UECE – Universidade Estadual do Ceará
CCT – Centro de Ciências e Tecnologia
Curso de Ciência da Computação

Biblioteca Gráfica: LibJHI-SDL 2.0

Alunos: Henrique Neto e João Gonçalves
Professora: Ana Luiza
E-mails: henrique.brandao@larces.uece.br;
joao.goncalves@larces.uece.br e analuiza@larces.uece.br

Índice

1.0 Biblioteca LibJHI-SDL e biblioteca SDL

A biblioteca LibJHI-SDL foi feita com intuito de torna mais simples e mais acessível as rotinas da biblioteca gráfica SDL. ***Simple DirectMedia Layer*** (SLD) é uma *lib* escrita na linguagem C, projetada para possibilitar acesso a elementos de áudio, teclado, mouse, joystick e recursos gráficos. É bastante utilizada em softwares de vídeos e imagem, emuladores e jogos populares.

Assim, a LibJHI-SDL abstrai vários elementos de código de mais baixo nível da biblioteca SDL, tornando mais fácil a criação de elementos gráficos que podem ser utilizados em algum programa. Tais elementos gráficos podem ser animações, janelas, carregamento de imagens, carregamento de sons, escrita de textos com fontes diversas, jogos, entre outros. Além disso, também oferece funções simplificadas para a interação com o teclado e com o mouse.

Com isso, programadores iniciantes podem desenvolver pequenos projetos que utilizem recursos gráficos sem muita complicação.

Como primeiro passo para a utilização da LibJHI-SDL, é necessária a instalação da biblioteca SDL. Logo abaixo, é indicado o comando para instalá-la no **Ubuntu.** Porém ela pode ser utilizada em outras distribuições Linux ou no Windows.

Para instalar a biblioteca SDL no sistema, abra o termine e digite a linha abaixo:

sudo apt-get install libsdl1.2-dev libsdl-image1.2-dev libsdl-mixer1.2-dev libsdl-ttf2.0-dev libsdl-gfx1.2-dev

Depois de ter instalado a *lib* SDL, pode-se começar a usar a LibJHI-SDL. Os procedimentos para a sua instalação e utilização podem ser vistos nos tópicos mais a frente.

2.0 LibJHI-SDL: Organização, Instalação, Utilização e Compilação de um programa utilizando-a

Após obter a biblioteca (mais a frente será dito como), um conjunto de diretórios podem ser verificados em sua organização. São os seguintes:

- **bin/** Contém os arquivos executáveis dos exemplos compilados da pasta examples
- **doc/** Contém a documentação da biblioteca
- **examples/** Contém os códigos de exemplos
- **fonts/** Contém fontes no formato ttf, utilizados nos textos
- **images/** Contém as imagens utilizadas nos exemplos
- **sounds/** Contém os sons utilizados nos exemplos
- **headers/** Contém os arquivos .h da biblioteca
- **src/** Contém o código fonte da biblioteca
- **project/** Contém o código de um novo projeto que utiliza a biblioteca

Além disso, a biblioteca vem com um arquivo chamado Makefile, que serve para auxiliar compilação. Ao baixar a biblioteca, entre em seu diretório e execute o comando *make* no terminal. Dessa forma, ele fará toda a compilação da biblioteca e dos exemplos, colocando os arquivos executáveis na pasta **bin/**.

2.1 Explicando o Makefile:

Duas **flags** foram utilizadas no Makefile da biblioteca. As flags "***all***" e "***clean***". A flag ***all*** será executada quando for digitado o comando ***make*** no terminal. A flag ***clean*** é executada em conjunto com o comando *make*, ou seja, *make clean.* Quando for digitado este último comando, todos os arquivos .o da pasta **src/** serão apagados. Utiliza-se o *make clean* para apagar todos arquivos de uma compilação antiga para se gerar uma compilação. Na flag ***all***, os dois primeiros comandos são os seguintes:

gcc -I${DIR_HEADER_LIB_PATH} ${COMP_FILES} -c
*mv *.o src/*

O primeiro comando (em vermelho) gera todos os arquivos objetos da biblioteca que são necessários para sua utilização. A variável *DIR_HEADER_LIB_PATH* contém o caminho do diretório dos headers (arquivos de cabeçalhos). Ela é útil para que no momento do include de algum .h da *lib*, seja somente necessário passar o nome do header .h, e não todo caminho até o arquivo. O segundo comando (em verde) move os arquivos objetos, que são gerados no primeiro comando, para a pasta **src/**.

