Interfaces Gráficas Aplicadas ao Ensino das Metodologias de P.O.O.

José Magno Lopes
ESTG-IPLEI
IST
magno@estg.iplei.pt

Mário Rui Gomes
IST-UN
INESC
mrg@inesc.pt

Manuel V. Rodrigues
BESCL
IST
mvrodrigues@bes.pt

Marco M. Ferreira
ESEL-IPLEI
ESTG-IPLEI
mpmf@bigfoot.com

1. SUMÁRIO

O documento descreve as vantagens da aplicação do desenvolvimento de interfaces gráficas no ensino/aprendizagem das metodologias da programação orientada aos objectos.

Assim, é efectuada uma descrição do trabalho desenvolvido no triénio lectivo de 1995/1998, na leccionação de uma disciplina de Programação Orientada por Objectos.

Como exemplo, são apresentados os dois trabalhos práticos mais representativos, desenvolvidos no âmbito da disciplina.

1.1 Palavras-chave

Programação Orientada por Objectos, Interfaces Homem-Máquina, Ensino/Aprendizagem, Computação Gráfica, Diagramas de Transição de Estados.

2. INTRODUCÃO

2.1 Contexto do Documento

O presente documento insere-se no trabalho desenvolvido no âmbito da leccionação, por parte do primeiro autor, no último triénio lectivo, da disciplina de Programação Orientada por Objectos da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Neste contexto foram desenvolvidas técnicas de Ensino/Aprendizagem aplicando os conhecimentos de Interacção Homem-Máquina adquiridos no trabalho de dissertação de Mestrado [4] do 1° autor, orientada pelo Doutor Mário Rui Gomes.

2.2 Objectivos

O principal objectivo proposto neste documento é o de evidenciar a importância da utilização do desenvolvimento de uma interface gráfica no ensino da Programação Orientada por Objectos, nomeadamente na aprendizagem das metodologias de herança e polimorfismo, bem como, o de realçar a necessidade da utilização das técnicas de P.O.O na implementação de interfaces gráficas.

Outro objectivo, não menos importante, é o de demonstrar a necessidade da utilização de uma hierarquia de operadores e estados na implementação de uma interface orientada por objectos, cujo comportamento seja definido por um diagrama de transição de estados.

Finalmente, a interface gráfica proposta nas aulas práticas, deve ser independente da ferramenta utilizada.

3. ENQUADRAMENTO

3.1 Programa da Disciplina

3.1.1 Teórica

A disciplina de Programação Orientada por Objectos está enquadrada 2° ano, 2° semestre do curso de Engenharia Informática em regime diurno e nocturno.

As aulas teóricas têm uma duração semanal de 2 horas, sendo leccionadas com o auxílio de retroprojecção de acetatos dos apontamentos da disciplina [1].

O programa da disciplina é constituído pelos seguintes capítulos: Introdução às técnicas de POO. Classes. Vectores, ponteiros e referências. *Overloading* de funções e operadores. Herança. Funções virtuais. Sistema de entrada/saída. Generacidade e excepções. Regras práticas de POO.

3.1.2 Prática

As aulas práticas têm uma carga horária de 3 horas semanais, leccionadas em laboratórios de computação, onde nas primeiras seis aulas são desenvolvidos exercícios de aplicação prática de conhecimentos básicos de P.O.O., sendo nas restantes aulas desenvolvida uma interface gráfica de aplicação das metodologias de herança, polimorfismo e *templates*.

Como ferramentas de desenvolvimento, nos anos lectivos 1995/97 foi utilizada o Borland C++ 5.0 e no ano lectivo 1997/98 o C++ Builder 1.0.

3.2 Avaliação da Disciplina

Nos anos lectivos de 1995/97 a avaliação foi efectuada a partir do desenvolvimento de um trabalho prático de elevado grau de dificuldade (editores de circuitos eléctricos e editores de jogos).

No ano lectivo de 1997/98 foi efectuada uma avaliação contínua através da realização de fichas aplicativas das aulas práticas (elaboradas pelos alunos fora do âmbito das aulas) e de um trabalho prático de nível de dificuldade média (gestão de fichas dentárias).

4. DESENVOLVIMENTO

4.1 Definição do Problema

O problema proposto consiste em demonstrar as vantagens da utilização de uma interface gráfica, em comparação com a abordagem clássica no ensino/aprendizagem das metodologias da programação orientada aos objectos. No presente documento vamos considerar como exemplo de abordagem clássica a gestão de contas bancárias.

Para tal, deve ser descrito o desenvolvimento de uma aplicação gráfica que possibilite a aquisição por parte dos alunos das técnicas de P.O.O., de modo natural, fácil e sobretudo visual; criando nos alunos a necessidade de pensar no futuro em termos objectos e não na forma tradicional.

No final de cada uma das etapas de desenvolvimento da aplicação é facultado aos alunos uma ficha de aplicação prática das aulas, de modo a permitr efectuar uma avaliação informativa e quantitativa dos conhecimentos adquiridos e dificuldades apresentadas quando confrontados com um novo problema.

A aplicação a desenvolver deve permitir efectuar a construção e manipulação de objectos gráficos definidos por dois pontos.

Nas secções seguintes vamos expor a sequência em como a aplicação (ver Figura 1) é desenvolvida no decorrer das aulas práticas.

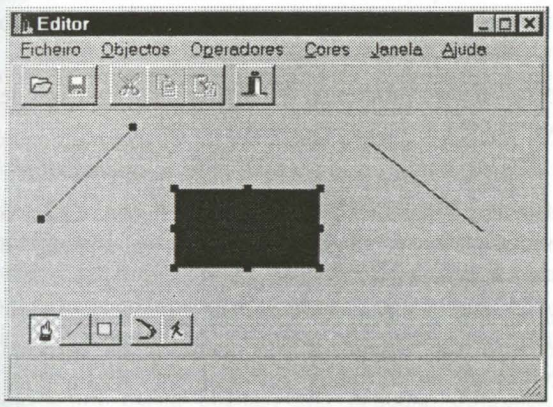


Figura 1: Aplicação a desenvolver nas aulas práticas.

