NOME: GUSTAVO HENRIQUE DE MEIRA *RA: 18015*
NOME: GABRIEL DA SILVA JULIO *RA: 18196*

PROJETO: CAMINHO ENTRE CIDADES DE MARTE

Sumário

Introdução

O projeto se trata da confecção de um grafo de caminhos para interligar cidades de Marte que permita exibir as rotas entre elas e desenhar o trajeto de uma à outra. Além disso, deve-se exibir as cidades em uma estrutura de dados chamada árvore, em uma página diferente da do grafo.

Na primeira aba, devemos permitir que o usuário selecione duas cidades distintas para o programa exibir, através de um mapa, os caminhos entre elas. Essas cidades são representadas por um ponto no mapa.

Já na segunda aba, como foi dito, devemos exibir as cidades em uma árvore binária, apenas listando-as.

Desenvolvimento

Datas, horas e tempo de desenvolvimento

A maioria dos horários de desenvolvimento está relacionada com o horário de aulas vagas que tivemos ao longo das semanas, na parte da manhã.

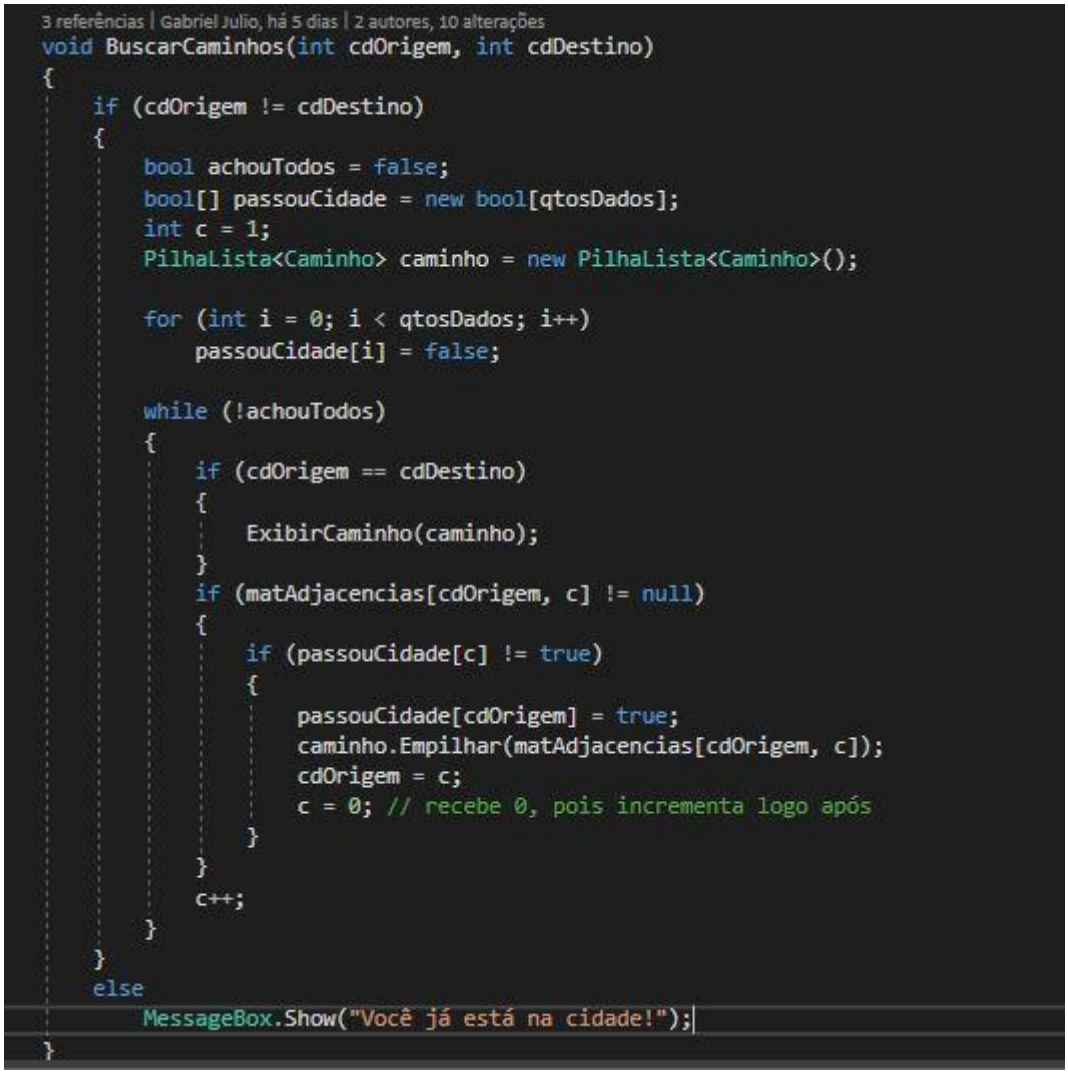
Datas	Horas	Modificações
24/05/2019	07:30 às 09:10	▪ Inserimos as classes Arvore, NoArvore, ListaSimples e Cidade, além da interface IGravarEmArquivo. ▪ Codificamos os construtores da classe Cidade e seus atributos, que serão preenchidos após a leitura de uma linha passada no construtor. ▪ Baseando-se no projeto de árvore de funcionários, desenvolvemos a parte de exibição da árvore de cidades, sem problemas.
27/05/2019	09:30 às 11:55	▪ Desenvolvimento do método CriarPontos(), havendo erros por conta do foreach do vetor de cidades, devido ao seu tamanho, 1000 no caso, que excedia o número de cidades no vetor, impossibilitando a inserção dos pontos cuja posição no vetor estava vazia, resultando em grande X vermelho no Picture box.
28/05/2019	07:30 às 11:55	▪ Começamos a desenvolver o algoritmo responsável por encontrar os caminhos entre as cidades.
29/05/2019	07:30 às 8:00	▪ Fizemos algumas alterações no algoritmo, dentre elas, fizemos uma pilha de pilhas, esta que ira guardar todos os caminhos

