FACULDADE ENGENHEIRO SALVADOR ARENA

ENGENHARIA DA COMPUTAÇÃO

**CÁLCULO E SIMULAÇÃO DE UM LANÇAMENTO BALÍSTICO COM ALVO MÓVEL**

Adriana Kaori Kakazu RA: 082220004

Beatriz dos Santos Buglio RA: 082220028

Diego de Souza Lima RA: 082220042

Lucas Oliveira Silva RA: 082220019

Vitória Kaori Kuriyama RA: 082220005

SÃO BERNARDO DO CAMPO  
2023

**SUMÁRIO**

**1 INTRODUÇÃO.............................................................................................. 3**

**2 METODOLOGIA**

**2.1 CÁLCULOS UTILIZADOS........................................................... 4**

2.2 DESCRIÇÃO DO PROGRAMA EM C#.......................................... 5

2.3 DESCRIÇÃO DO BANCO DE DADOS.......................................... 8

2.4 CONEXÃO DO PROGRAMA COM O BANCO DE DADOS....... 10

**3 RESULTADOS DA SIMULAÇÃO.............................................................. 14**

**4 CONCLUSÕES.............................................................................................. X**

**REFERÊNCIAS BIBLIOGRÁFICAS............................................................ X**

**1 INTRODUÇÃO**

Desde os seus primórdios, uma das maiores aplicações da programação é a simulação e resolução de cálculos matemáticos complexos, muitas vezes aplicados a situações específicas da física, visando obter resultados que sejam os mais próximos da realidade possível. O objetivo desse trabalho é, justamente, simular uma dessas situações, obter resultados utilizando ferramentas de Cálculo Avançado, através de um programa desenvolvido em C# e vinculá-los obtidos a um banco de dados.

Esse trabalho visa encontrar a trajetória de um projétil, que, disparado a partir do solo em um terreno plano e horizontal, é lançado obliquamente com o objetivo de atingir um alvo, representado por um meteoro, que apresenta uma trajetória descendente vertical com relação ao solo, com uma velocidade inicial constante.

O ângulo de disparo deve ser fornecido pelo usuário, enquanto a altura da qual o alvo cai e a distância entre o canhão de onde será feito o disparo do projeto e o alvo móvel foram estabelecidos previamente pelos desenvolvedores. Esses se utilizaram de valores razoáveis encontrados através de pesquisas. A velocidade inicial de disparo, por sua vez, pode ser estabelecida pelo usuário ou ser automática conforme o ângulo selecionado.

Os cálculos utilizados foram feitos através de um software desenvolvido em C#, utilizando-se das Séries de Taylor e MacLaurin como ferramentas lógicas (estas tiveram sua lógica traduzida para a linguagem de programação selecionada e foram aplicadas dessa maneira). Os resultados produzidos foram o intervalo de tempo , a informação de qual momento de sua trajetória o projétil se encontrava quando atingiu o alvo (ascendente ou descendente) e, caso o ângulo fornecido não resultasse no alvo sendo atingido, uma sugestão de intervalo possível de valores.

Esses resultados são, por fim, armazenados em um banco de dados, para que consultas posteriores possam ser realizadas. Por sua vez, o banco de dados foi desenvolvido utilizando a linguagem SQL, e modelado através das chamadas SPs (Stored Procedures), com sua conexão ao programa sendo estabelecida através do chamado CRUD. A trajetória é, então, demonstrada através de uma simulação em Windows Forms, também programada em C#.

**2** **METODOLOGIA**

2.1 CÁLCULOS UTILIZADOS

Para calcular os cossenos e senos utiliza-se a série de Maclaurin, onde, a partir de um ângulo ajustado para ser menor ou igual a noventa, respeitando o círculo trigonométrico e convertido para radianos, multiplicando por e dividido por 180. Com isso, o programa entra em um loop onde primeiro verifica em qual iteração se encontra e, a partir disso, define qual derivada de seno ou cosseno está no momento.

A partir disso, pode-se começar a somar os resultados usando a fórmula (Soma = Soma + (Derivada\_Atual \* Angulo^2)/Iteração!) e, assim, são representadas as infinitas somas da série. Após isso é feito um teste para confirmar se é necessário somar mais uma iteração. Nele, é verificada a precisão, sendo adotada, nesse caso, a de 1\*10^-5. Para isso, é calculado o resultado da fórmula (Teste = (1/(iteração+1)!) \* Angulo^(Iteração+1)) e, caso o resultado do Teste seja menor que valor da precisão, podemos parar a soma, caso contrário, o processo é repetido. No caso da tangente é calculado o seno e cosseno de ambos do angulo e, com eles, é obtida a tangente como tan = sen/cos.