O bloco de código seguinte compila os exemplos da pasta **example/**:

```
for i in `seq 1 ${EXAMPLES}`; \
do \
        echo "COMP EXAMPLE $$i ";\
        gcc examples/exemplo$$i.c  ${OBJECT_FILES} -I${DIR_HEADER_LIB_PATH}
-o bin/exemplo$$i ${LIBSTATIC};\
done
```

A comando em vermelho compila todos os exemplos que seguem a nomenclatura exemplo[número].c. A variável ***OBJECT_FILES*** possui os arquivos .o compilados da *lib*. A variável ***DIR_HEADER_LIB_PATH*** tem o mesmo significado que foi dito anteriormente. A ***LIBSTATIC*** contém os comandos para as bibliotecas estáticas utilizadas.

Após esse comando, todos os executáveis gerados são colocados na pasta **bin/**, seguindo a seguinte nomenclatura: exemplo[número].c.

2.2 Compilando a biblioteca:

Para compilar a biblioteca LibJHI-SDL, basta estar em seu diretório raiz e digitar o comando *make* no terminal:

.../libgraph$ make

2.3 Criando Seu Próprio Projeto:

Para a criação de um novo projeto, existe uma pasta chamada ***project/***. Nela, um ambiente já está todo preparado para a criação de um novo projeto, contendo diretórios de som, imagem e fonte. Neles, você coloca os seus próprios arquivos de som, imagem e fonte que serão utilizados no seu projeto. As pastas estão organizadas de maneira semelhante ao diretório principal:

- **bin/** Pasta em que o Makefile irá colocar o executável do seu projeto
- **fonts/** Pasta para armazenar as fontes de texto
- **images/** Pasta para armazenar as imagens

- **sounds/** Pasta para armazenar os sons
- **headers/** Colocar os arquivos .h criados no seu projeto
- **src/** Colocar os arquivos .c criados no projeto

2.4 Explicando o Makefile do seu Projeto:

Antes de usar o Makefile do seu projeto, é preciso que a biblioteca LibJHI-SDL esteja compilada, ou seja, com seus arquivos **.o**'s gerados. Para isso, basta digitar *make* na pasta principal da biblioteca. Este Makefile contém 6 variáveis para facilitar seu uso:

```
LIBCOMP=../src/*.o

LIBSTATIC=-lSDL -lm -lSDL_mixer -lSDL_image -lSDL_ttf

DIR_HEADER_LIB_PATH=../headers

DIR_HEADER_PROJECT_PATH=headers/

YOURFILES=src/*.c

BINARY=bin/project
```

A variável LIBCOMP contém a localização dos arquivos .o da biblioteca. A LIBSTATIC contém os comando para as bibliotecas estáticas utilizadas. A YOURFILES irá conter a referência para todos arquivos .c que você criará no projeto. Dessa forma, todo arquivo.c que você criar, deve estar na pasta **src/**.

A variável DIR_HEADER_LIB_PATH tem o mesmo significado dito anteriormente. A variável DIR_HEADER_PROJECT_PATH, possui o caminho para diretório **header/** do seu projeto. Dessa forma quando você criar um arquivo .h, ele deverá ser colocado na pasta header. Assim o include no arquivo .c pode ser feito de maneira direta, sem indicar todo o caminho do arquivo.

A variável BINARY basicamente irá conter o nome do seu executável final do projeto. Ele é colocado na pasta **bin/**. Por essa razão, a variavel recebe o nome **bin/nome_do_executável.** Por padrão ela já vem com o nome project.

Para compilar o seu projeto, basta dar um comando *make*, dentro da pasta **project/**. Depois disso, basta executar o arquivo executável dentro da pasta **bin/**.

3.0 Funções da biblioteca LibJHI-SDL 2.0

O download da biblioteca, como também sua documentação completa, explicando todas as funções e estruturas, podem ser encontrados em:

www.larces.uece.br/joao/libjhi-sdl

4.0 Exemplos utilizando a biblioteca LibJHI-SDL 2.0

Para executar os exemplos, basta acessar a pasta **bin/** e executá-los. Na pasta **examples/**, estão os seus códigos fontes com comentários que explicam cada passo. O comando abaixo mostra

a execução do exemplo 1.