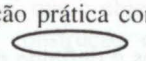
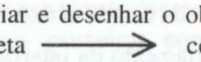
4.2 Objectos

De modo a tornar a tarefa de aprendizagem, por parte dos alunos, mais fácil, são escolhidos os objectos simples: linha (segmento de recta) e rectângulo.

Para resolver o problema proposto, é definida uma interface de objectos com as seguintes características:

- *Atributos*: nome, primeiro ponto, segundo ponto, cor de linha e cor de fundo.
- *Métodos*: desenhar, mover, sobre, envolvente do rectângulo e tipo.

O comportamento desta interface é definida pela classe *TObjecto*. As classes derivadas *TObjectoLinha* e *TObjectoRectangulo* definem o comportamento respectivamente do objecto linha e objecto rectângulo.

O objectivo desta fase é que os alunos adquiram as noções básicas de herança e funções virtuais. A ficha de aplicação prática consiste em criar e desenhar o objecto elipse  e objecto seta  com as mesmas funcionalidades da aula prática. Com o objecto elipse pretende-se que o aluno saiba até que ponto adquiriu as noções transmitidas. Por outro lado, com o objecto seta pretende-se confrontar o aluno com o problema de implementar um objecto definido por três pontos (abertura e dimensão da extremidade da seta são definidos pelo 3º ponto).

Ao definir a hierarquia de objectos anterior, torna-se por mais evidente que para o aluno é mais sugestivo criar objectos gráficos e vê-los desenharem-se ou saber porque não se desenham, do que criar contas bancárias e ver se são escritas, ou não determinadas mensagens no écrã.

4.3 Construção de Objectos

A construção de objectos é efectuada através da selecção de dois pontos utilizando os eventos do rato.

O controle dos eventos do rato associados à janela activa é efectuado pelo estado da operação activa.

Considera-se um operador toda a ferramenta, que permita efectuar uma dada operação definindo o comportamento dos eventos associados à janela activa desencadeados através do rato. Este comportamento vai depender do estado actual da operação efectuada pelo operador.

No caso particular apenas são considerados os seguintes eventos associados ao rato:

- ***BEP*** - Botão esquerdo pressionado;
- ***RM*** - Rato movido;
- ***BES*** - Botão esquerdo solto.

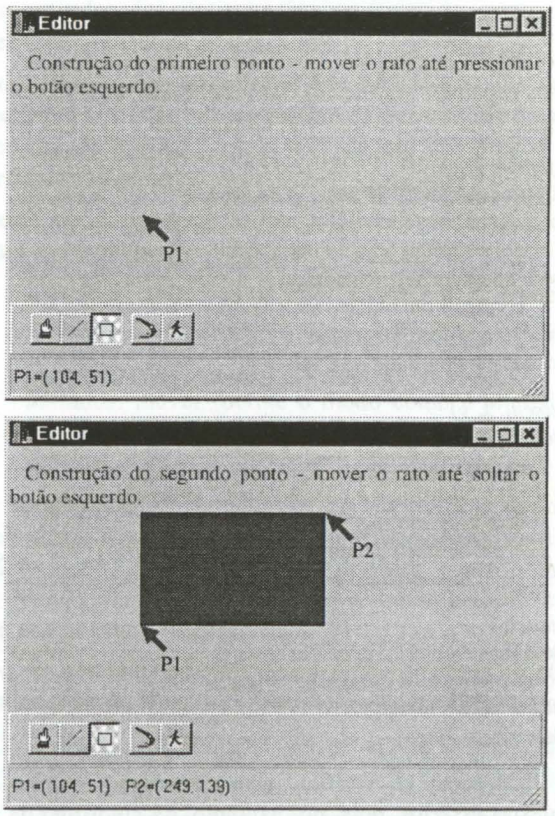


Figura 2: Fases de construção de um objecto.

Como se verifica na Figura 2, o comportamento da operação de construção de um objecto (no caso particular linha e rectângulo) definido por dois pontos pode ser, descrito pelo diagrama de transição de estados representado na Figura 3.

O **Estado CPP** define o comportamento de resposta aos eventos do rato: ***RM*** quando o utilizador ainda não pressionou o botão esquerdo do rato e ***BEP*** quando o utilizador pressiona o botão esquerdo.

O **Estado CSP** define o comportamento de resposta aos eventos do rato: ***RM*** quando o utilizador ainda não soltou o botão esquerdo do rato e ***BES*** quando o utilizador solta o botão esquerdo.

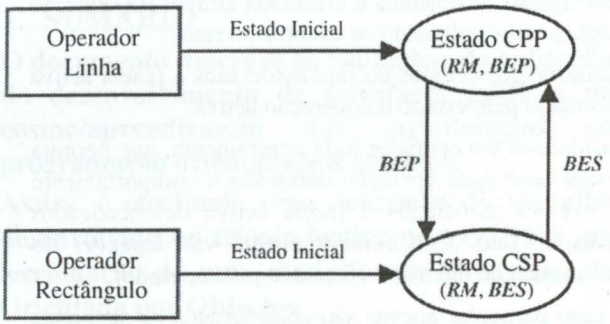


Figura 3: Diagrama de transição de estados da operação de construção de um objecto linha e rectângulo.

Para que os alunos entendam a vantagem e necessidade da utilização de metodologias orientadas por objectos, apresentamos-lhes duas abordagens.

