UNIVERSIDADE FEDERAL DE SÃO CARLOS – UFSCAR

DEPARTAMENTO DE COMPUTAÇÃO

ESTRUTURAS DE DADOS

PROF. DR. ROBERTO FERRARI JR.

DOCUMENTAÇÃO DO JOGO 1: COMPUTER RAIN

ARIANE CRISTINA GOMES
743507 - arianecgomes@hotmail.com

BRUNA FERNANDES PRATES
743513 - brunafpratess@gmail.com

ESTHER CALDERAN HOFFMANN
743529 - esthercalderan@gmail.com

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

São Carlos
2018

1. Ficha Técnica

Nome do Jogo: Computer Rain.

Categoria: Aplicação de Pilha.

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: Pegue peças de computador de acordo com a ordem da pilha à direita da tela.

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

Autores: Ariane Cristina Gomes, Bruna Fernandes Prates, Esther Calderan Hoffmann e Thaís Gagliardo Dordan. Estudantes da Universidade Federal de São Carlos – UFSCar.

2. O Jogo

Computer Rain é uma variação de uma aplicação de pilha, na qual peças de computadores caem em direção ao personagem e devem ser coletadas por este em uma ordem pré-determinada. Ao recolher os elementos corretamente, ganha-se o jogo. Caso ocorra a inserção de uma peça errada na pilha, não seguindo a disposição de objetos estipulada, o jogo é encerrado. A única exceção ocorre quando se armazena previamente uma estrela, a qual possui a função de permitir que um elemento pego erroneamente seja desempilhado imediatamente.

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:

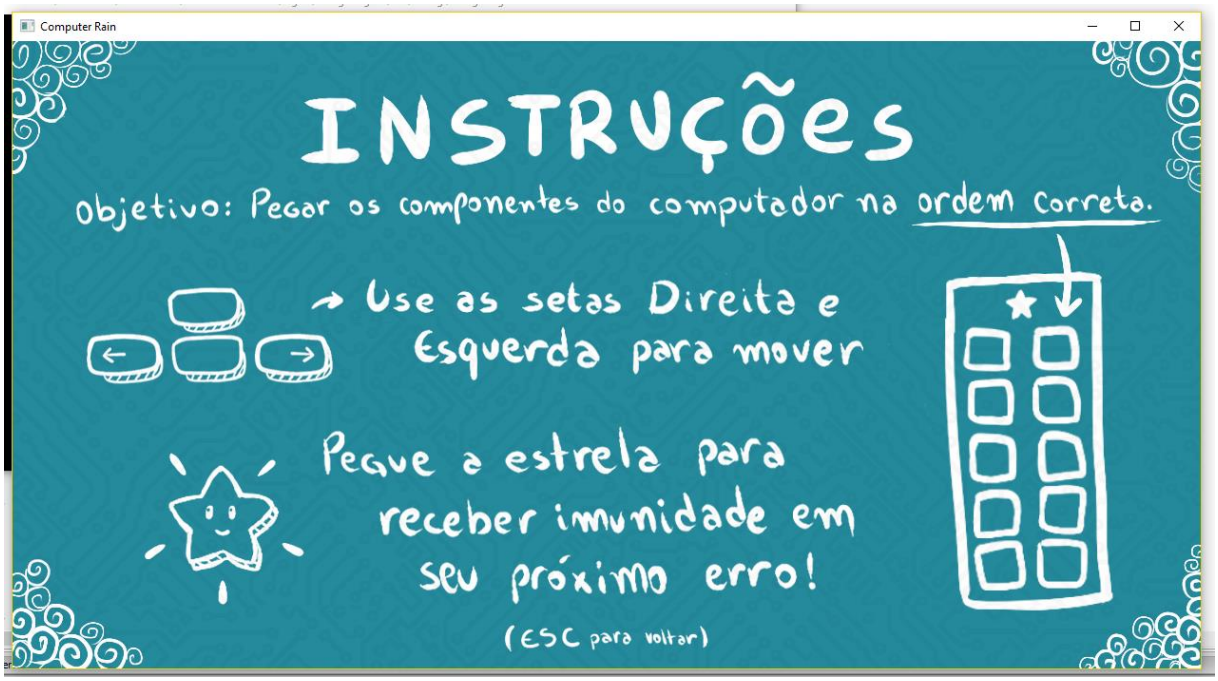

Figura 2: *Screenshot da opção Instruções.*

Ao jogador voltar ao menu e clicar no botão Jogar, é aberta a tela principal do jogo, Figura 3, onde as peças de computador caem pela tela e são também exibidas as pilhas P e PAux, que são, respectivamente, a pilha pré-determinada e a pilha a ser montada pelo jogador.

