UNIVERSIDADE FEDERAL DE SÃO CARLOS – UFSCAR

DEPARTAMENTO DE COMPUTAÇÃO

ESTRUTURAS DE DADOS

PROF. DR. DIEGO FURTADO SILVA.

DOCUMENTAÇÃO DO JOGO:
BINARY FRUIT TREE

ARIANE CRISTINA GOMES
743507 - arianecgomes@hotmail.com

THAÍS GAGLIARDO DORDAN
743596 – thais.dordan@hotmail.com

São Carlos
2018

1. Ficha Técnica

Nome do Jogo: Binary Fruit Tree.

Categoria: Aplicação de Árvore.

Ferramentas de Programação Utilizadas: Utilizamos a linguagem C++ em conjunto com a biblioteca SFML e compilamos no programa Code Blocks.

Descrição Breve do Jogo e do Modo de Jogar: Colete frutas e disponha-as nas barracas para formar uma árvore AVL.

Teclas de Comando: Setas do teclado e botão esquerdo do mouse.

Autores: Ariane Cristina Gomes e Thaís Gagliardo Dordan. Estudantes da Universidade Federal de São Carlos – UFSCar.

2. O Jogo

O jogo tem como objetivo a coleta de frutas que caem durante o jogo, para que sejam depositadas nas barracas próximas, estas que estão dispostas em forma de árvore binária de busca balanceada AVL.

Assim que o jogador insere as frutas na árvore binária de barracas, ele pode se deparar com situações como: inserção em posição errada e inserção de chave repetida, o que o leva ao fim do jogo. Além disso, o jogador também pode se deparar com possíveis rotações e ele deve escolher qual rotação deve ser feita. Ao escolher a rotação errada, ele pode perder o jogo.

Em contrapartida, se todas as inserções e rotações forem feitas com êxito, o jogador vence o jogo e pode jogar novamente após vencer ou perder.

3. Print Screens da Execução

O jogo inicia com a tela de menu, representada na Figura 1:

Figura 1: *Screenshot do menu do jogo.*

Nessa tela é possível obter as instruções de como jogar, através do botão Instruções. A tela de instruções pode ser observada em seguida, na Figura 2:

Figura 2: *Screenshot da opção Instruções.*

Ao jogador voltar ao menu e clicar no botão Jogar, é aberto o cenário do jogo, apresentado a seguir pela Figura 3.

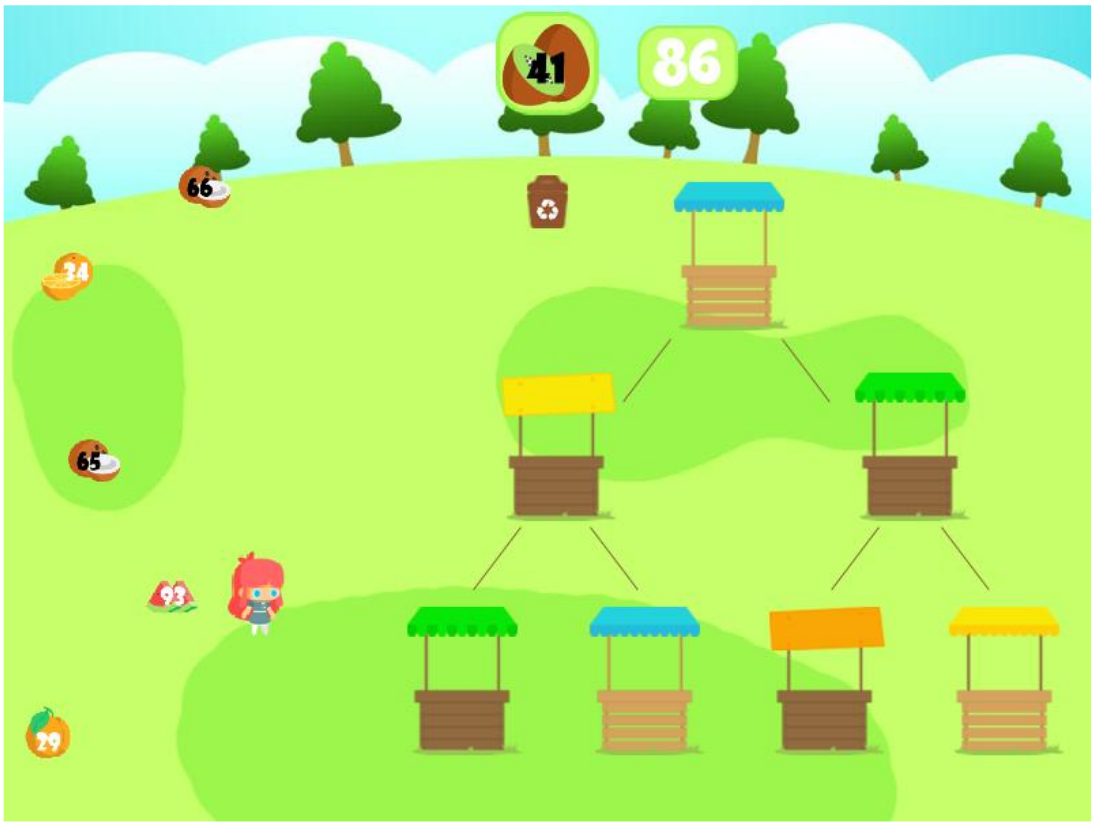

Figura 3: *Screenshot do cenário do jogo.*

De acordo com as frutas que caem, o jogador deve escolher algum número representado por estas e levá-las às barracas para que a árvore AVL seja preenchida de maneira correta.