…/libjhi-sdl/bin$ ./exemplo1

OBS: O código e o executável de todos os exemplos indicados abaixo, podem ser visualizados quando se baixa a biblioteca.

4.1 Exemplo 1: *Inicializando uma janela*

Neste exemplo, uma simples janela é criada com nome e tamanho indicados.

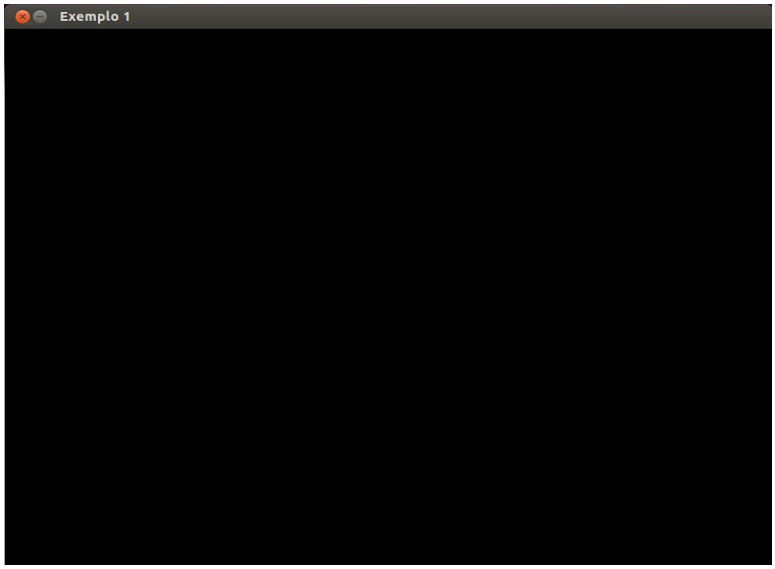


Figura 1: Janela Criada

4.2 Exemplo 2: *Desenhando formas básicas em uma janela*

Neste exemplo, os possíveis desenhos geométricos são desenhados na tela.

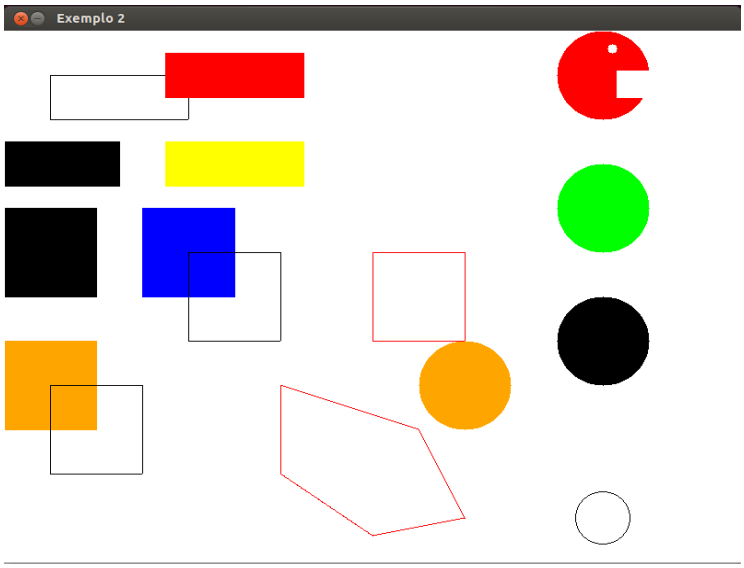


Figura 2: Figuras geométricas

4.3 Exemplo 3: *Carregando um arquivo de Imagem*

Neste exemplo, uma imagem é carregada na tela.

Figura 3: Imagem carregada na tela

4.4 Exemplo 4: *Carregando um arquivo de Imagem e desenhando um Texto na Tela*

Neste exemplo, uma imagem é carregada na tela, desenhando-se também um texto.

