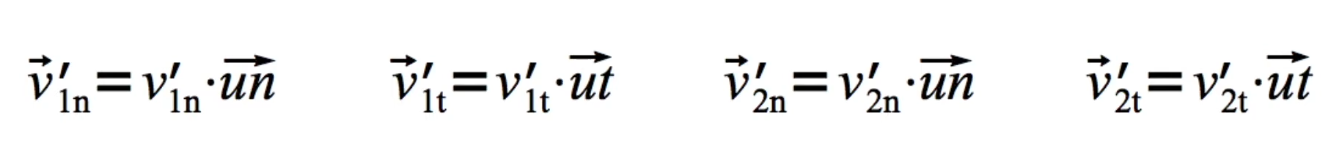
Descrição gerada automaticamente

Agora encontramos as novas velocidades normais para o depois da colisão omitindo a massa uma vez que as bolas têm todas a mesma massa

Uma imagem com texto, diagrama, Tipo de letra, file

Descrição gerada automaticamente

De seguida queremos converter as normais escalares e as novas velocidades em vetores



E por fim atualizamos as velocidades das nossas bolas:

Uma imagem com Tipo de letra, escrita à mão, branco, texto

Descrição gerada automaticamente

O método que foi explicado anteriormente foi recomendado por stackoverflow e foi retirado do seguinte artigo: <https://www.vobarian.com/collisions/2dcollisions2.pdf>

Existia um problema em que as bolas ficavam sobrepostas umas com as outras. Foi resolvido adicionado uma distância mínima em que as bolas podem estar umas das outras e empurrar-las caso esta não seja superior:

Uma imagem com bola, Jogos e desportos indoor, desporto, Jogos

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

# Colisão com os cantos da mesa:

A colisão com os cantos da mesa é muito mais simples bastando detetar se o Y ou o X da bola está a tocar na mesa e inverter a velocidade do x ou do y da bola a realizar contacto conforme o lado da mesa que está a ser tocado.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente´

Existia, no entanto, um bug em que as bolas ficam coladas aos cantos da mesa durante o toque:

Uma imagem com bola, ovo, amarelo

Descrição gerada automaticamente

Isto foi resolvido enviando a bola para as coordenadas da mesa:

Uma imagem com texto, captura de ecrã, software, ecrã

Descrição gerada automaticamente

Adicionei também abrandamento na velocidade quando acerta em uma parede visto no próximo enxerto de código:

Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente

# Entrada nos buracos:

Defini um array para as localizações dos vários buracos da mesa utilizando a sua posição absoluta no centro de cada buraco.

Uma imagem com texto, captura de ecrã, Tipo de letra, design

Descrição gerada automaticamente

A colisão é verificada da mesma maneira que a colisao entre as bolas na parte do handlecollisions no gameworld. Foi criado mais uma função para verificar a distancia de um vetor a outro:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Se a distancia da bola ao buraco for menor que 42 (radio do buraco) então a bola é dada como que tenha entrado no buraco, passando esta para invisível e parada.

Uma imagem com texto, captura de ecrã, Tipo de letra

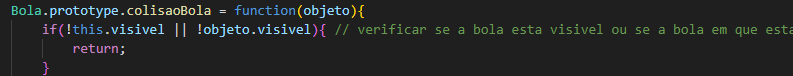
Descrição gerada automaticamente

No fundo adicionei variável de visibilidade e coloco invisível se entrar no buraco.

Adicionei também no draw,u pdate e nas colisões check’s para verificar se esta visível antes de interagir com o objeto garantindo que uma bola que tenha entrado no buraco está fora do resto do jogo.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente



# Logica de jogo

Não consegui acabar a lógica do jogo toda sendo que faltam algumas das regras que se encontram em um jogo normal devido a falta de tempo. Falta ter em conta as faltas do jogo, e o estado de vitória.

Faço a cotação de quantas bolas cada jogar inserte nos buracos a fim de depois saber que estamos na última bola para acabar mas ainda não diferencio se o jogador esta a meter as bolas certas. Tenho que criar uma instancia de Jogador que tenha o tipo de cor a que este está a jogar e recolocar bolas erradas em jogo.

Caso a bola branca seja introduzida volta ao centro e passa o turno.

O jogo reinicia quando a bola preta for introduzida.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

# Conclusão

O jogo encontra-se em um estado funcional, mas incompleto faltando implementar algumas das regras que se encontram no snooker tal como apenas poder jogar com o primeiro tipo de bolas inserido. Estando no entanto cumpridas as regras de desenvolvimento a), b), d), e) , g)

# Referencias

Rotação: [They’ve got atan, You Want atan2 | The Sinepost (wordpress.com)](https://sinepost.wordpress.com/2012/02/16/theyve-got-atan-you-want-atan2/)

Colisão: <https://www.vobarian.com/collisions/2dcollisions2.pdf>

Stackoverflow colisao : [android - Pool game in java - ball collision algorithm - Stack Overflow](https://stackoverflow.com/questions/18189096/pool-game-in-java-ball-collision-algorithm)