		encontrados (Cada caminho é uma pilha!). ▪ *Fizemos também uma alteração para o programa desenhar os pontos a partir dos dados da árvore.*
30/05/2019	*07:30 às 8:00*	▪ *Algoritmo que procura caminhos finalizado.* ▪ *O algoritmo do grafo nos trouxe dificuldade principalmente no momento de retornar por trechos presentes em outras possibilidades, algo que não era necessário ser feito em um labirinto, já que um caminho encontrado já basta. Essa dificuldade nos obrigou a compilar passo a passo o programa para enterdmos em que ponto ignorava caminhos possíveis.* ▪ *Criamos uma interface ICopiavel que nos possibilita usar o método de cópia no algoritmo.* ▪ *Um exemplo do uso do método de cópia está na parte em que empilhamos uma cópia da pilha "Caminho" na pilha de pilhas. Fazemos isso pois logo abaixo desempilhamos o ultimo caminho da pilha para testar todas as possiblilidades, e sem uma cópia, a pilha que foi empilhada seria alterada.*
31/05/2019	*09:30 às 11:55*	▪ *Terminamos a parte de desenhar os caminhos.* ▪ *Descobrimos um erro no algoritmo de procura de caminhos. Quando se ter que ir de uma cidade A, para uma cidade D e existe um caminho, A-B-D, o algoritmo funciona corretamente. Porém, se existir uma cidade C no meio, cujo caminho é A-C-B-D, e o programa já tiver encontrado este primeiro caminho (A-B-D) ele acaba ignorando o*

		novo, já que o vetor "passouCidade" não deixa ele voltar para a cidade B.
10/06/2019	*07:30 às 8:20*	▪ *Conseguimos consertar o erro citado anteriormente apenas mudando o passouCidade[i] = false na cidade que foi desempilhada, fazendo com que o programa possa voltar para a cidade anterior.* ▪ *Adicionamos pequenas mudanças na hora de desenhar os caminhos. O usuário agora tem que selecionar o header da linha do caminho no DataGridView para exibi-lo na tela.* ▪ *Mostramos o melhor caminho também, dessa forma, finalizando o projeto.*

