

**Universidade Federal de São Carlos
Departamento de Computação
Estrutura de Dados**

Documentação trabalho 3

Renata Sarmet Smiderle Mendes, 726586, renatassmendes@hotmail.com.

Rodrigo Pesse de Abreu, 726588, abreu.rodriigo97@gmail.com.

Rafael Bastos Saito, 726580, rafaelbsaito@hotmail.com.

1. Funcionamento do jogo

O jogo PokeBallz, basicamente, consiste em um plano que gradativamente aumenta o número de ovos presentes na tela, concomitantemente ao número de rodadas. Ou seja, ao iniciar o jogo, na parte mais baixa da tela, terá entre 0 a 8 ovos distribuídos aleatoriamente, podendo possuir uma poke ball entre eles. Conforme cada rodada é jogada, todas as fileiras de ovos elevam-se um nível acima.

Na parte superior da tela, o Ash estará movimentando seu braço podendo disparar pokeballs ao comando do usuário.

- **Cenário:** O cenário é composto por vários poke eggs, que aparecem concomitante ao número de rodadas. Ou seja, a cada rodada jogada, uma fileira de ovos será acrescentada, no nível mais inferior da tela, contendo de 0 a 8 ovos. Na parte mais superior há um pêndulo movimentando-se, destinado ao lançamento de poke balls.



Foto Cenário

- **Poke Eggs:** Os poke eggs aparecem no plano conforme cada rodada é ultrapassada. Eles aparecem com um índice em seu centro, representando quantas vezes a poke ball deverá colidir com ele para o mesmo desaparecer.



Poke Egg

- **Rodada:** Ela é encerrada quando todas as poke balls disponíveis ,naquela rodada, já tiverem sido disparadas ou todos os poke eggs presentes estiverem sido destruídos.
- **Poke Balls:** São lançadas do nível mais alto do plano. Possuem o intuito de colidir com os poke eggs dispostos pelo plano. As poke balls destruídas no plano, quando conquistadas pelo player, possuem o intuito de acrescentar a quantidade de lançamentos disponíveis a partir da próxima rodada.



Poke Ball

A sequência de telas até o início do jogo é:

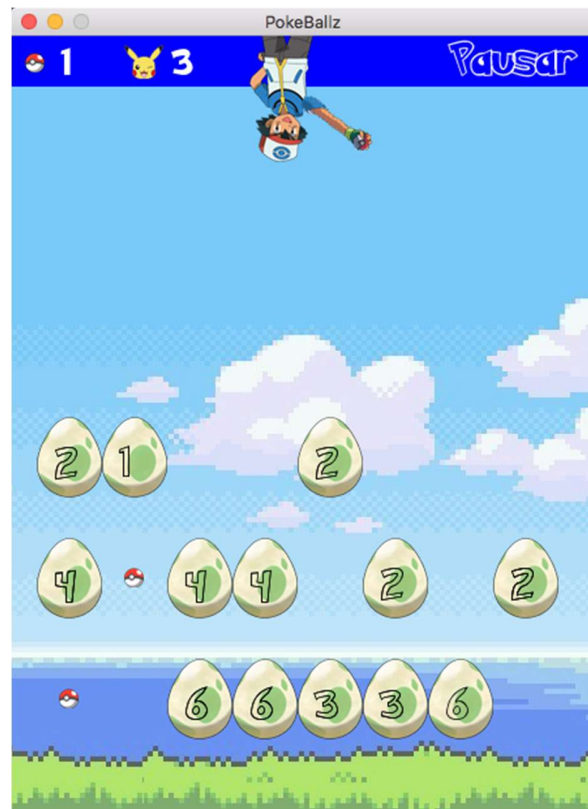


Clique para iniciar

Tela Inicial



Tela opções iniciais



Jogo em andamento

A sequência de instruções presentes no jogo é:



Primeira Instrução



Segunda Instrução



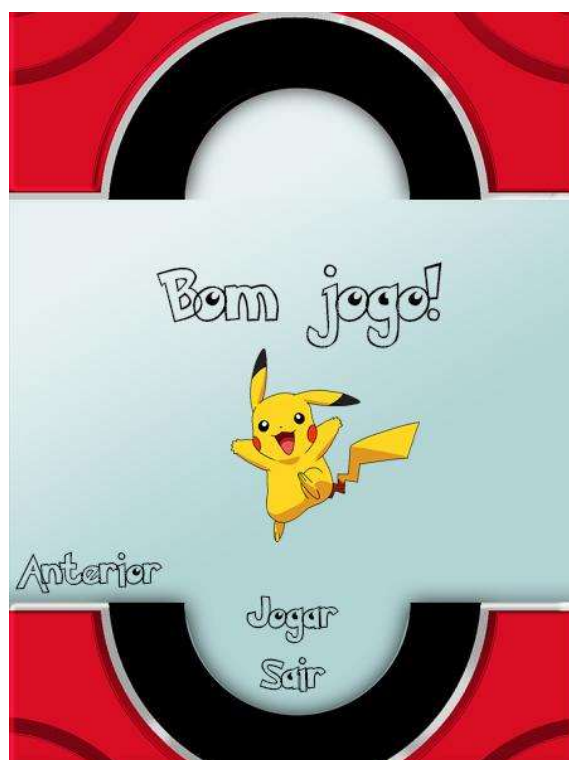
Terceira Instrução



Quarta Instrução



Quinta Instrução



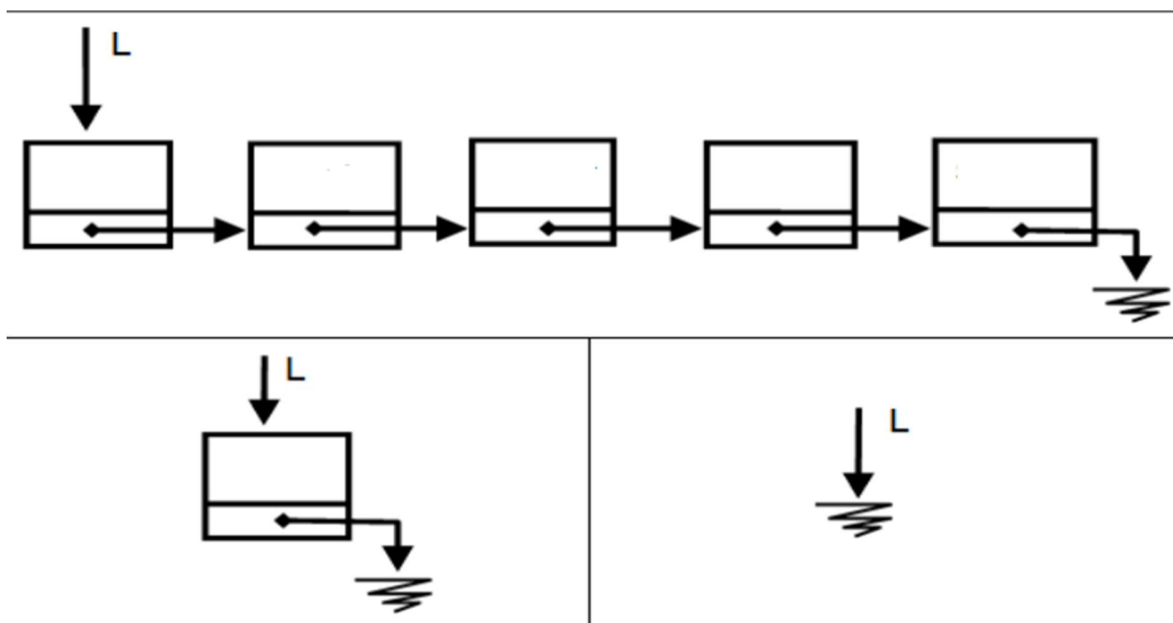
Sexta Instrução

2. Estruturação do código

Para a implementação do jogo foram utilizadas a estrutura de Lista, vista em sala, implementada em C++.

- **Lista Simples Encadeada**

Foi utilizado a estrutura de dados Lista para manipulações necessárias dentro do jogo, em especial no plano.



Lista Geral

Foi criada a classe Lista Simples para representar os métodos de uma lista comum.

```

12 class ListaSimples
13 {
14 public:
15     ListaSimples();
16     ~ListaSimples();
17
18     void Cria(Nodetype *x);
19     bool Vazia();
20     void Insere(Nodetype *x);
21     void InsereADireita(Nodetype *x);
22     void ProcuraRemove(int x, bool &DeuCerto);
23     void DeletaTudo();
24
25     Nodetype* PegaElementoN(int n) const; // pega o N-esimo elemento da lista
26     int QuantidadeElementos() const; // conta a quantidade de elementos da lista
27
28     Nodetype *P; // Indica sempre o primeiro da Lista
29
30 private:
31
32     void Remove(Nodetype *Premove, bool &DeuCerto); // metodo privado pois quem deve ser chamado eh o procura remove
33 };