4.3.1 Abordagem tradicional

Na abordagem tradicional, isto é, orientada para funções, temos os seguintes algoritmos das funções de resposta aos eventos.

```
BEP (janela, x, y)
    definir os pontos P1=P2=(x, y) do objecto
    caso o operador seja linha
        construir um objecto linha
    caso o operador seja rectângulo
        construir um objecto rectângulo
    desenhar o objecto na janela
    transitar para o estado de CSP

RM (janela, x, y)
    caso o operador seja linha ou rectângulo
        caso o estado seja CPP
            identificar a posição P1=(x, y) do objecto
        caso o estado seja CSP
            definir o ponto P2=(x, y) do objecto
            redesenhar o objecto na janela

BES (janela)
    redesenhar o objecto na janela
    colocar o objecto no vector de objectos
    transitar para o estado de CPP
    redesenhar a janela
```

Como facilmente se verifica, esta abordagem origina testes desnecessários, pois, por exemplo, no algoritmo da função de resposta ao evento ***RM*** após pressionarmos o botão esquerdo sabemos que estado é CSP, no caso inverso, se o botão esquerdo ainda não foi pressionado ou o botão esquerdo tiver sido solto sabemos que o estado é CPP.

Outro problema se coloca à medida que o número de operadores e estados de operação vai aumentando, gerando-se uma confusão tamanha, sendo quase impossível efectuar a verificação de erros.

Por fim, sempre que para operações semelhantes exista apenas uma pequena diferença de comportamento, o método tradicional tem tendência a levar-nos à duplicação de código.

Para resolver os problemas propostos, existe a necessidade de olhar o problema na perspectiva orientada aos objectos.

4.3.2 Abordagem orientada aos objectos

Na abordagem orientada aos objectos vamos pensar de forma inversa, isto é, identificar os objectos e definir o comportamento destes.

Assim, podemos facilmente identificar uma hierarquia de operadores (operador linha e operador rectângulo), onde cada um dos operadores tem um estado inicial e um estado actual, em função da fase da operação desempenhada.

Para resolver este problema, é definida uma interface de operadores com as seguintes características:

- *Atributos*: nome, estado actual, objecto operado
- *Métodos*: BEP, RM, BES, criar objecto (que define o modo como é criado o objecto a operar) e seleccionar operador.

Os métodos de resposta do operador aos eventos ***BEP***, ***RM*** e ***BES*** são delegados nos métodos de resposta respectivos do estado actual do operador. Assim pode dizer-se que o comportamento de um operador é definido pelo comportamento do seu diagrama de transição de estados.

O comportamento da interface de operadores é definida pela classe *TOperador*. As classes derivadas *TOperadorLinha* (na função criar objecto é criado de um objecto linha) e *TOperadorRectangulo* (na função criar objecto é criado de um objecto rectângulo) definem, respectivamente, o comportamento do operador linha e operador rectângulo.

Falta-nos agora definir uma hierarquia de estados (estado CPP e estado CSP), definida pela interface com as seguintes características:

- *Atributos*: nome e operador respectivo
- *Métodos*: BEP, RM, BES

Assim, os estados devem definir o comportamento de resposta aos eventos, e não, como na perspectiva tradicional, onde para cada evento existia uma resposta em função do estado actual.

O comportamento da interface de estados é definida pela classe *TEstado*. As classes derivadas *TEstadoCPP* e *TEstadoCSP*, definem, respectivamente, o estado de CPP e estado de CSP.

Na classe *TEstadoCPP* temos:

BEP *(janela, x, y)*
definir os pontos P1=P2=(*x*, *y*) do objecto do operador
criar objecto do operador
desenhar o objecto do operador na *janela*
transitar para o estado de CSP

RM *(janela, x, y)*
identificar a posição P1=(*x*, *y*) do objecto

Na classe *TEstadoCSP* temos:

RM *(janela, x, y)*
definir o ponto P2=(*x*, *y*) do objecto
redesenhar o objecto na *janela*

BES *(janela)*
redesenhar o objecto na *janela*
colocar o objecto no vector de objectos
transitar para o estado de CPP
redesenhar a *janela*

Como facilmente se verifica, esta abordagem evita os testes desnecessários, pois, cada estado define apenas o comportamento de resposta aos eventos que lhe dizem respeito.

À medida que o número de operadores e estados de operação vai aumentando, apenas é necessário definir os comportamentos desses novos operadores e o comportamento dos estados desses operadores. Note-se que, na abordagem orientada aos objectos podemos "herdar" o que for comum e sempre que para operações semelhantes exista apenas uma pequena diferença de comportamento, podemos sempre definir uma nova função de interface (virtual) que defina o comportamento somente dessa diferença.

Nesta fase, tem-se como objectivo que os alunos percebam, finalmente, a grande vantagem da utilização de uma abordagem orientada aos objectos. Tal, só pode ser conseguido, a nosso ver, através do desenvolvimento de uma interface gráfica orientada por objectos, pois "nada melhor, do que ver para crer ...".

Por outro lado, a utilização de uma interface de operadores e estados, permite alargar a abrangência a qualquer tipo de interface homem-máquina, conhecendo apenas os diagramas de transição de estados que definem o comportamento de cada operador da interface.

De modo, a que aluno perceba as vantagens da utilização da abordagem orientada por objectos, é facultada aos alunos uma ficha de aplicação prática onde devem ser criados de modo interactivo os objectos elipse e seta. Note-se, que o objecto seta deve ser construído por duas fases: construção da linha (dois primeiros pontos) e construção da extermidade da seta (3º ponto).

Note-se que, no exemplo clássico da gestão de contas bancárias é difícil conceber um raciocínio semelhante ao efectuado para a construção de objectos gráficos.

4.4 Selectores de Objectos

Ao seleccionar um objecto devem ser desenhados os seus selectores, como se vê na Figura 4.