**Observação: não houve auxílio da monitoria no desenvolvimento desse projeto.*

Imagens do desenvolvimento



```
3 referências | Gabriel Julio, há 5 dias | 2 autores, 10 alterações
void BuscarCaminhos(int cdOrigem, int cdDestino)
{
    if (cdOrigem != cdDestino)
    {
        bool achouTodos = false;
        bool[] passouCidade = new bool[qtosDados];
        int c = 1;
        PilhaLista<Caminho> caminho = new PilhaLista<Caminho>();

        for (int i = 0; i < qtosDados; i++)
            passouCidade[i] = false;

        while (!achouTodos)
        {
            if (cdOrigem == cdDestino)
            {
                ExibirCaminho(caminho);
            }
            if (matAdjacencias[cdOrigem, c] != null)
            {
                if (passouCidade[c] != true)
                {
                    passouCidade[cdOrigem] = true;
                    caminho.Empilhar(matAdjacencias[cdOrigem, c]);
                    cdOrigem = c;
                    c = 0; // recebe 0, pois incrementa logo após
                }
            }
            c++;
        }
    }
    else
        MessageBox.Show("Você já está na cidade!");
}
```

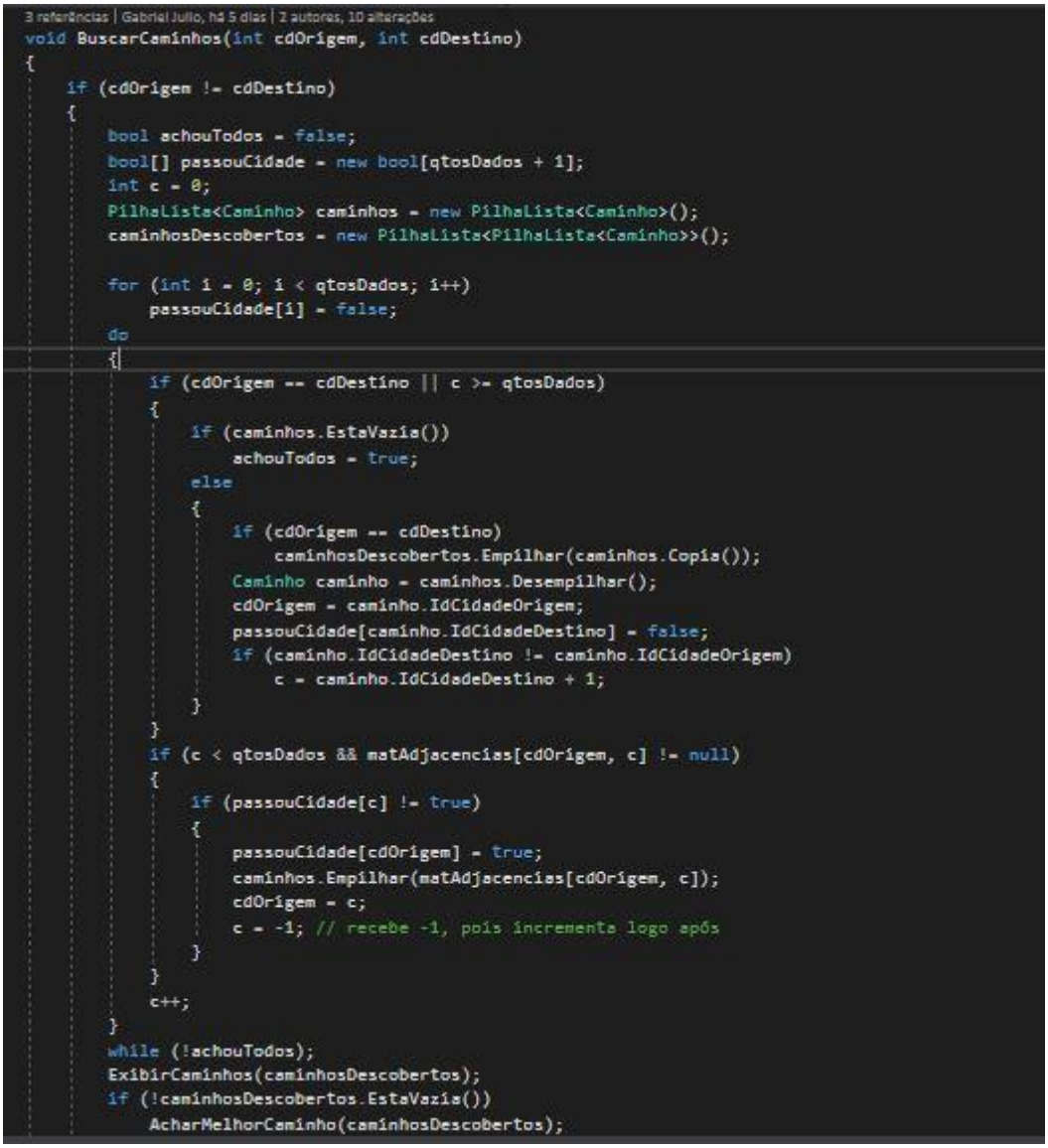
28/05 – Método de achar caminhos versão 1.

```
void BuscarCaminhos(int cdOrigem, int cdDestino)
{
    if (cdOrigem != cdDestino)
    {
        bool achouTodos = false;
        bool[] passouCidade = new bool[qtosDados];
        int c = 1;
        PilhaLista<Caminho> caminhos = new PilhaLista<Caminho>();
        PilhaLista<PilhaLista<Caminho>> caminhosDescobertos = new PilhaLista<PilhaLista<Caminho>>();

        for (int i = 0; i < qtosDados; i++)
            passouCidade[i] = false;
        do
        {
            if (caminhos.EstaVazia())
                achouTodos = true;
            if (cdOrigem == cdDestino || c == qtosDados)
            {
                if (cdOrigem == cdDestino)
                    caminhosDescobertos.Empilhar(caminhos);
                Caminho caminho = caminhos.Desempilhar();
                cdOrigem = caminho.IdCidadeOrigem;
                c = caminho.IdCidadeDestino + 1;
            }
            if (matAdjacencias[cdOrigem, c] != null)
            {
                if (passouCidade[c] != true)
                {
                    passouCidade[cdOrigem] = true;
                    caminhos.Empilhar(matAdjacencias[cdOrigem, c]);
                    cdOrigem = c;
                    c = 0; // recebe 0, pois incrementa logo após
                }
            }
            c++;
        }
        while (!achouTodos);
        ExibirCaminhos(caminhosDescobertos);
```