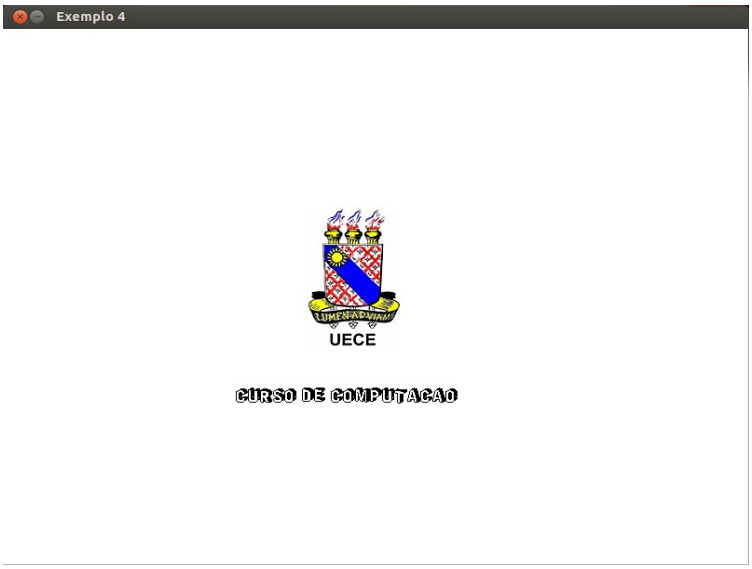


Figura 4: Carregando uma imagem e desenhando um texto

4.5 Exemplo 5: *Carregando um arquivo de Imagem, cortando-se partes dela e deixando a sua*

cor de fundo transparente

Neste exemplo, vários clips do personagem ***Pac man*** são desenhados separadamente na tela. Os clips de cada posição do pac man são cortados de uma imagem que contém todas as posições.

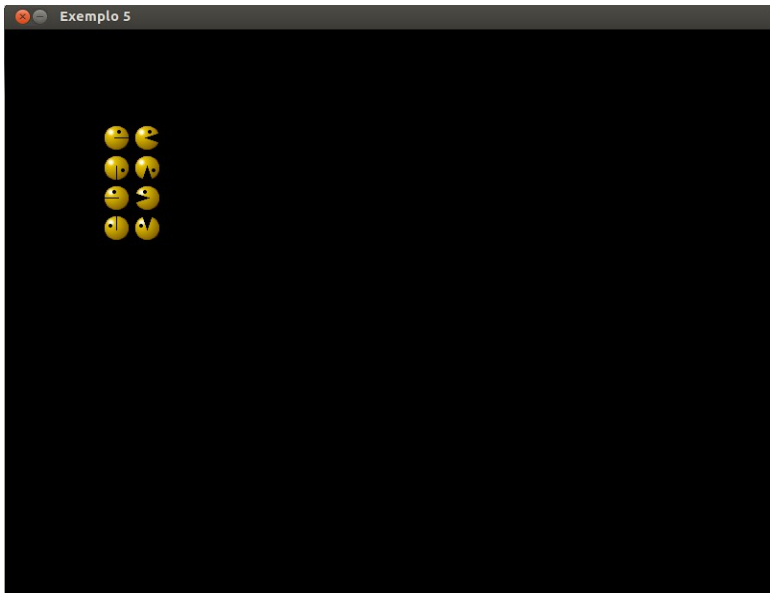


Figura 5: Clips do Pac Man desenhados

4.6 Exemplo 6: *Carregando efeitos de som e capturando eventos do mouse*

Neste exemplo, clicando-se com o botão direito do mouse é emitido um som e clicando-se com o botão esquerdo é emitido outro som.

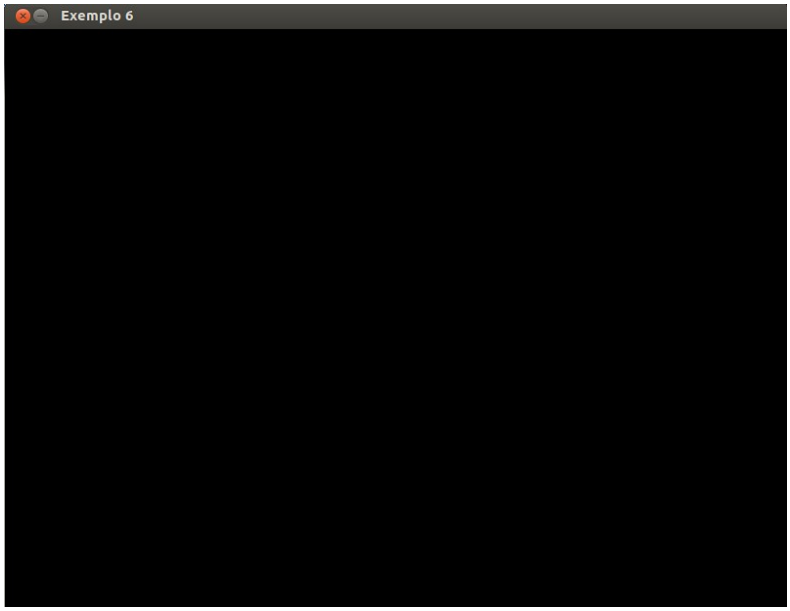


Figura 6: Janela tocando algum som dependendo do botão do mouse clicado

4.7 Exemplo 7: *Carregando Música e capturando eventos do mouse*

Neste exemplo, clicando-se com o botão direito do mouse é tocada uma música e clicando-se com o botão esquerdo a música é pausada.