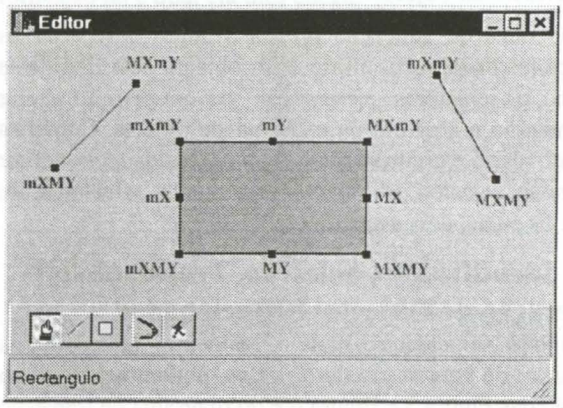


Figura 4: Representação de selectores de objectos.

Para um objecto em geral, existem oito selectores, dos seguintes três tipos:

- alteração em X (mX e MX)
- alteração em Y (mY e MY)
- alteração em X e Y (mXmY, mXMY, MXmY e MXMY)

das coordenadas do primeiro ou segundo ponto que definem o objecto a alterar.

Em termos gerais, um selector não é mais que um objecto rectângulo, que ao ser movimentado permite alterar uma coordenada em X, Y ou X e Y de um dos pontos que definem o objecto.

Deste modo, podemos definir uma interface de selectores com as seguintes características:

- *Atributos*: endereço da coordenada em X a alterar, endereço da coordenada em Y a alterar e o tipo de selector
- *Métodos*: mover (define o modo como é alterado o objecto ao mover-se o selector) e devolver tipo de selector.

O comportamento da interface de selectores é definido pela classe *TSelector*. As classes derivadas *TSelectorAltX*, *TSelectorAltY* e *TSelectorAltXY*, definem, o modo como se deve alterar um objecto ao mover um selector de cada um dos três tipos respectivos.

Assim, por uma questão de coerência, podemos definir um selector de objecto com as seguintes características:

- *Atributos*: objecto a mover ou alterar e vector de 8 selectores.
- *Métodos*: mover (move os selectores ao mover-se o objecto), sobre (devolve o endereço do selector sobre o qual se encontra um ponto, ou 0) e desenhar.

O comportamento de um selector de objecto é definido pela classe TSelectorObjecto. A classe derivada TSelectorObjectoLinha define o comportamento de um selector de um objecto linha.

Deste modo, é necessário aumentar a funcionalidade da interface de objectos, acrescentando-lhe o atributo: selector de objecto e os métodos seleccionar e desseleccionar objecto.

Como ficha de aplicação prática o aluno deve criar os selectores do objecto elipse e do objecto seta. Para este último, os selectores devem permitir alterar a dimensão e abertura da extremidade da seta. Com esta ficha o aluno é confrontado com a necessidade de definir um novo selector de objecto com quatro selectores, de entre os quais, dois específicos.

4.5 Identificação, Selecção, Translação e Alteração de Objectos Seleccionados

De modo semelhante, para o raciocínio efectuado na operação de construção de objectos, podemos agora, em função das Figura 5 e Figura 6 e do diagrama de transição de estados representado na Figura 7, tornar a hierarquia de operadores e estados, abrangente para a identificação, selecção, translação e alteração de objectos.

Assim sendo, à interface de operadores é necessário introduzir o atributo selector, que permite efectuar alterações nos objectos seleccionados através do movimento deste.

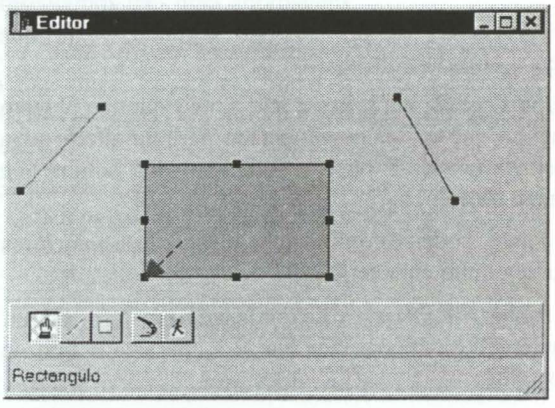


Figura 5: Translação dos objectos seleccionados ao translacionar o rectângulo, relativamente à Figura 4.

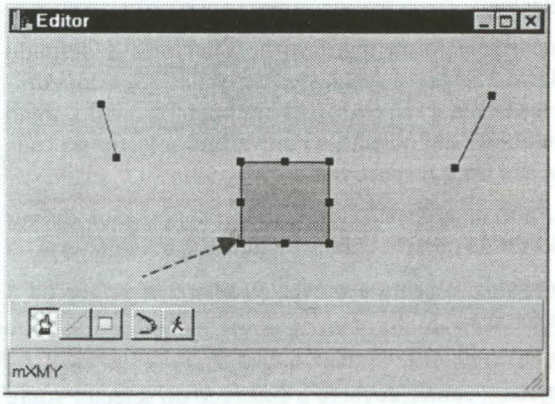


Figura 6: Alteração dos objectos seleccionados a partir do selector mXMY do rectângulo.

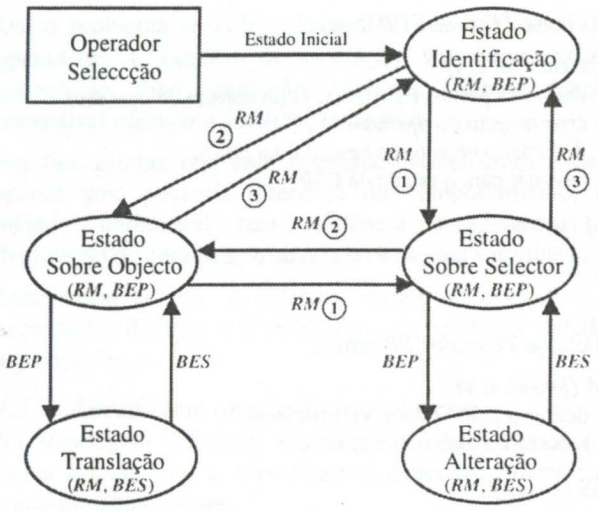


Figura 7: Diagrama de transição de estados das operações de identificação, selecção, translação e alteração.