29/05 – Método de achar caminhos com algumas alterações.

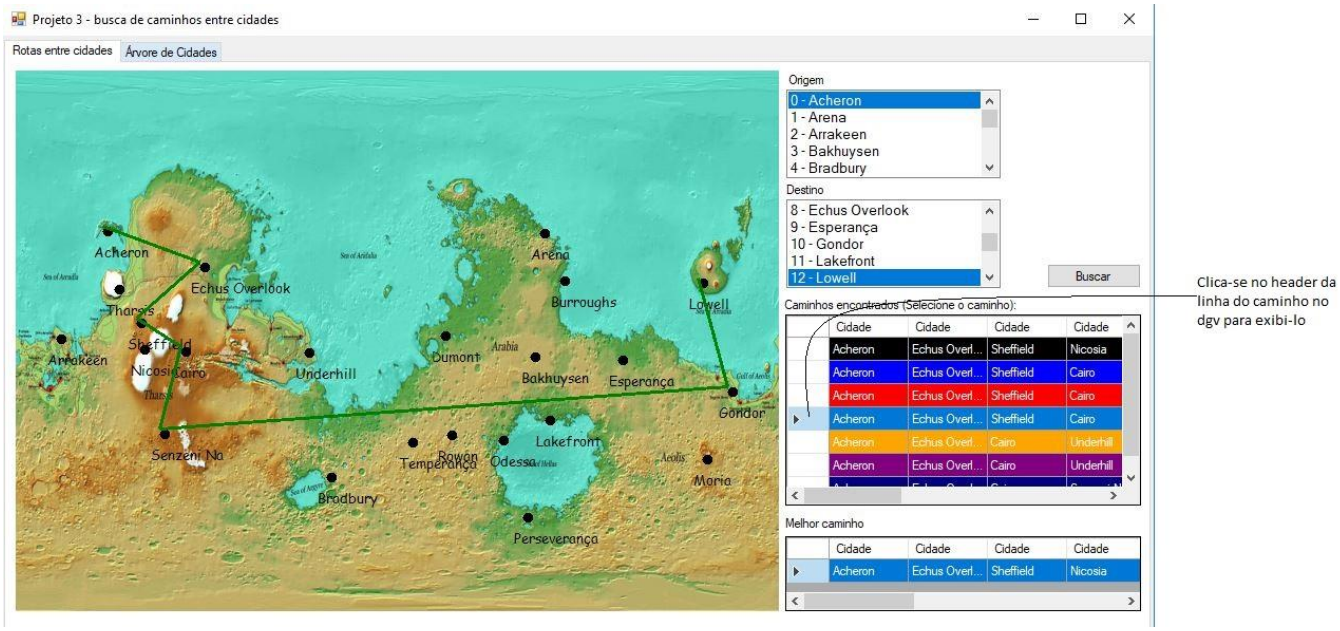
```
void BuscarCaminhos(int cdOrigem, int cdDestino)
{
    if (cdOrigem != cdDestino)
    {
        bool achouTodos = false;
        bool[] passouCidade = new bool[qtosDados];
        int c = 1;
        PilhaLista<Caminho> caminhos = new PilhaLista<Caminho>();
        PilhaLista<PilhaLista<Caminho>> caminhosDescobertos = new PilhaLista<PilhaLista<Caminho>>();

        for (int i = 0; i < qtosDados; i++)
            passouCidade[i] = false;
        do
        {
            if (caminhos.EstaVazia())
                achouTodos = true;
            else
            {
                if (cdOrigem == cdDestino)
                    caminhosDescobertos.Empilhar(caminhos.Copia());
                Caminho caminho = caminhos.Desempilhar();
                cdOrigem = caminho.IdCidadeOrigem;
                c = caminho.IdCidadeDestino + 1;
            }
            if (c < qtosDados && matAdjacencias[cdOrigem, c] != null)
            {
                if (passouCidade[c] != true)
                {
                    passouCidade[cdOrigem] = true;
                    caminhos.Empilhar(matAdjacencias[cdOrigem, c]);
                    cdOrigem = c;
                    c = 0; // recebe 0, pois incrementa logo após
                }
            }
            c++;
        }
        while (!achouTodos);
        ExibirCaminhos(caminhosDescobertos);
```