Para calcular a posição do meteoro foi utilizada a função horária do espaço (S = S0 + V0\*T + (A/2) \* T^2), considerando a aceleração como - 9,8m/s^2 (devido ao referencial da gravidade), V0 como velocidade inicial em Y (obtido por VY = sen(angulo) \* V0), S0 como a posição inicial em Y do meteoro e o tempo t como um valor recebido pela função. Como não há movimento horizontal a posição no eixo X não se altera.

Já o cálculo da posição do projetil é separado em dois eixos, vertical e horizontal. Para isso, é calculada inicialmente a posição X usando a equação da posição em movimento retilíneo uniforme (S = S0 + V0\*T), já que não há mudança na aceleração. S0(posição inicial) é 0 porque o movimento parte do solo, V0 é a velocidade inicial em X (VX = Cos(Angulo) \* V0) e o tempo é a variável recebida.

Como o eixo Y possui movimento parabólico, é adotada uma equação da trajetória definida por (Y = Tan(Angulo) \* X – G/(2V0^2 \* Cos^2(Angulo)) \* X^2), onde X é a posição atual dele neste eixo, G é 9.8 m/s^2 e V0 é a velocidade inicial de lançamento oferecida pelo usuário.

Por fim o cálculo utilizado para descobrir o momento de contato entre o meteoro e o projetil é obtido através da comparação |PosiçãoMeteoro – PosiçãoProjetil| <= 125, tanto para o eixo X quanto o Y, sabendo que 125 é o raio do meteoro, significando que o projetil atingiu o meteoro em algum lugar da área do meteoro.

2.2 DESCRIÇÃO DO PROGRAMA EM C#

A criação do software consiste na definição de um meteoro que cai verticalmente com velocidade constante a partir de uma posição inicial e um projetil lançado a partir do solo que será lançado em um movimento obliquo os dois seguirão os mesmos intervalos de 1 segundo, onde serão calculadas as suas posições através de fórmulas oferecidas no documento do trabalho com base em um ângulo que terá seu seno e cosseno obtidos pela série de Maclaurin. Assim, será possível observar o movimento de ambos os objetos em relação ao tempo até um possível momento de encontro entre os dois objetos. Para isso, podem ser utilizados o modulo da diferença da posição e altura de ambos para verificar se eles então no mesmo ponto, assim achando o ponto final do lançamento.

Tanto o meteoro quanto o projetil são objetos de classe Objeto que á classe pai de ambas. Esta possui como seus atributos velocidade inicial, espaço inicial, espaço atual tanto na coordenada X quanto na Y, velocidade em X e Y, ângulo, seno e cosseno.

A classe Meteoro deriva de Objeto e possui as mesmas características, com adição do método construtor que pede velocidade e posição inicial e da capacidade de definir seu seno e velocidade em Y. Há, também, o método de movimentação responsável por calcular a posição do meteoro em função do tempo utilizando a função horária do espaço do MUV (Movimento Uniformemente Variado).

O meteoro, apesar de poder ter sua posição inicial modificada, tem suas características iniciais previamente definidas. São estas a distância horizontal de 4000 metros, a altura de 5000 metros (distância vertical) e a velocidade inicial de 100 m/s que não pode ser alterada. Além disso, o meteoro é considerado um corpo extenso de 250 metros de diâmetro.

A classe Projetil, também derivada de Objeto, pede a velocidade inicial e o ângulo em seu construtor. Este calcula seu seno, cosseno, velocidade em X e em Y e define sua posição inicial como 0. Dentre os métodos exclusivos de Projetil temos o movimento em X, que calcula a posição X do projetil com base em seu ângulo, tempo atual e velocidade em X utilizando a função horaria do espaço do movimento retilíneo uniforme. Enquanto para calcular a posição em Y é utilizada a equação da trajetória, que determina a altura do projétil em função da posição em X, considerando também a velocidade inicial e ângulo de lançamento. Por fim, há o método de mudar valores, que altera as características do projétil com base em uma nova velocidade inicial e ângulo de lançamento.

Neste programa há duas classes importantes que abrigam a lógica por trás dos cálculos. A primeira delas é a classe FunçõesFisica a qual possui como objetivo realizar cálculos com base em funções da física que, através das variáveis oferecidas, devolve um resultado. Ela é responsável por todos os cálculos físicos do programa.

A segunda é a classe Ângulo, que possui como objetivo descobrir o seno, cosseno e tangente de um ângulo. Esta classe possui algumas funções internas que são utilizadas para atingir esse objetivo, como a de arrumar ângulo, que ajusta o ângulo com base no círculo trigonométrico, permitindo o usuário utilizar ângulos maiores que 90 que serão considerados como equivalentes aos menores e por fim converte o ângulo para radianos. A classe possui, também, a função de fatorial, já que não há uma biblioteca previamente instalada no C# para este objetivo.

