Trabalho POO – FracFly:
Integrantes:

André Felipe da Silva
Eliézer Ribeiro
Mário Saldanha
Lúcio Leal Bastos

Documentação:
==========

Desenvolvedor:
==========
Compilaçao:

Nosso jogo utiliza várias bibliotecas de terceiros, algumas podem ser obtidas diretamente por APT-GET outras necessitam de compilacao;

Utilizando os comandos:

make pacotes - instala todas bibliotecas SDL disponiveis no repositorio (distribuicao ubuntu)

make dependencias - mostra uma lista de dependencias que precisam ser compiladas manualmente;

Cada um dos comandos abaixo mostra os passos para instalacao, em ordem de dependencias que precisam ser compiladas manualmente:

make depM4
make depAutoconf
make depAutomake
make depFreetype
make depSDL_ttf
make depSDL_mixer

Apos todas dependencias estarem instaladas:

make

o jogo pode ser executado atraves do comando

./fracfly

Fluxo de execução:
==========
Através da classe FracFly, é criado um objeto oJogo, a classe FracFly, possui o seguinte fluxo:

NoInic: Inicializa bibliotecas do SDL, como SDL_Init e SDL_TTF e inicializa variáveis que necessitam ser inicializadas antes da criação do plano de exibição;

NoEvento: Controla os eventos de entrada (como teclado, mouse, joystick), possui métodos virtuais que devem ser implementados para cada estado;

NoLaco: Controla a alteração de dados das entidades;

NaRenderizacao: Agrega ao plano de exibição todas camadas pertencente a entidades.

NaLimpeza: Coletor de lixo, limpa superfícies (SDL);

Criação de Entidade:

FEntidade: Cria uma entidade genérica, possui atributos necessários para criar um objeto com suporte a movimento, colisao, carregamento de um recurso de imagem;

FEntidadeTexto: Cria uma entidade de texto básica, alem dos recursos basicos de uma entidade permite carregar um recurso ou um arquivo de fonte;

FEntidadeBotao: Entidade com atributos e metodos para funcionar como um botao, como “hover” (Metodo AoPassarPorCima ), e clique (AoClicarDireito/ AoClicarEsquerdo), através da colisão com o mouse;

Controle de Colisão: Toda entidade quando se move (por causa de algum evento) tem chance de colidir com outro objeto, a entidade tem capacidade de verificar a sua posição com a do objeto colidido e tomar ações de acordo com a entidade;

Tratamento de Exceção: Através da classe Exceçoes, os atributos estáticos msgErro, classErro e linhaErro, geram na saída padrão mensagens de erro formatas (eventualmente pode ser considerado a criação de uma janela “modal” no próprio jogo informando o erro;

Estados: A classe FGerenciadorEstados, faz a troca de estados, por exemplo a troca da Intro pelo menu, menu pelo jogo ou options e do retorno dos estados para o menu;

Essa classe (FGerenciadorEstados), chama as instancias estaticas de cada estado previamente já carregadas, apenas uma instancia de cada estado pode ser carregada durante a execucao do jogo (construtor privados);

Cada estado possui sua propria especializacao da classe FEstado, tornando cada estado independente um do outro, cada um com suas peculiaridades;

Fractal:

Inicialmente foi conceituado utilizar um fundo fractal, porém devido ao curto prazo e alguns detalhes de implementação, não foi possivel utilizar com plena funcionalidade o fractal, que seria usado como fundo, na pasta TesteFractal é possivel verificar como seria a implementação do Fractal, porem é meramente demonstrativo uma vez que não foi documentado nem organizado da mesma forma que o framework do Simulador de Voo;

Usuário:

==========

Navegacao:

É possivel navegar usando o mouse e/ou as setas direcionais e enter para selecionar a opcao desejada.

Esc, volta ao Menu principal, ou sai do jogo, caso já esteja no estado Menu;

na Intro é possivel apertar enter/esc/clique do mouse, para pular (intro tem tempo de 3 segundos antes de entrar no jogo)

Menu:

Inicio:

Botao de inicio de jogo, troca para o Estado do jogo,

Opções:

Possui apenas uma lista das resoluções encontradas na maquina, porem ainda nao é possivel mudar essas opcões;

Sair:

Sai do programa, Esc no menu principal e fechando o programa tem o mesmo efeito;

No Jogo:

O jogo permite, atualmente, navegar pela tela de fundo usando WASD, mirar com o mouse e atirar com o botao direito e esquerdo, os canhões sao independentes e deveriam apontar para onde o mouse esta, porem necessitam de ajustes nos codigos de angulo;

Créditos:

Engine: SDL – http://www.libsdl.org/
Framework: SDL Tutorials – http://www.sdltutorials.com/
Imagens: Lúcio (cockpit, tiro, canhoes, efeitos de luz), google imagens
Fontes: http://www.netfontes.com.br/
Referencias, exemplos e tudo mais: google

Agradecimentos:
Prof. Gerson Cavalheiro – Por nos dar oportunidade de fazer um trabalho do tipo.
SDL Tutorials – Por criar tutoriais explicando – muito bem – o funcionamento básico de um framework voltado para jogos e sobre SDL.