A classe TOperadorSeleccao derivada de TOperador, define o comportamento da identificação, selecção, translação e alteração de objectos seleccionados.

A classe TestadoIdentificacao, derivada de TEstado define o comportamento geral de identificação (evento ***RM***), enquanto que as classes derivadas desta, TEstadoSobreObjecto e TEstadoSobreSelector apenas acrescentam o respectivo comportamento para o evento ***BEP***. Por seu lado, as classes derivadas de TEstado, TEstadoTranslacao e TEstadoAlteracao definem, respectivamente, o comportamento da fase de translação e alteração dos objectos seleccionados.

Nesta fase, os alunos já devem ser capazes de definir e implementar uma nova operação, por exemplo, efectuar a operação de colocar após copiar objectos seleccionados.

A ficha de aplicação prática consiste em permitir identificar, seleccionar, translacionar e alterar o objecto elipse e o objecto seta. Tendo este último, que possibilitar ao utilizador a possibilidade de alterar interactivamente a a dimensão e abertura da sua extremidade.

4.6 Leitura e Gravação de Objectos em Ficheiro

A leitura e gravação de objectos em ficheiro é descrita pelas seguintes fases:

- Leitura e gravação de um objecto em ficheiro;
- Definição e implementação de uma lista de objectos utilizando uma lista genérica (*templates*). que saiba mover manipular os objectos nela contidos;
- Leitura e gravação em ficheiro dos objectos da lista.

A ficha de aplicação prática consiste em permitir atribuir estas funcionalidades aos objectos elipse e seta.

4.7 Aplicação

A aplicação a desenvolvida apresenta as principais características:

- *Atributos*: menu de opções, barra de comandos, barra de operações, barra de mensagens, lista de objectos, vector de estados, vector de operadores, o operador seleccionado, cor de linha e cor de fundo actual.
- *Métodos*: funções de resposta aos eventos ***BEP***, ***RM*** e ***BES***, desenhar os objectos e gestão dos restantes atributos.

Deve salientar-se, que o comportamento dos métodos de resposta aos eventos ***BEP***, ***RM*** e ***BES*** da aplicação é delegado no comportamento dos métodos de resposta aos eventos respectivos do operador seleccionado, que por sua vez delega o comportamneto respectivo no estado actual do operador.

No final destas etapas, os alunos devem ser capazes de desenvolver uma aplicação em qualquer âmbito utilizando correctamente e de modo eficiente as metodologias orientadas por objectos e as técnicas de interacção homem-máquina expostas no decorrer das aulas práticas.

5. TRABALHOS MAIS IMPORTANTES DESENVOLVIDOS NA DISCIPLINA

No decorrer do triénio lectivo de 1995/98 foram desenvolvidos trabalhos de elevado cariz criativo e técnico. De onde se salientam os dois trabalhos mais relevantes, desenvolvidos respectivamente, pelo 3º e 4º autor do presente documento.

5.1 Editor e Simulador de Circuitos Digitais

5.1.1 Definição do problema

O trabalho proposto consistiu no desenvolvimento de uma aplicação gráfica orientada aos objectos que possibilitasse efectuar a construção e manipulação directa de circuitos digitais [2]. A aplicação teve igualmente como propósito efectuar a simulação e teste do circuito digital construído.

5.1.2 Elementos

Os circuitos digitais são a peça fundamental da micro-electrónica, coexistindo com a electrónica analógica (não abordada).

Estes circuitos são constituídos por pequenos elementos dividindo-se estes em **entradas**, **portas**, **saídas** e **pontes** (ligações). São este elementos que ligados entre si possibilitam a criação de circuitos integrados e outros produtos finais.

As **entradas** são elementos que servem como fonte de informação aos circuitos, isto é, apenas debitam dados.

VCC 

GND 

CLOCK 

As **portas** são elementos que transformam ou apenas transportam informação, isto é, aplicam uma transformação e debitam-na ou apenas deixam passar a informação tal e qual ela lhes chega.

AND 

OR 

NOT 

NAND 

NOR 

XOR 

FlipFlop - JK 

As **saídas** são elementos que servem para visualizar a informação computada, isto é, apenas recebem informação para mostrar, sendo estes aqueles que permitem que se teste o resultado do circuito.

LED 

BCD 4 Entradas 

BCD 7 Entradas 

5.1.3 Editor de circuitos

As principais operações em elementos de um circuito digital são:

- Criação de elementos (Figura 9);
- Identificação e selecção de elementos (Figura 8);
- Ligação de elementos (Figura 10);
- Translação de elementos;
- Cópia e remoção de elementos;

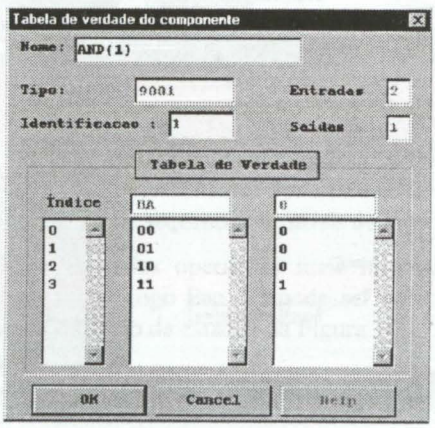


Figura 8: Identificação de elementos de um circuito digital.

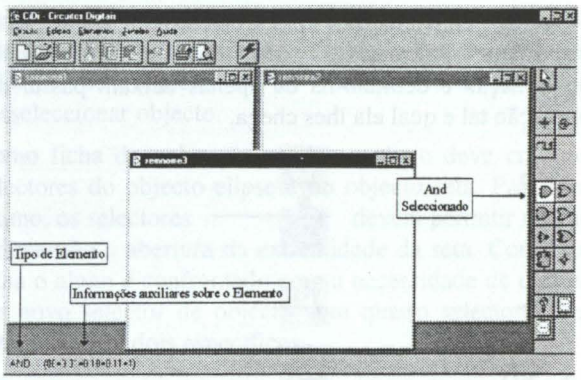


Figura 9: Criação de um elemento de um circuito.