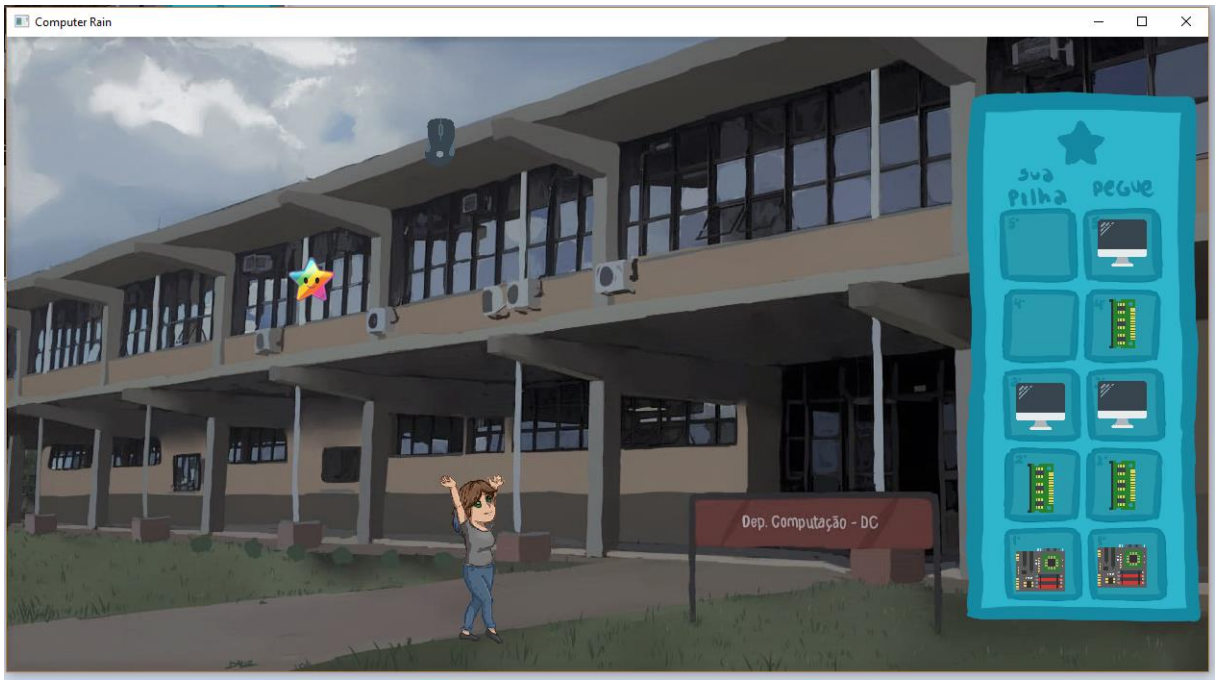

Figura 3: *Screenshot da tela principal do jogo.*

Caso o jogador pegue uma peça errada e não possua uma estrela para desempilhá-la, ele perderá o jogo e será redirecionado para a tela Perdeu, ilustrada na Figura 4:

Figura 4: *Screenshot da tela Perdeu.*

E por fim, caso o jogador consiga completar a pilha sem erros, ele ganhará o jogo e será redirecionado para a tela Ganhou, como mostra a Figura 5:

Figura 5: *Screenshot da tela Ganhou.*

4. A Estrutura de Dados

A estrutura de dados utilizada no desenvolvimento do Computer Rain é a Pilha, utilizada para a organização das peças que caem pela tela.

No jogo, são criadas as pilhas P e PAux, sendo P uma pilha pré-determinada com números aleatórios, para que o jogador empilhe peças em PAux de acordo com a sequência gerada para a pilha P.

A cada momento que o jogador pega uma peça, o jogo empilha a peça em PAux e compara se a peça obtida coincide com a peça de mesmo índice da pilha P. Sendo assim, caso a peça seja empilhada incorretamente e o jogador possuir uma estrela, é possível desempilhar de PAux o elemento colocado de forma equivocada.