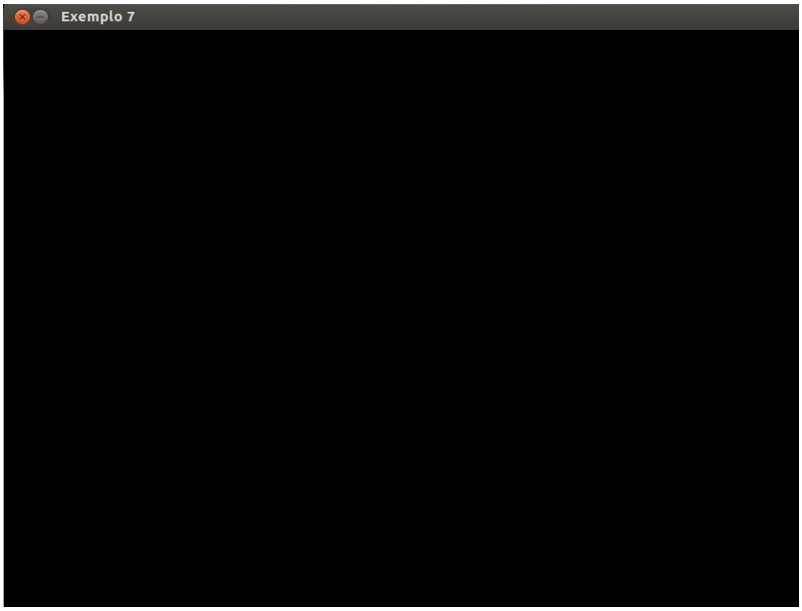


Figura 8: Janela ou com uma música pausada ou sendo tocada

4.8 Exemplo 8: *Capturando posição (x,y) do mouse quando ele é movido*

Neste exemplo, é capturada as coordenadas (x, y) do mouse e impresso na tela seus valores.

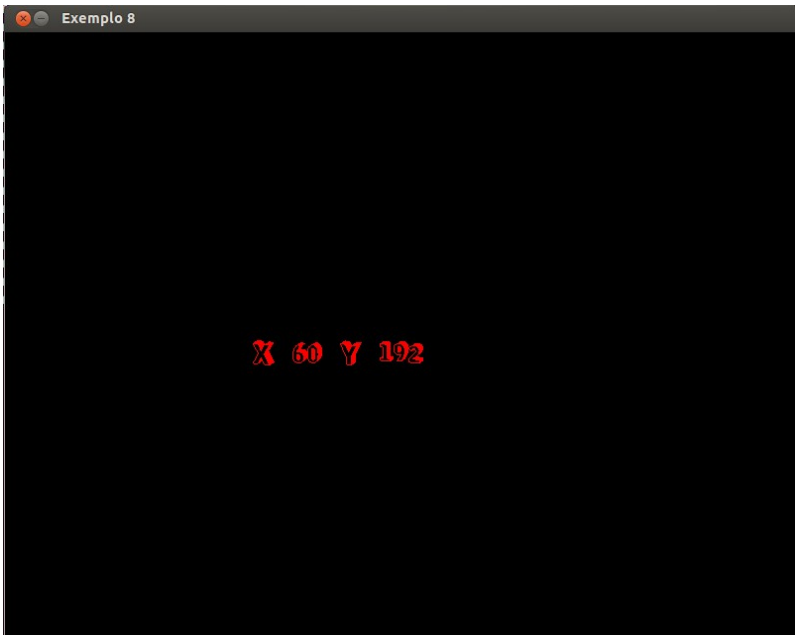


Figura 8: Janela com as coordenadas correntes do mouse sendo mostrada

4.9 Exemplo 9: *Realizando uma animação simples*

Neste exemplo, a animação de uma bola caindo é mostrada na tela.

