Relatório do

Trabalho de Programação Avançada 2019/2020

Fase 2

Marco Domingues- 2018013362

Índice

1- Descrição sintética acerca das opções e decisões tomadas na implementação

Parte Lógica

O programa contém várias classes base que representam objectos ou partes dos mesmos, como por exemplo a classe nave, que representa uma nave e contém todos os seus dados e a classe Recursos que representa os recursos que uma nave / planeta têm.

Os dados do jogo representados de forma fragmentada pelas classes base são centralizados na classe Jogo, essa classe guarda quase todos os dados do jogo, sendo que não guarda os dados sobre a posição dos objetos no estado de estadia no planeta, que se encontram no estado. Essa excepção acontece uma vez que é um resíduo da primeira fase de entrega do trabalho prático e, como mais nada interfere com aquele estado (ao contrário dos outros estados), decidi não criar problemas inesperados, ainda que vá contra a estrutura do programa.

Os estados do programa têm uma Interface base, chamada de IEstado, esta interface contém as funções para execução dos estados. A mesma é implementada pela classe EstadoAdapter, que é uma adaptador de estados e que tem como objetivo fazer uma implementação inicial dos estados de maneiras a que as classes estado o possam extender e usar as funções que lhes forem favoráveis.

A junção entre os estados e dados do jogo é feita pela classe MaqEstados, a máquina de estados que serve de ligação entre a interface gráfica e a parte lógica do programa.

Parte Gráfica

A execução deste programa começa na classe interface, esta classe começa por criar uma máquina de estados, onde é iniciada a parte lógica do programa. Essa maquina de estados é enviada para a classe JogoObservavel, dando também a essa classe posse dos dados. Esta classe vai ser responsável por atualizar as vistas do jogo, para tal, a mesma tem as funções que os componentes da parte gráfica deverão chamar e, para além disso, *extend* a classe PropertyChangeSupport, para o efeito de atualização dos ecrãs.

Depois da criação das classes acima, a classe Interface ainda cria, antes de criar a primeira janela, um PaneOrganizer. esta classe é responsável por fazer ligação entre a interface e a classe Base, onde são definidos os traços visuais base do ambiente gráfico.

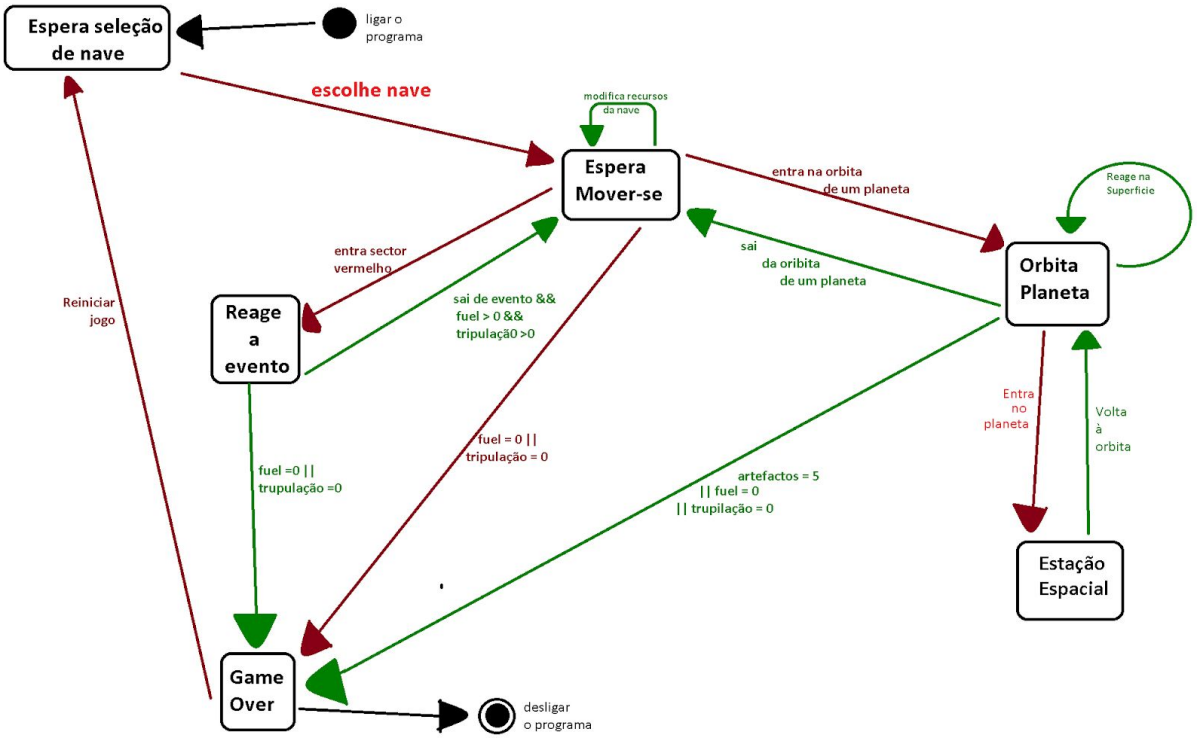
A classe Base trata de colocar o background que esteja de acordo com o estado de jogo em que se está, de criar a MenuBar que contêm as opções de salvar o jogo e dar *load* a um jogo e a classe BasePane.