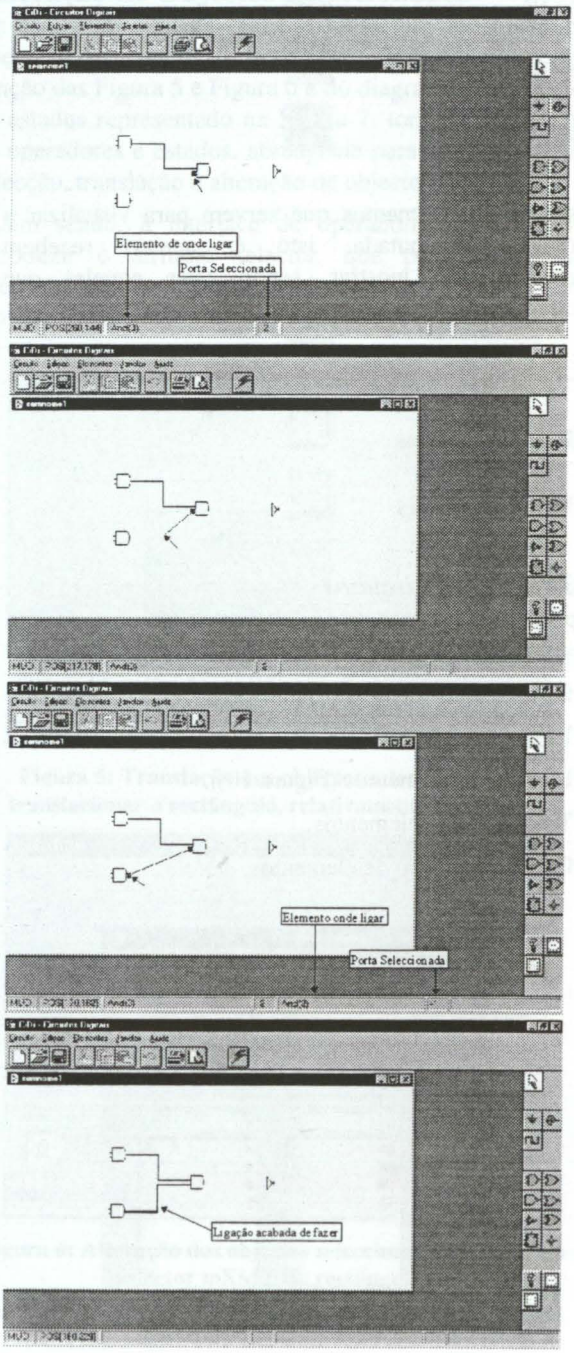


Figura 10: Fases de ligação de elementos de um circuito.

O comportamento das operações mais importantes podem ser definidas pelo diagrama de transição de estados da Figura 11.

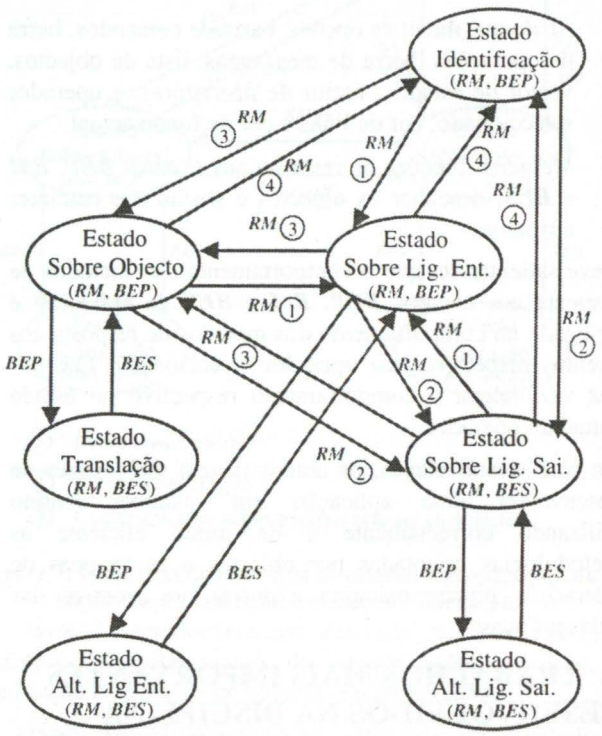


Figura 11: Diagrama de transição de estados que define o comportamento das operações de um editor de circuitos.

5.1.4 Simulador de circuitos

A simulação de circuitos (Figura 12) é efectuada por níveis, do seguinte modo:

Inserção na lista todos os elementos fontes de dados.

Enquanto existirem elementos na lista:

Retirar o primeiro elemento da lista.

Ordenar-lhe que calcule as suas saídas.

Inserir, no fim da lista, os elementos ligados às saídas do elemento calculado, se estes ainda não foram lá colocados.

Retirar da lista o elemento calculado.

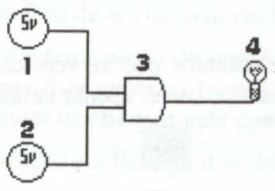


Figura 12: Exemplo de um circuito digital.

5.1.5 Reflexão final

A aplicação desenvolvida pode ser utilizada para construir circuitos digitais, como ferramenta de apoio ao ensino/aprendizagem da micro-electrónica.

5.2 Pac 3D

5.2.1 Definição do problema

O trabalho proposto consistiu no desenvolvimento de uma aplicação gráfica orientada aos objectos (Figura 13) que possibilitasse efectuar a construção de níveis de um jogo estilo *PacMan* tridimensional [3]. A aplicação, teve igualmente, como propósito, o de implementar o modo de jogo e definir as sequências de níveis deste.