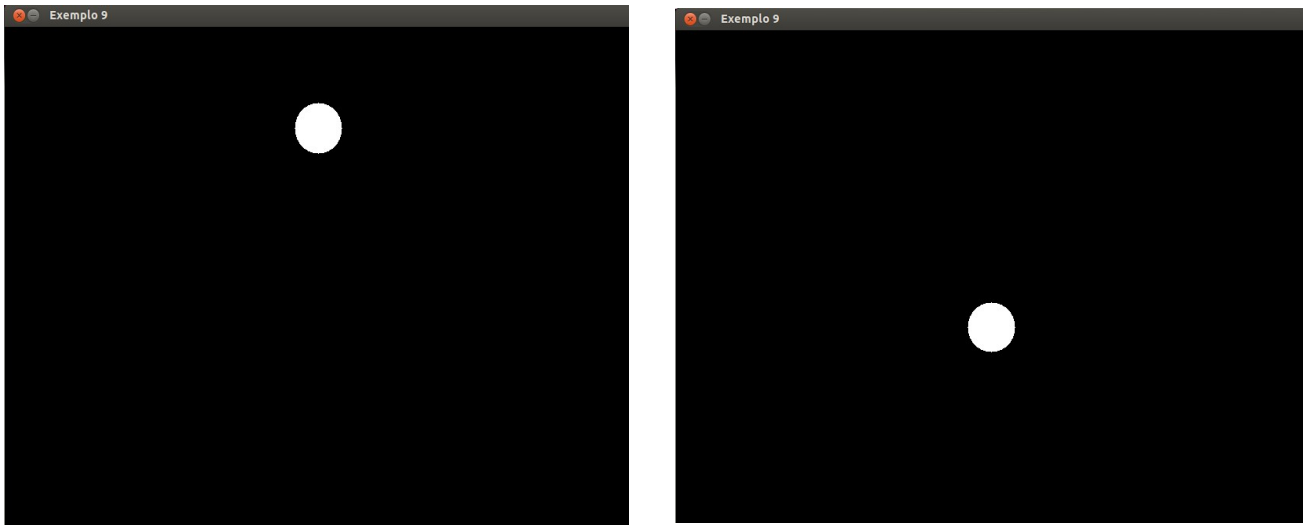


Figura 9: Bola Caindo

4.10 Exemplo 10: *Interagindo com o teclado*

Neste exemplo, se for digitado alguma das teclas LEFT, UP, RIGHT ou DOWN, seu respectivo nome será mostrado na tela.

Figura 10: Cada imagem mostra o nome da respectiva tecla pressionada no teclado

4.11 Exemplo 11 e 12: *Pac Man com teclado e com joystick*

Neste exemplo, o personagem ***Pac Man*** é desenhado na tela e é possível movê-lo usando-se as teclas UP, RIGHT, LEFT e DOWN e os direcionais do joystick. A cada movimento do Pac Man, um som também é tocado.