A classe BasePane é responsável por dividir o ecrã em duas secções, uma secção central para que apareçam a interface gráfica relacionada com os estados do jogo e uma seção, no fundo do ecrã, responsável por enviar algumas imagens informativas ao utilizador. Essas mesmas mensagens são atualizadas nesta classe.

As várias classes que representam graficamente um estado do jogo são responsáveis por criar e posicionar os seus botões, assim como indicar aos mesmos o que fazer quando o botão é premido.

Voltando para a classe Interface, é também lá criada uma segunda janela. Esta janela é constituída pela classe List Root. Essa classe tem como função mostrar em todos os momentos ao utilizador os dados da nave e permitir que o utilizador faça gestão dos recursos que têm ao seu dispor. Dessa gestão de recursos é necessário __ a capacidade de haver trocas internas de recursos. A mecânica utilizada para tal funcionalidade passa por utilizar a classe EditarRecursos, que se trata de uma nova janela que é aberta, lá, o utilizador pode indicar que recurso trocar por qual de forma a que seja feita a troca.

2- Diagrama de classes



3- Classes usadas no programa

LandingCraft:
Serve para simular o drone de exploração que é usado na busca de recursos nos planetas.

Naves:
Serve para simular uma nave. As características da nave adequam-se ao tipo de nave escolhida no início do programa.

Officer:
Serve para simular um oficial da nave. Esta classe guarda o cargo do oficial e se este se encontra vivo.

Planeta:
Serve para simular um planeta, este tem uma cor aleatória e características conforme a sua cor.

Posicao:
Serve para guardar as posições dos aliens, recursos e drone aquando o programa está na fase de mineração.

Recursos:
Serve para guardar as quantidades dos vários recursos que várias entidades têm ao longo do jogo.

SpaceStation:
Serve para indicar que tipo de ações podem ser feitas/já foram feitas numa estação espacial

EsperaMover:
Serve para simular o estado “espera mover”, onde o jogador pode decidir entre avancar no espaço, ver dados da nave ou fazer modificações nos recursos internos da nave.

EstadoAdapter:
Serve como adaptador de estados, criando funções usáveis em todos os estados que o estendam.

FimDeJogo:
Estado que simula o final do jogo, este indica se o jogador ganhou ou perdeu e se quer desligar o programa ou jogar de novo.

IEstado:
Estado base de todos os outros estados.

MaqEstados:
classe que gere o chamamento de outros estados.

MercadoNaEstacaoEspacial:
Estado que representa a estadia da nave numa estação espacial, sendo que aqui é possível trocar recursos por outros itens cuja disponibilidade é baseada nas regras do jogo.

OrbitaPlaneta:
classe que representa o estado onde a nave estaria na órbita de um planeta, é responsável, em certas circunstâncias, por criar um planeta e estação espacial.

ReageEvento:
Classe que tem como objetivo simular os efeitos de um evento aleatório.

ReageSuperficie:
Classe que pretende representar a parte de mineração do jogo. Apesar de ser uma classe, esta não tem qualquer interferência com o jogador conforme as novas regras jogo. A sua existência é anterior à partilha das novas regras.

SelectVeicle:
Classe que representa o estado inicial do jogo, onde o jogador escolhe a sua nave.

Base:
Classe utilizada para criar a base da janela principal do jogo, inclui as MenuBars e as imagens de fundo.

BasePane:
Classe utilizada para criar a base do ecrã principal do jogo, todas as classes de interface gráfica que são ativadas aquando se esta em determinado estado, esta classe faz as "divisões" invisíveis no ecrã e mostra também o log.

EscolheNavePane:
Classe gráfica que representa no ecrã o estado de escolha de nave.

EsperaMoverPane:
Classe gráfica que representa no ecrã o estado de espera mover.

EstacaoEspacialPane:
Classe gráfica que representa no ecrã o estado existente quando se esta numa estação espacial.

FimDeJogoPane:
Classe gráfica que representa no ecrã o estado existente quando está no final do jogo, após ganhar ou perder.

MostrarRecursos:
Classe gráfica que abre uma nova janela onde o utilizador pode ver os dados sobre o planeta em que se encontra.

OrbitaPlanetaPane:
Classe gráfica que representa o estado em que o jogador está na órbita do planeta.

ReageEventoPane:
Classe gráfica que representa o estado em que o jogador passa por um evento.

Imagens:
Classe que tem como função importar imagens para o jogo.

Constantes:
Interface que tem como função guardar alguns valores.