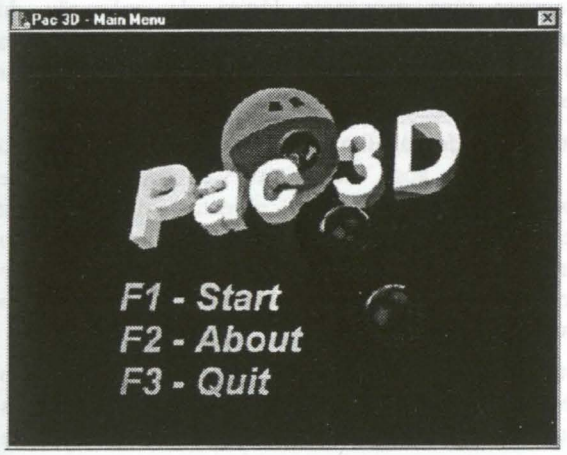


Figura 13: Jogo Pac 3D.

O objectivo do jogo a duas dimensões mantém-se, ou seja, apanhar toda a comida existente no nível sem se deixar apanhar pelos fantasmas. No entanto, a dificuldade aumenta, pois já existe a noção de altura, não se podendo cair das passadeiras onde se efectua o movimenta. Por existir uma terceira dimensão torna-se lógica a capacidade de saltar, sendo este o meio usado para passar pelos fantasmas (na versão original ter-se-ia que comer uma comida especial que permitiria comer os fantasmas) ou para passar entre plataformas não unidas.

5.2.2 Objectos

O jogo Pac 3D é constituído pelos seguintes objectos:

- Pacman
- Fantasma (tenta apanhar o pacman)
- Comida (dá energia ao pacman)
- Passadeira (onde se movimentam o pacman e o fantasma e sobre a qual é colocada a comida)
- Direcção (objecto que define as direcções possíveis para os fantasmas)

5.2.3 Editor

O editor de níveis do jogo Pac 3D representado na Figura 14 possibilita as seguintes operações principais:

- Selecção (permite selecionar, mover, alterar e remover objectos)
- Colocação de passadeiras;
- Colocação de comida;
- Definição de direcções de um fantasma (define o local, modo e temporização do movimento de um fantasma).
- Visualização tridimensional do nível construído.

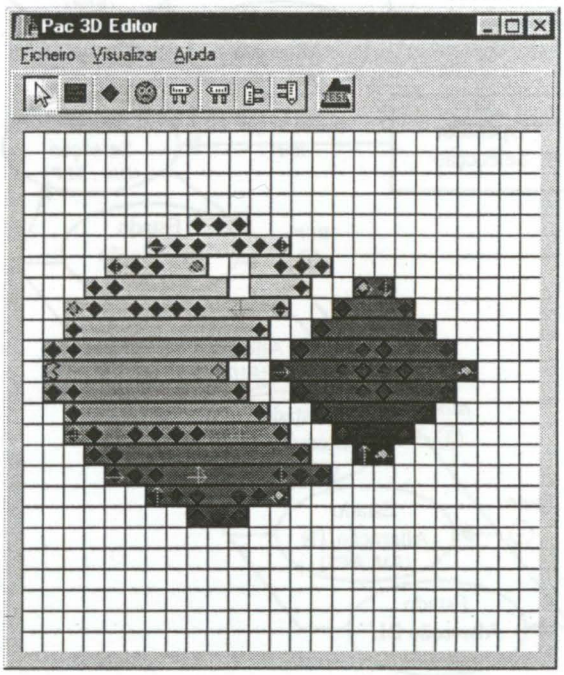


Figura 14: Editor de níveis do jogo Pac 3D.

As sequências de níveis é estipulada pela sua edição, como se representa na Figura 15.

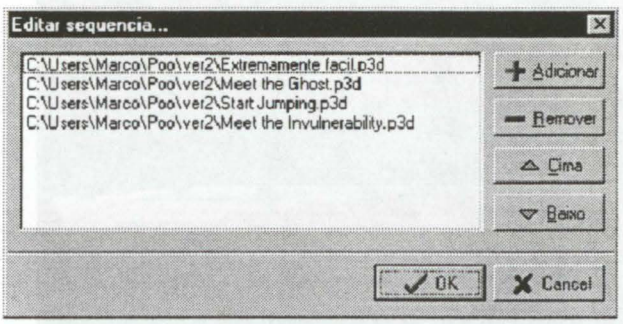


Figura 15: Edição de sequências de níveis do jogo Pac 3D.

O comportamento das operações mais importantes do editor de níveis do jogo Pac 3D pode ser definido pelo diagrama de transição de estados da Figura 16.