30/05 – Método de achar caminhos com as alterações citadas no desenvolvimento.



```
3 referências | Gabriel Julio, há 5 dias | 2 autores, 10 alterações
void BuscarCaminhos(int cdOrigem, int cdDestino)
{
    if (cdOrigem != cdDestino)
    {
        bool achouTodos = false;
        bool[] passouCidade = new bool[qtosDados + 1];
        int c = 0;
        PilhaLista<Caminho> caminhos = new PilhaLista<Caminho>();
        caminhosDescobertos = new PilhaLista<PilhaLista<Caminho>>();

        for (int i = 0; i < qtosDados; i++)
            passouCidade[i] = false;
        do
        {
            if (cdOrigem == cdDestino || c >= qtosDados)
            {
                if (caminhos.EstaVazia())
                    achouTodos = true;
                else
                {
                    if (cdOrigem == cdDestino)
                        caminhosDescobertos.Empilhar(caminhos.Copia());
                    Caminho caminho = caminhos.Desempilhar();
                    cdOrigem = caminho.IdCidadeOrigem;
                    passouCidade[caminho.IdCidadeDestino] = false;
                    if (caminho.IdCidadeDestino != caminho.IdCidadeOrigem)
                        c = caminho.IdCidadeDestino + 1;
                }
            }
            if (c < qtosDados && matAdjacencias[cdOrigem, c] != null)
            {
                if (passouCidade[c] != true)
                {
                    passouCidade[cdOrigem] = true;
                    caminhos.Empilhar(matAdjacencias[cdOrigem, c]);
                    cdOrigem = c;
                    c = -1; // recebe -1, pois incrementa logo após
                }
            }
            c++;
        }
        while (!achouTodos);
        ExibirCaminhos(caminhosDescobertos);
        if (!caminhosDescobertos.EstaVazia())
            AcharMelhorCaminho(caminhosDescobertos);
```

10/06 – Método de achar caminhos finalizado e com a alteração no passouCidade, igual citado na desenvolvimento.



Exemplo do funcionamento.

Conclusão

Após realizarmos o projeto, tivemos maior conhecimento do funcionamento de um aplicativo de rotas, pois sabíamos que o grafo era usado em aplicações do tipo, ma não como era usado. Assim, estudamos pela apostila essa estrutura de dados para a elaboração do projeto, ampliando nossos conhecimentos.

Além de termos aprendido sobre grafos, aprendemos muito sobre as classes de desenho no C#, como Graphics e Pen; e sobre o evento Paint dos componentes gráficos.

Treinamos recurssão em alguns métodos relacionados à árvore e como inserir registros lidos nessa estrutura.