Seguindo para as funções principais, há a de cosseno, que devolve o valor do cosseno de um ângulo pedido utilizando a série de Taylor com precisão de 10^-5 para achar o valor pedido. A função seno, por sua vez, utiliza o mesmo método de cálculo para devolver seu valor, porém, levando em consideração as derivadas de seno que possuem ordem diferente da do cosseno. Por fim, para o cálculo da tangente é necessário apenas utilizar o resultado do método de seno e dividí-lo pelo do cosseno.

A classe Cérebro é a parte mais importante do programa, sendo responsável por estabelecer a lógica. Nela são criados objetos da classe meteoro e projetil, além das diversas variáveis que são utilizadas na lógica do programa.

Seu primeiro método é o definir, que guarda a velocidade e ângulo pedidos e limpa as demais variáveis para evitar a persistência de dados de testes anteriores, utilizando o método Resetar. Também há os métodos MudarPosiçãoMeteoro, que altera a posição inicial do meteoro, PrepararMeteoro, que devolve o valões dessa posição, a função ProjPosY, que calcula a posição y do meteoro no tempo atual e a diferença dessa posição em relação a última posição dele, verifica se está subindo ou descendo e, por fim, retorna o valor da diferença arredondada dividida pela proporção de pixel e metros(1px=10m).

Também existem funções parecidas para calcular a posição X do projetil, X do meteoro e Y do meteoro nas quais fazem o mesmo, há as funções que verificam se o meteoro e projétil atingem o solo, funções que devolvem em texto a posição atual dos objetos e as funções de testes de valor ideal, que retornam uma lista de valores ideais de ângulo ou velocidade inicial.

Para isso, o usuário fornece um ângulo ou uma velocidade inicial e o programa faz uma bateria de testes internos buscando com base em uma das duas informações. Ocorre, então, a verificação de se o ângulo oferecido é suficiente para que o projétil acerte o meteoro com alguma velocidade, ou se há algum ângulo que permita que a velocidade inicial oferecida acerte o objetivo.

Por fim, há o método de contato, que calcula a diferença da posição final do meteoro e do projétil e verifica se valor em módulo delas é menor que o diâmetro do meteoro. Caso positivo, é considerado que houve o contato entre os dois, devolvendo a resposta para essa pergunta em um valor booleano (*true* ou *false*).

O programa começa em uma tela inicial que permite o usuário digitar um valor de ângulo e velocidade inicial. Além disso, há dois botões de busca: um para busca com base no ângulo oferecido, que devolve uma lista de velocidade na qual o projétil acertará o meteoro, e outro que faz o oposto. Há, também, um botão que leva para uma tela que permite alterar a posição inicial do meteoro e um outro que permite abrir uma tela que se conecta com o banco de dados para mostrar os resultados de testes anteriores.

Com o ângulo e velocidade oferecidas pelo usuário é possível entrar na janela do simulador, que é dividida em duas partes. Uma consiste nas informações da simulação que é formada pela posição do meteoro, do projetil, tempo total, de subida, descida e resultado, enquanto outra se resume a botões para recomeçar a simulação e retornar à tela inicial.

Na parte visual é estabelecida uma proporção 1:10, onde 1 px equivale a 10 metros. Na tela é formado um painel que marca os pontos por onde tanto o meteoro quanto o projétil passam, formando sua trajetória, uma caixa vazia invisível que marca a posição de chão, servindo de referencial, e duas que marcam o fim do painel, impedindo que o projetil invada a tela de informações. Ambos o meteoro e projétil são objetos físicos da tela que seguem uma trajetória com base nos cálculos realizados em tempo real pela classe cérebro que, além disso, resolve a parte lógica.

A parte que mais trouxe dificuldades foi a visual, pois, para posicionar um objeto, é necessário converter para pixel sua posição. Essa, por sua vez, precisa de um referencial, neste caso, a caixa invisível chamada chão que é considerada a posição zero, para ser utilizada como posição inicial do projetil. Porém, para o meteoro, o estabelecimento do referencial se mostra mais dificultoso, pois, além da necessidade de conversão em pixel, deve-se somar a posição em X do meteoro e subtrair a em Y, já que os pixels são contados de cima para baixo, logo, para subir, deve haver uma subtração de pixels.

Para fazer a animação é necessária a criação de um timer que cria ciclos que se repetem até que chegue o fim do programa, permitindo observar a movimentação dos objetos. Para que estes se movam, é preciso calcular a diferença das posições atuais e anteriores de ambas que, por sua vez, será convertida em pixels adicionada à esquerda para o movimento horizontal e subtraída de cima durante o movimento vertical.