5. Diagrama da Arquitetura de Software

O diagrama da arquitetura de software é determinado pela Figura 6, abaixo:

Figura 6: *Diagrama da Arquitetura de Software.*

Na arquitetura que foi desenvolvida, como se pode ver na imagem acima, foi utilizada a pilha “burra” sendo Pilha.h, arquivo que consta a struct e as funções primitivas da pilha.

Já na pilha “inteligente”, denominada TAD.h, são definidas as funções para manipulação da pilha durante a implementação.

Além disso, toda a interface do jogo pode ser encontrada no arquivo Logic.h, que faz uso somente da pilha inteligente.

6. Diagrama da Implementação da Pilha

Na Figura 7, pode-se observar o diagrama da implementação da pilha:

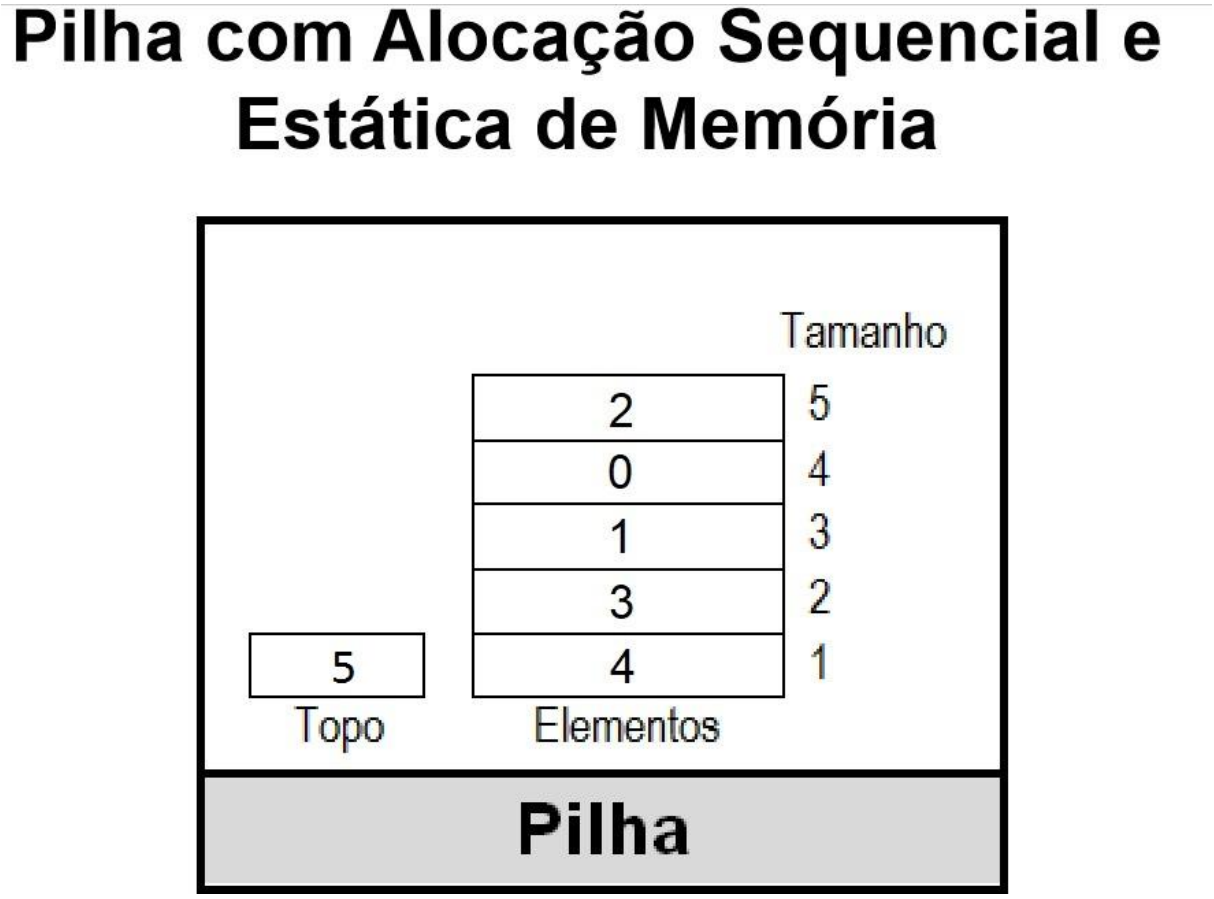

Figura 7: *Diagrama da Implementação da Pilha.*

A pilha utilizada no jogo respeitou as regras de uma pilha com alocação sequencial e estática de memória, ou seja, existe um P.Topo que aponta para o último índice da pilha, sendo assim, somente é possível desempilhar P.Topo (e atualizá-lo quando isso ocorre) e empilhar em índice maior que P.Topo (se houver espaço na pilha para a inserção de mais um elemento).