```

Classe Lista Simples

Também foi criada a classe Plano, subclasse de Lista Simples, e é responsável por manipular os poke eggs presentes no jogo.

```

14 class Plano : public ListaSimples{
15 public:
16     Plano();
17     ~Plano();
18     void InsereNplano(int rodadaAtual);
19     void desenha_todos_plano(sf::RenderWindow& window);
20     bool get_perdeu() const;
21     void reseta_perdeu_false();
22 private:
23     int contador;
24     int quantidadeJaInseridaID;
25     bool perdeu;
26     bool jaTeveNovaPokeball;
27 };

```

Classe Plano

A principal diferença dessa classe e de uma lista normal é a inserção. Quando uma nova rodada é iniciada, todos os elementos da lista, ainda presentes, devem subtrair um valor às suas variáveis "y", fazendo-as aparecer um nível acima no plano. Em seguida, aleatoriamente, é preenchido a linha mais abaixo com novos poke eggs e/ou poke ball. Esse método também é responsável por definir o fim do jogo. Quando os elementos do plano sobem um nível acima, é comparado com o limite do plano e, se o poke egg alcançar esse limite, o game over é decretado.

```

43 void Plano::InsereNplano(int rodadaAtual)
44 {
45     int i,entraOuNao, r, valor;
46     bool DeuCerto;
47     float _x, _y;
48     Nodetype *no;
49     no = P;
50
51     jaTeveNovaPokeball = 0;
52     while(no != NULL){
53         no->valorText.setString(to_string(no->get_valor()));
54         no->set_posicao(no->get_x(),no->get_y()-93);
55         no->valorText.setPosition(no->get_x(),no->get_y());
56
57         if(no->get_y()<=LIMITE_PLANO){
58             if(no->get_valor()==-1)
59                 ProcuraRemove(no->get_id(), DeuCerto);
60             else{
61                 DeletaTudo();
62                 perdeu = 1;
63             }
64         }
65         no = no->get_next();
66     }
67

```

Método InsereNPlano da classe Plano - parte 1

```

68     if(!perdeu){
69         for(i=0;i<TAMANHO;i++){
70             entraOuNao = rand() % 2;
71             if(entraOuNao==1){
72                 r = (rand() % 2 + 1);
73                 valor = rodadaAtual*r; // valor a ser inserido
74                 quantidadeJaInseridaID +=1;
75                 no = new Nodetype();
76                 no->carregar("imagens/ovo.png");
77                 no->set_id(quantidadeJaInseridaID);
78                 no->set_valor(valor);
79                 no->valorText.setString(to_string(no->get_valor()));
80                 _y = 510; // valor correspondente à primeira linha de ovos
81                 _x = i * 50 + 45; // i * (distancia entre um ovo e outro) + (centro do primeiro ovo)
82                 no->set_posicao(_x, _y);
83                 if(no->get_valor() >= 20)
84                     no->valorText.setPosition(no->get_x(), no->get_y());
85                 else
86                     no->valorText.setPosition(no->get_x(),no->get_y());
87
88                 Insere(no);
89             }
90         }
91         else{
92             int a;
93             a = (rand() % 3);
94             if(!jaTeveNovaPokeball && a == 1){
95                 jaTeveNovaPokeball = 1;
96             }
97         }
98     }
99 }

```

Método InsereNPlano da classe Plano - parte 2

```

95
96     valor = -1;
97     quantidadeJaInseridaID +=1;
98     no = new Nodetype();
99     no->carregar("imagens/pokebola.png");
100    no->set_id(quantidadeJaInseridaID);
101    no->set_valor(valor);
102    no->valorText.setString("");
103    _y = 510; // valor correspondente à primeira linha de ovos
104    _x = i * 50 + 45; // i * (distancia entre um ovo e outro) + (centro do primeiro ovo)
105    no->set_posicao(_x, _y);
106    no->valorText.setPosition(no->get_x(),no->get_y());
107
108    Insere(no);
109 }
110 }
111 }
112 }
113 }

```

Método InsereNPlano da classe Plano - parte 3

- **Nó**
A classe Nodetype foi implementada para compor a lista.

```

12 class Nodetype
13 {
14 public:
15     //CONSTRUTOR
16     Nodetype();
17
18     //DESTRUTOR
19     ~Nodetype();
20
21     //FUNCOES
22     virtual void carregar(std::string nomearquivo);
23     virtual void desenhar(sf::RenderWindow& renderWindow);
24     virtual bool colidiu(pokeball& _pokeball);
25
26     float get_x() const;
27     float get_y() const;
28
29     void set_posicao(float _x, float _y);
30
31     Nodetype* get_next() const;
32     void set_next(Nodetype *);
33     int get_id() const;
34     void set_id(int);
35     void CopiaNode(Nodetype *original);
36     void set_valor(int);
37     int get_valor() const;
38