As partes do programa responsáveis por descobrir os cossenos e senos, a abstração do meteoro e projetil em classes, a preparação das fórmulas da física utilizadas, a formação dos cálculos que verificam a posição dos objetos em relação ao tempo e a verificação do momento em que eles se tocam caso isso aconteça não apresentaram grandes dificuldades de desenvolvimento.

2.3 DESCRIÇÃO DO BANCO DE DADOS

Com o objetivo de orientar a criação e modelamento do banco de dados, foram feitos diagramas com o auxílio da ferramenta BR Modelo, apresentando as entidades presentes no banco, os atributos de cada uma delas e as relações que apresentam entre sí (trata-se de um banco de dados relacional). Os modelos MER e DER, desenvolvido a partir do anterior, são representados a seguir:

Diagrama

Descrição gerada automaticamente

Diagrama, Esquemático

Descrição gerada automaticamente

Os modelos foram desenvolvidos a partir da análise da proposta, aonde foi inicialmente determinada a necessidade de três entidades: Usuário, Projétil e Meteoro. Nesse esquema, o usuário introduz os dados que serão aplicados no cálculo da trajetória do projétil, o que levou à criação de uma entidade associativa, denominada Lanca, cuja função é fazer a associação entre as entidades Usuário e Projétil através do lançamento do projétil. Para englobar a interação entre as entidades Projétil e Meteoro, foi criada outra entidade associativa, essa denominada Intercepta, representando a interceptação da trajetória do projétil pelo meteoro.

Ao banco de dados criado, foi atribuído o mesmo nome do programa, “LancamentoBalistico", reforçando a ligação entre os dois. As tabelas, por sua vez, foram criadas em conformidade com o modelamento de entidades nos diagramas MER e DER. Visando a facilitação do acesso ao cadastro e solicitação de dados, bem como a melhora no processamento dos dados, foram criadas SPs (Stored Procedures) e Triggers.

O resultado produzido ao final deste processo consiste de três arquivos diferentes, criando uma maior organização, o que facilita o entendimento da lógica de cada um dos scripts criados. O primeiro arquivo se trata do script contendo a criação do banco de dados e suas tabelas, o segundo do script de criação das SPs e o terceiro contendo o script de criação das Triggers.

A criação e modelagem do banco de dados, bem como o desenvolvimento de cada um dos três scripts, tiveram como base o conteúdo ministrado em sala de aula, além do livro “modelagem de dados”, do autor José Osvaldo.

2.4 CONEXÃO DO PROGRAMA COM O BANCO DE DADOS

Ao iniciar o projeto, foi considerada a aplicação do modelo DAO (Data Access Object), que acessa e encapsula o mecanismo de acesso a dados enquanto esconde os detalhes de execução da origem desses dados. Esse modelo é muito utilizado ao lidar com bancos de dados relacionais, como o que foi desenvolvido.

Porém, ao considerar a própria documentação da Microsoft, decidiu-se por utilizar o modelo DAL(Data Access Layer – Camada de acesso a dados), já que este possui melhor adequação e é, em tese, uma evolução do próprio DAO. Nessa camada são encontrados todos os comandos de acesso a dados (SELECT, INSERT, UPDATE e DELETE). Nessa camada foram criados todos os métodos que inserem os dados na *Database* (banco de dados), além dos métodos que retornem dados específicos que foram armazenados em simulações anteriores.

Visando desenvolver um software eficiente e estético, foi escolhido o padrão de design MVC (Model View Controller). Esse modelo consiste em separar a interface que é apresentada ao usuário (View), os dados armazenados na *Database* (Model) e a lógica que rege a aplicação desenvolvida, ou seja, o programa em C# (Controller).

Nesse sistema, a camada View é a parte visual, responsável pela interação do usuário com a aplicação, recebendo solicitações e fornecendo os dados em resposta. A camada Model, por sua vez, é responsável pela comunicação com o banco de dados, tendo como função receber as solicitações recebidas pela camada View, encontrar e fornecer os dados correspondentes a ela. Por fim, a camada Controller é a parte lógica do software, responsável por estabelecer a comunicação entre a interface utilizada na View com o banco de dados tratado na camada Model.

A conexão com o banco de dados foi desenvolvida no programa como ilustrado nas capturas de telas a seguir:

Classe Conexão:

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Classe Cadastro:

Texto

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

**3 RESULTADOS DA SIMULAÇÃO**

**4 CONCLUSÕES**

**REFERÊNCIAS BIBLIOGRÁFICAS**