7. Trechos da Implementação

Abaixo, pode-se observar a criação da função Iguais, que retorna verdadeiro quando há êxito na comparação de dois elementos do mesmo índice da pilha P (pilha fixa) e da pilha PAux (pilha do jogador), além de também retornar positivamente quando o jogador possuir uma estrela no ato de pegar a peça. Em contrapartida, retornará falso quando o jogador pegar uma peça e ela não ser igual à correspondente na pilha pré-determinada.

```
    bool Iguais(Pilha &P, Pilha &PAux, bool &Estrela, bool &gameOver,
int &winner) {
    int i, X = 0, XAux = 0, cont = 0;
    Pilha P1, P2;
    bool DeuCerto, DeuCertoAux, DeuCerto1, DeuCerto2;

    Cria(P1);
```

```
    Cria(P2);

    while(!Vazia(P)) {
        Desempilha(P, X, DeuCerto);
        Empilha(P1, X, DeuCerto1);
    }

    while(!Vazia(PAux)) {
        Desempilha(PAux, XAux, DeuCertoAux);
        Empilha(P2, XAux, DeuCerto2);
        cont++;
    }

    for(i = 0; i < cont; i++) {
        Desempilha(P1, X, DeuCerto1);
        Empilha(P, X, DeuCerto);
    }

    while(!Vazia(P2)) {
        Desempilha(P2, XAux, DeuCerto2);
        Empilha(PAux, XAux, DeuCertoAux);
    }

    if(DeuCerto && DeuCertoAux){
        if(X != XAux){
            if(Estrela){
                Estrela = !Estrela;
                Desempilha(PAux, XAux, DeuCerto);
                winner--;
                if(DeuCerto) {
                    while(!Vazia(P1)) {
                        Desempilha(P1, X, DeuCerto1);
                        Empilha(P, X, DeuCerto);
                    }
                    return true;
                }
            }
        }
        else {
            while(!Vazia(P1)) {
                Desempilha(P1, X, DeuCerto1);
                Empilha(P, X, DeuCerto);
            }
            return true;
        }
    }
    gameOver = true;
    return false;
}
```

Para isso ocorrer e para que haja sempre a melhor reusabilidade e portabilidade, é necessário desempilhar a pilha fixa e a pilha do jogador e empilhar os elementos das mesmas em pilhas auxiliares P1 e P2, para que os elementos não sejam perdidos.

Quando um elemento da pilha do jogador é desempilhado, é acionado um contador para termos o controle de qual elemento devemos comparar em P, sendo assim, foi empilhado de novo em P e em PAux até que P2 seja vazia, ou seja, os índices dos elementos de P e PAux serão iguais.

A partir disso, os elementos são comparados e faz-se a verificação da existência de estrela, como já mencionado. Além disso, são atualizadas as variáveis winner e gameOver, que correspondem a situação do jogo a partir do resultado da função.

Em ambos os casos de retorno, o restante da pilha fixa é empilhado até que a pilha fique cheia novamente, com os mesmos elementos.

8. Atribuição de tarefas no grupo

O grupo procurou realizar todas as tarefas em conjunto, entretanto, as alunas Ariane Cristina Gomes e Thaís Gagliardo Dordan se encarregaram majoritariamente da lógica com a estrutura de dados e de sua implementação.

Em contrapartida, as alunas Bruna Fernandes Prates e Esther Calderan Hoffmann foram as responsáveis pelos trabalhos referentes ao design e suas aplicações no jogo.

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.

Em contrapartida, sentimos muita dificuldade em trabalhar com a biblioteca gráfica SFML para implementação em C++, pois notou-se que há pouco material disponível para a consulta de possíveis problemas relacionados à biblioteca.

Ademais, o resultado foi satisfatório e agradou todas as 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

- Músicas retiradas do site: https://freesound.org/