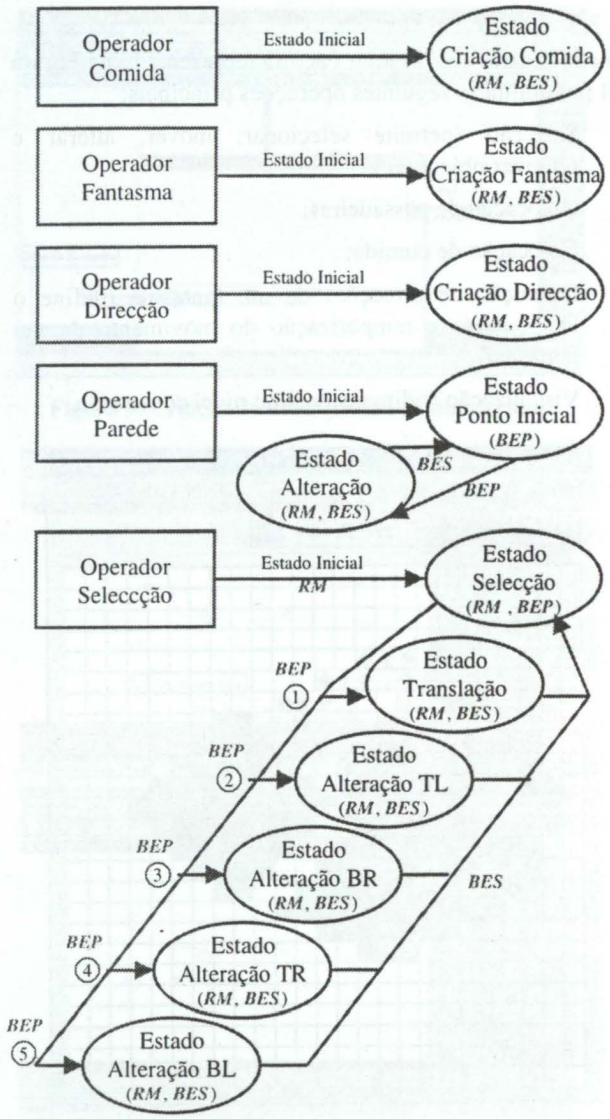


Figura 16: Diagrama de transição de estados que define o comportamento das operações de um editor de circuitos.

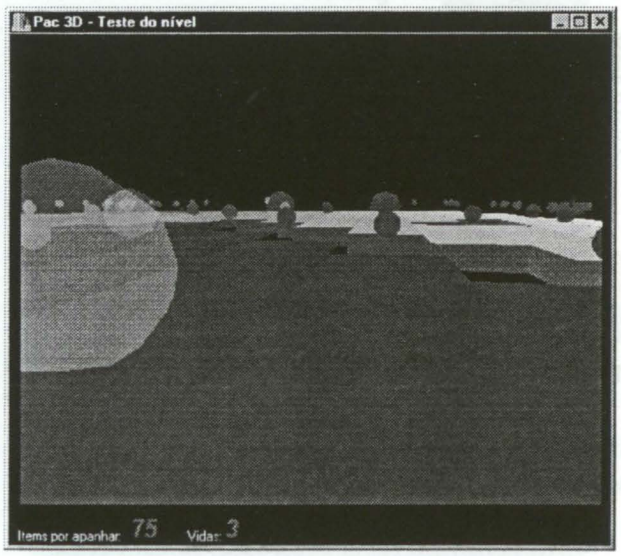


Figura 17: Jogo Pac 3D do nível representado na Figura 14.

5.2.4 Modo de jogo

O modo de jogo do Pac 3D (Figura 17) foi implementado usando a *API OpenGL* para efectuar a representação, visualização e jogo de modo tridimensional.

5.2.5 Reflexão final

A grande vantagem deste jogo consiste no jogador poder construir os seus próprios níveis e definir as sequências deste. Realmente que bom seria podermos criar os níveis dos nossos jogos favoritos, por exemplo, *Tetris*, *Arkanoíde*, *PacMan*, etc.

6. CONCLUSÕES

Pensamos, com este documento, ter conseguido demonstrar uma sequência lógica de aplicação do trabalho desenvolvido no âmbito da dissertação [1] na leccionação da disciplina de Programação Orientada por Objectos.

O intuito de todo trabalho desenvolvido neste triénio foi o de transmitir aos alunos os conhecimentos de metodologias orientadas aos objectos, por forma a poderem aplicar esses conhecimentos no desenvolvimento de projectos de elevado nível científico e tecnológico, de preferência utilizando a computação gráfica como suporte.

Pensamos ter ficado claro ao longo da descrição do presente documento, o porquê da utilização de aplicações gráficas para o ensino/aprendizagem das metodologias de P.O.O., quando comparadas com os métodos clássicos.

Com a avaliação contínua introduzida no ano lectivo de 1997/98 conseguiu-se que os alunos adquirissem, mais facilmente, as técnicas e conhecimentos no decorrer da leccionação da disciplina, e não apenas, quando se aplicavam, mais a fundo, no desenvolvimento do trabalho final.

Com a utilização da hierarquia de objectos, operadores e estados utilizada para implementar interfaces gráficas de manipulação directa, cujo comportamento é definido por diagramas de transição de estados, pensamos ter sido incutido nos alunos a grande diferença entre saber utilizar uma linguagem de programação orientada por objectos e saber pensar e programar de modo orientado aos objectos.

Os alunos apresentam extrema dificuldade em entender o conceito de operador e de estado quando lhes é transmitido, mas após a sua compreensão, a maioria deles não apresenta grandes dificuldades em definir diagramas de transição de estados para as suas operações e implementar o comportamento destas.

Finalmente, e sem queremos ser pretenciosos, pensamos que o modo como esta disciplina está estruturada pode ser de grande utilidade para quem aplica outras técnicas de ensino/aprendizagem e tem dificuldades em que os alunos adquiram as metodologias da programação orientada por objectos.

7. REFERÊNCIAS

[1] J. Magno. Apontamentos da disciplina de Programação Orientada por Objectos, ESTG-IPLEI, 1995/1998.

[2] M. Rodrigues. Editor de Circuitos Digitais, Trabalho de Programação Orientada por Objectos, ESTG-IPLEI, 1996.

[3] M. Ferreira. Jogos Pac 3D, Trabalho de Programação Orientada por Objectos, ESTG-IPLEI, 1998.

[4] J. Magno. Ensino por Construção de Objectos Físicos - ECOFÍS, Dissertação de Mestrado, IST, 1996.

[5] J. Morgado. Ambiente Interactivo para o Ensino da Geometria Descritiva - GEODES, Dissertação de Mestrado, IST, 1996.

[6] M. R. Gomes. OO-AGEF Modelação de Sistemas de Interacção Homem-Máquina Baseados em Manipulação Directa, Dissertação de Doutoramento, IST, 1991.

[7] L. Heiny. Windows Graphics Programming with Borland C++, Coriolis Group Book, 1992.