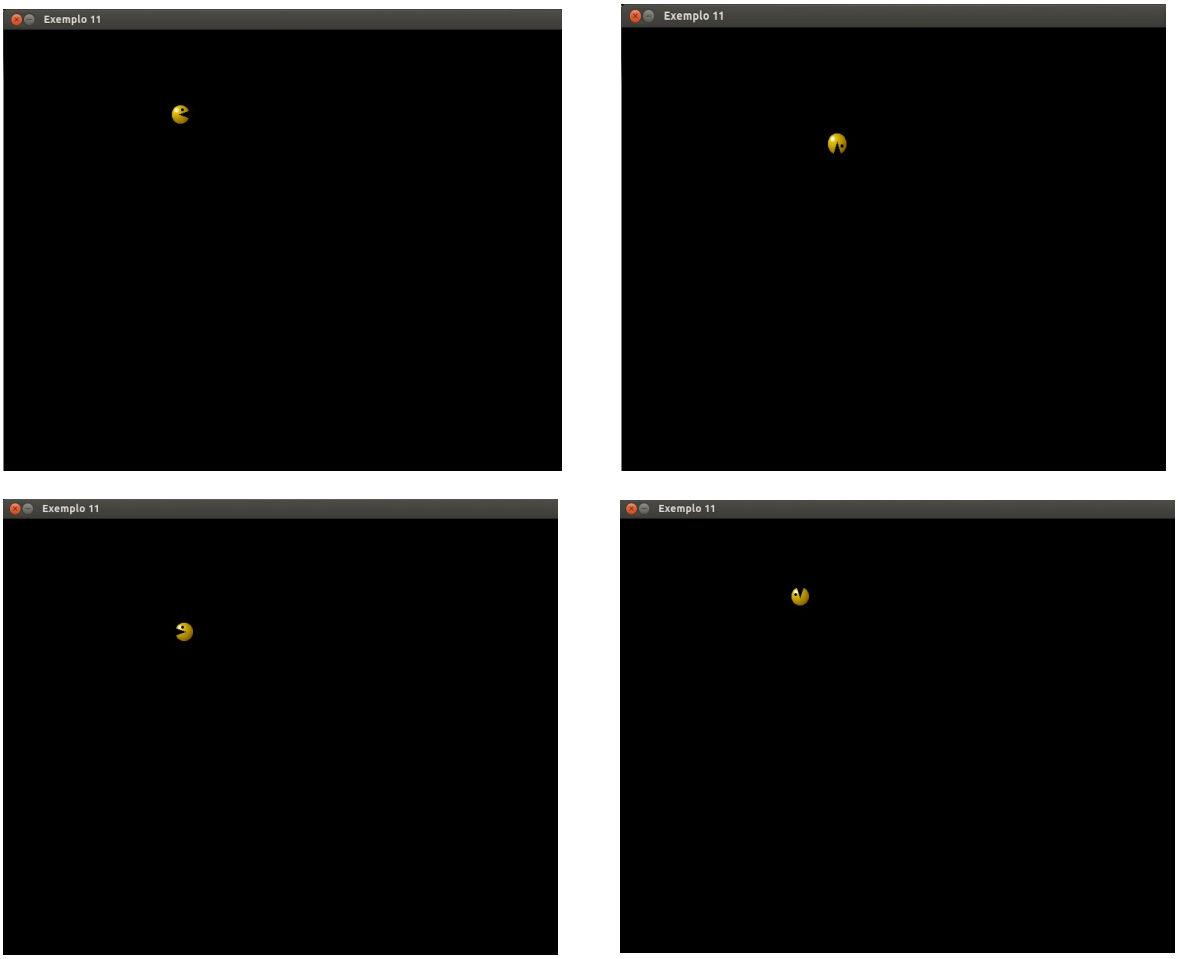
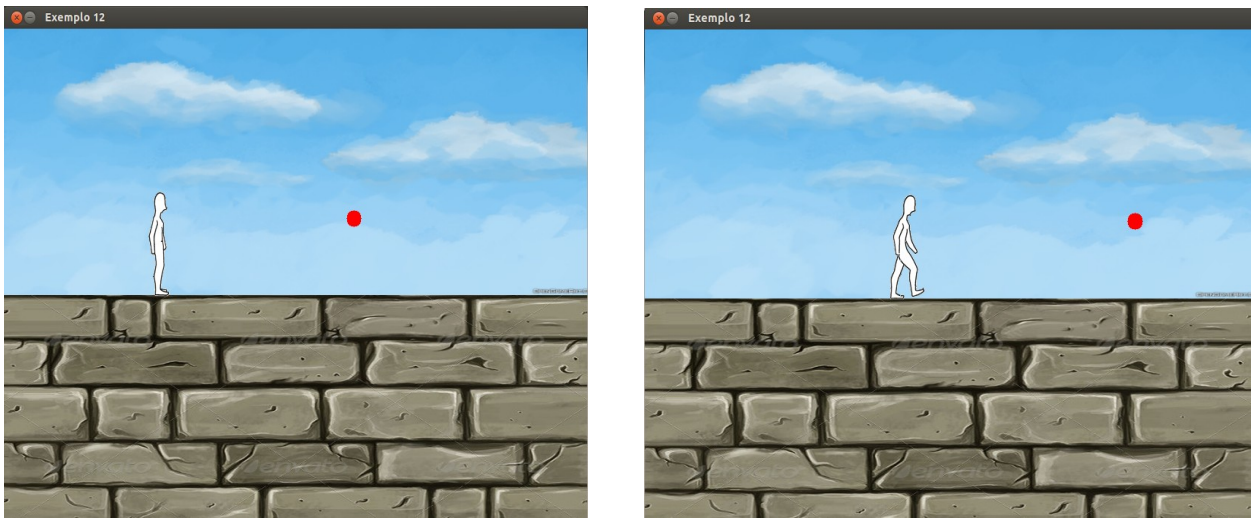


Figura 11: Movimentos do Pac man de acordo com a tecla digitada

4.12 Exemplo 13: *Joguinho com seu próprio personagem*

Neste exemplo, um joguinho foi criado com um personagem que realiza alguns movimentos como andar, pular, abaixar e morrer. O objetivo é não deixar uma bola vermelha lhe atingir.