EditarRecursos:
Classe gráfica que cria uma nova janela onde o jogador pode transformar os recursos que estão dentro da nave, conforme as regras do jogo.

Interface:
Classe onde o código do jogo começa a correr, responsável por criar as duas janelas existentes e chamar as classes que as permitem completar.

JogoObservavel:
Classe responsável por atualizar as várias janelas quando alguma mudança é feita no jogo.

ListRoot:
Classe responsável por representar a segunda janela de jogo, onde mostra todos os dados da nave e permite fazer alguma ações internas à mesma.

PaneOrganizer:
Classe que serve como ponte entre o main e a classe Base.

4- Relações entre classes

Interface**:**

Inclui como variáveis:
UItexto

UItexto**:**

Inclui como variáveis:
MaqEstados

MaqEstados:

Inclui como variáveis:
IEstado

Jogo:

Inclui como variáveis:
Planeta
Naves
SpaceStation

Naves:

Inclui como variáveis:
Officer
Recursos
LandingCraft

Planeta:

Inclui como variáveis:
Recursos

EsperaMover:

Extends classe:
EstadoAdapter

EstadoAdapter:

Implements classe:
Estado
Inclui como variáveis:
Jogo

FimDeJogo:
Extends classe:
EstadoAdapter

MercadoNaEstacaoEspacial
Extends classe:
EstadoAdapter

OrbitaPlaneta:
Extends classe:
EstadoAdapter

ReageEvento:
Extends classe:
EstadoAdapter

ReageSuperficie:
Extends classe:
EstadoAdapter

SelectVeicle:
Extends classe:
EstadoAdapter

EscolheNavePane:
Extends classe:
HBox
Implements classe:
Constantes
PropertyChangeListener
Inclui como variáveis:
JogoObservavel

EsperaMoverPane:
Extends classe:
HBox
Implements classe:
Constantes
PropertyChangeListener
Inclui como variáveis:
JogoObservavel

EstacaoEspacialPane:

Extends classe:
VBox
Implements classe:
Constantes
PropertyChangeListener
Inclui como variáveis:
JogoObservavel

FimDeJogoPane:

Extends classe:
HBox
Implements classe:
Constantes
PropertyChangeListener
Inclui como variáveis:
JogoObservavel

MostrarRecursos:

Extends classe:
Stage
Implements classe:
Constantes
Inclui como variáveis:
JogoObservavel

OrbitaPlanetaPane:

Extends classe:
HBox
Implements classe:
Constantes
PropertyChangeListener
Inclui como variáveis:
JogoObservavel

ReageEventoPane:

Extends classe:
HBox
Implements classe:
Constantes
PropertyChangeListener
Inclui como variáveis:
JogoObservavel

Base:

Extends classe:
VBox
Implements classe:
Constantes
PropertyChangeListener
Inclui como variáveis:
JogoObservavel
BasePane

BasePane:

Extends classe:
BorderPane
Implements classe:
Constantes
PropertyChangeListener
Inclui como variáveis:
JogoObservavel
EscolheNavePane
EsperaMoverPane
EstacaoEspacialPane
FimDeJogoPane
OrbitaPlanetaPane
ReageEventoPane

EditarRecursos:

Extends classe:
Stage
Implements classe:
Constantes
Inclui como variáveis:
JogoObservavel
BasePane

Interface:

Extends classe:
Application
Implements classe:
Constantes
Inclui como variáveis:
MaqEstados
JogoObservavel
PaneOrganizer

JogoObservavel:

Extends classe:
PropertyChangeSupport
Implements classe:
Constantes
Inclui como variáveis:
JogoObservavel

ListRoot:

Extends classe:
BorderPane
Inclui como variáveis:
JogoObservavel

5- Funcionalidades

Funcionalidades implementadas:

Escolha de nave;
Diferentes tipos de nave;
Planetas e as suas características;
Troca de recursos dentro da nave;
Compra com recursos nas estações espaciais;
Eventos, tanto aleatórios como escolhidos pelo user;
WormHoles;
Mineração automática dos planetas;
Luta automática com os aliens encontrados no planeta;
Logger que permite ver os dados do jogo
Interface gráfica.

Funcionalidades não implementadas:

-

6- Notas finais:

Durante a execução do programa não foram encontrados nenhuns erros no programa.

Não foi implementada uma parte gráfica para o estado de reação no planeta uma vez que este é um estado robotizado e não fazia sentido que o jogador fica-se a ver o computador jogar, para além disso, tendo em conta que este é um trabalho universitário e não para um público real, a criação desta interface gráfica não acrescentaria nada ao mesmo.

Por fim, o ambiente usado para a criação do projeto foi eclipse, o mesmo usado nas aulas práticas, sendo que, caso haja algum problema em relação à passagem para o ambiente netbeans, assim como em qualquer outro problema ou dúvida, estarei disponível para resolver os mesmos.