Se houver necessidade de realizar uma rotação para completar a árvore, o jogador irá se deparar com as possíveis opções de rotação, conforme a Figura 4 ilustra.

Figura 4: *Screenshot das opções de rotação.*

Ao selecionar uma das opções, o jogo continua normalmente somente se a opção escolhida for a rotação que deve ser feita para manter a árvore balanceada.

Outrossim, após a inserção de todas as frutas de forma correta, respeitando o balanceamento da árvore, o jogador vence, possibilitando o mesmo a jogar novamente, como ilustra a Figura 5.

Figura 5: *Screenshot da tela "Ganhou".*

Se caso o jogador não conseguir montar a árvore de maneira correta, o aviso de que o mesmo perdeu aparece, como mostra a Figura 6, e ele também poderá jogar novamente. Os casos em que o jogador perde são: inserção em posição errada, inserção de chaves iguais, rotação feita de modo errado e limite de tempo excedido.

Figura 6: *Screenshot da tela "PerdeuPosicao".*

4. A Estrutura de Dados

A estrutura de dados utilizada no desenvolvimento do Binary Fruit Tree é uma Árvore Binária de Busca Balanceada do tipo AVL, utilizada nas barracas, para dispor uma ordem correta de inserção das frutas.

A cada momento que o jogador colide com uma das barracas, o jogo insere a fruta que o jogador está carregando consigo (disposta no canto superior direito da tela) nas árvores AVL "ArvoreJogador" e "ArvoreCerta".

A árvore "ArvoreJogador" possui inserções conforme o jogador coloca, sem se preocupar com regras de uma árvore AVL verdadeira. Já a "ArvoreCerta" faz todas as inserções e movimentações de uma árvore AVL correta. Sendo assim, após a inserção o jogo compara as duas árvores com o intuito de verificar se o jogador inseriu de forma semelhante à "ArvoreCerta". Em caso positivo, o jogo continua. Caso contrário, o jogador perde o jogo.

As rotações acontecem de forma semelhante, entretanto inicialmente são realizadas somente na "ArvoreCerta" e, após o jogador escolher a rotação que deseja fazer, o jogo verifica se a rotação escolhida foi a que se perpetuou em "ArvoreCerta". Caso seja, o jogo remove a "ArvoreJogador" inicial, cria uma nova igual à "ArvoreCerta" e o jogo continua. Caso contrário, o jogador perde o jogo.

5. Arquitetura de Software

Na arquitetura escolhida para esse jogo foi utilizada uma árvore, cuja implementação encontra-se nos arquivos "Arvore.h" e "Arvore.cpp".

Além disso, toda a interface do jogo pode ser encontrada nos arquivos "Game.h" e "Game.cpp", que faz uso da integração da árvore com os projetos gráficos da biblioteca SFML.

Ademais, o arquivo "main.cpp" é o que possibilita a iniciação do jogo em si.

6. Características da Árvore Binária

A árvore binária utilizada no jogo respeitou as regras de uma árvore binária de busca balanceada AVL com alocação dinâmica de memória, ou seja, a cada elemento inserido ou removido, um nó da árvore é, respectivamente, alocado ou desalocado, de acordo com a posição dos nós relacionados ao que sofrerá a alteração.

O objetivo é sempre preservar o balanceamento correto da árvore AVL, realizando rotações para que isso se cumpra, se necessário.

7. Trechos da Implementação

A seguir, pode-se observar a criação da função VerificaBalanceamento, que é utilizada na inserção e remoção de nós, para verificar se o balanceamento da árvore foi comprometido.

```
TipoArvore VerificaBalanceamento (TipoArvore *A) {
    int fb = FatorBalanceamento(&(*A));

    if (fb < -1) {

        if(FatorBalanceamento(&(*A)->Esq) > 0)
            (*A)->Esq = RotacaoSimplesEsquerda(&(*A)->Esq);

        (*A) = RotacaoSimplesDireita(&(*A));

    } else if (fb > 1) {

        if(FatorBalanceamento(&(*A)->Dir) < 0)
            (*A)->Dir = RotacaoSimplesDireita(&(*A)->Dir);

        (*A) = RotacaoSimplesEsquerda(&(*A));

    }

    return (*A);
}
```

Se caso o balanceamento esteja correto, a função retornará a árvore sem nenhuma alteração. Caso contrário, já realiza as rotações necessárias ao chamar as funções relacionadas à estas. Após o balanceamento ser dado como certo, a função irá retornar a árvore.

8. Atribuição de tarefas no grupo

O grupo procurou realizar todas as tarefas em conjunto, ou seja, as alunas Ariane Cristina Gomes e Thaís Gagliardo Dordan se encarregaram tanto da junção do design com a parte lógica e da aplicação da estrutura de dados no jogo em si, quanto da criação da estrutura de dados.

9. Conclusão

Ao executar o trabalho, aprendeu-se na prática qual a importância da estrutura de dados em uma codificação. Ainda, foi possível notar que o tipo de estrutura de dados auxilia na máxima portabilidade e reusabilidade da implementação feita, pois não foi necessário nenhum tipo de alteração para o jogo funcionar com êxito em diferentes computadores.

Além disso, com a experiência anterior na ferramenta de design SFML, tornou-se mais fácil a resolução de problemas causados pela mesma.

Ademais, o resultado foi satisfatório e agradou as ambas integrantes do grupo, pois, apesar de iminentes dificuldades ao implementá-lo, pôde-se aprender de forma prática e divertida um conceito muito importante.

10. Direitos Autorais

- Imagens retiradas do site: https://br.depositphotos.com/
- Músicas retiradas do site: https://freesound.org/