```

Classe Nodetype - parte 1

```

39     sf::Sprite _sprite;
40     static sf::Text valorText;
41
42 private:
43     Nodetype *next;
44     int id;
45     int valor;
46     bool carregou;
47
48     sf::Font fonte;
49     sf::Texture _imagem;
50     std::string nome_arquivo;
51 };

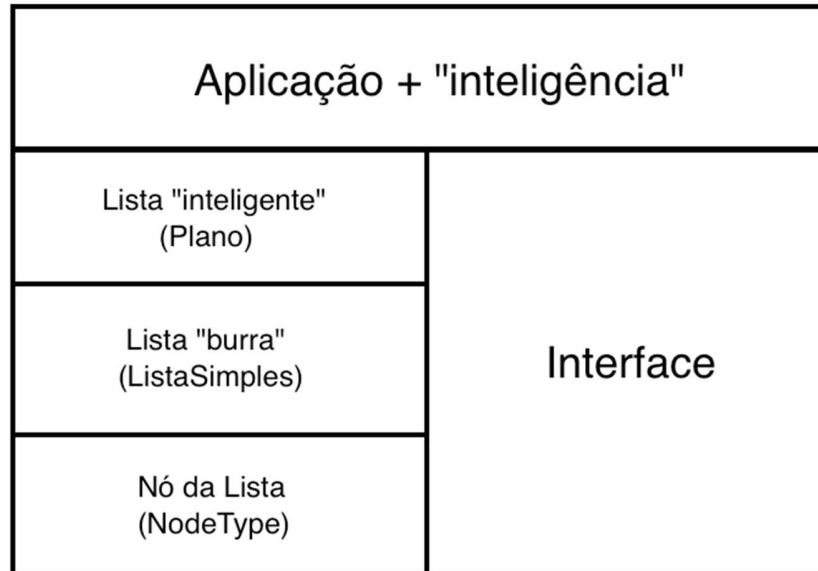
```

Classe Nodetype - parte 2

Quando a pokeball atinge um Poke Egg (nó da lista), seu valor é diminuído, até chegar em 0. Quando chega em 0, ele é removido da lista Plano.

3. Diagramas

a. Diagrama da arquitetura do software



b. Diagrama de classes do software

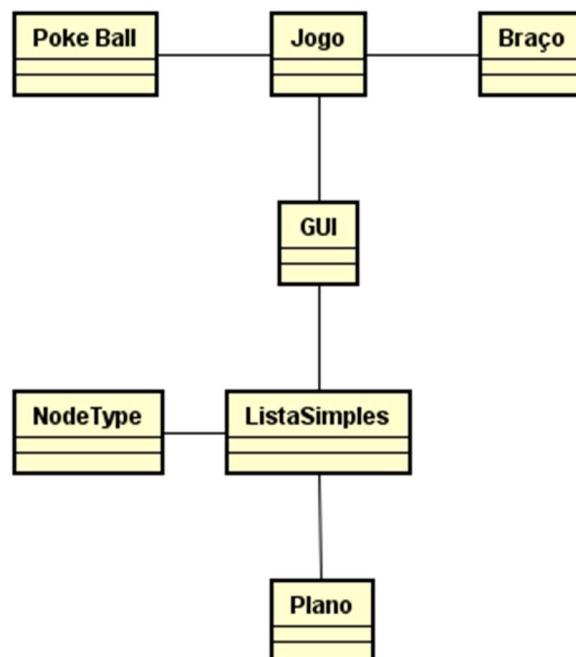


Diagrama de Classes

4. Conclusão

A conclusão que o grupo teve após a realização do terceiro trabalho da disciplina de Estrutura de Dados, ministrada pelo professor Roberto Ferrari, foi muito satisfatória.

O desenvolvimento do jogo foi mais rápido que os outros dois, pois foi utilizado vários conceitos e estruturas do primeiro e segundo trabalho catalisando seu desenvolvimento.

A distribuição entre o grupo para a execução do projeto, foi repartido de tal maneira: Renata e Rodrigo com a parte do desenvolvimento do código, implementação das listas, etc; Renata e Rafael com o desenvolvimento do sfml; Rafael com a parte de design.

O grupo gostou bastante de desenvolver o trabalho, pois abrangeu fatores como: criatividade, lógica de programação, design e logística.

Como última conclusão, o grupo agradece o professor por apresentar um método de estudos eficiente e atrativo para os alunos.