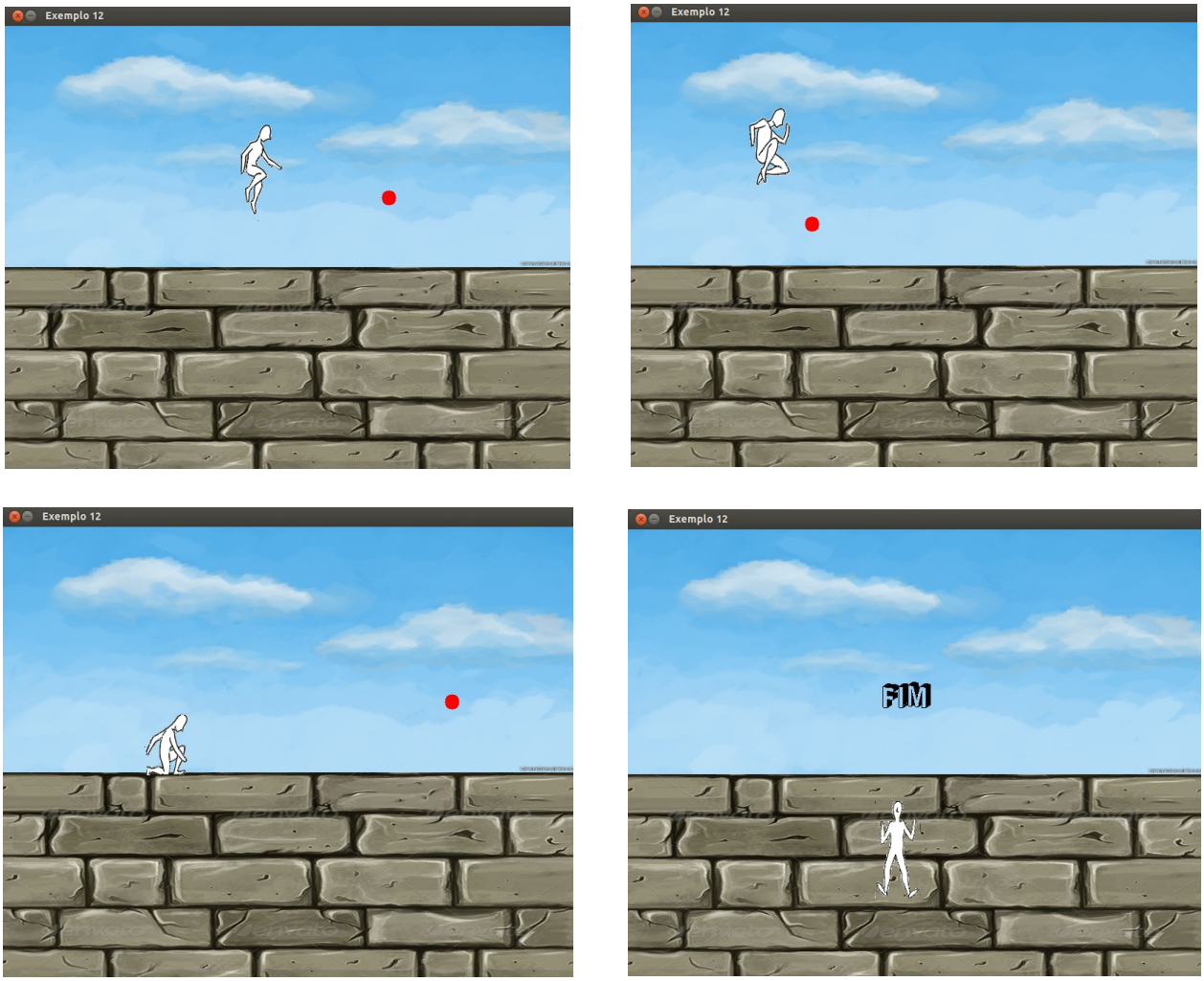


Figura 12: Movimentos do personagem quando uma tecla é pressionada ou quando o jogo termina