visionlab
visualization laboratory

Pontifícia Universidade Católica
do Rio de Janeiro

Introdução a Programação de Jogos

Aula 07 – Utilizando Imagens na PlayLib

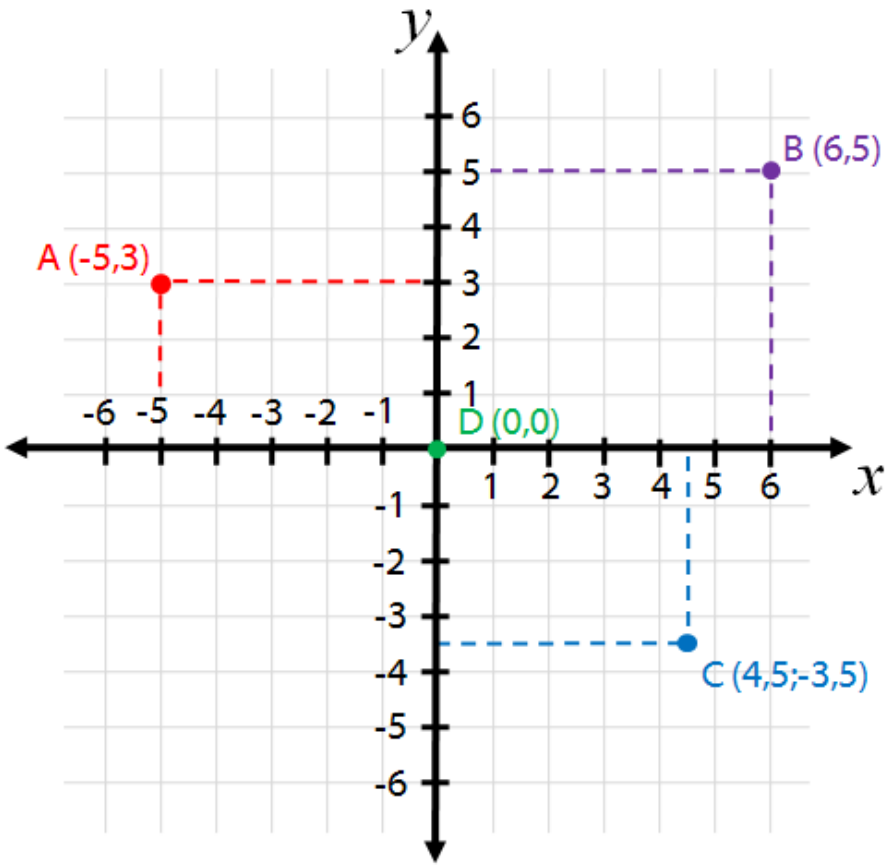
Prof. Augusto Baffa
< abaffa@inf.puc-rio.br>

Biblioteca Gráfica - PlayLib

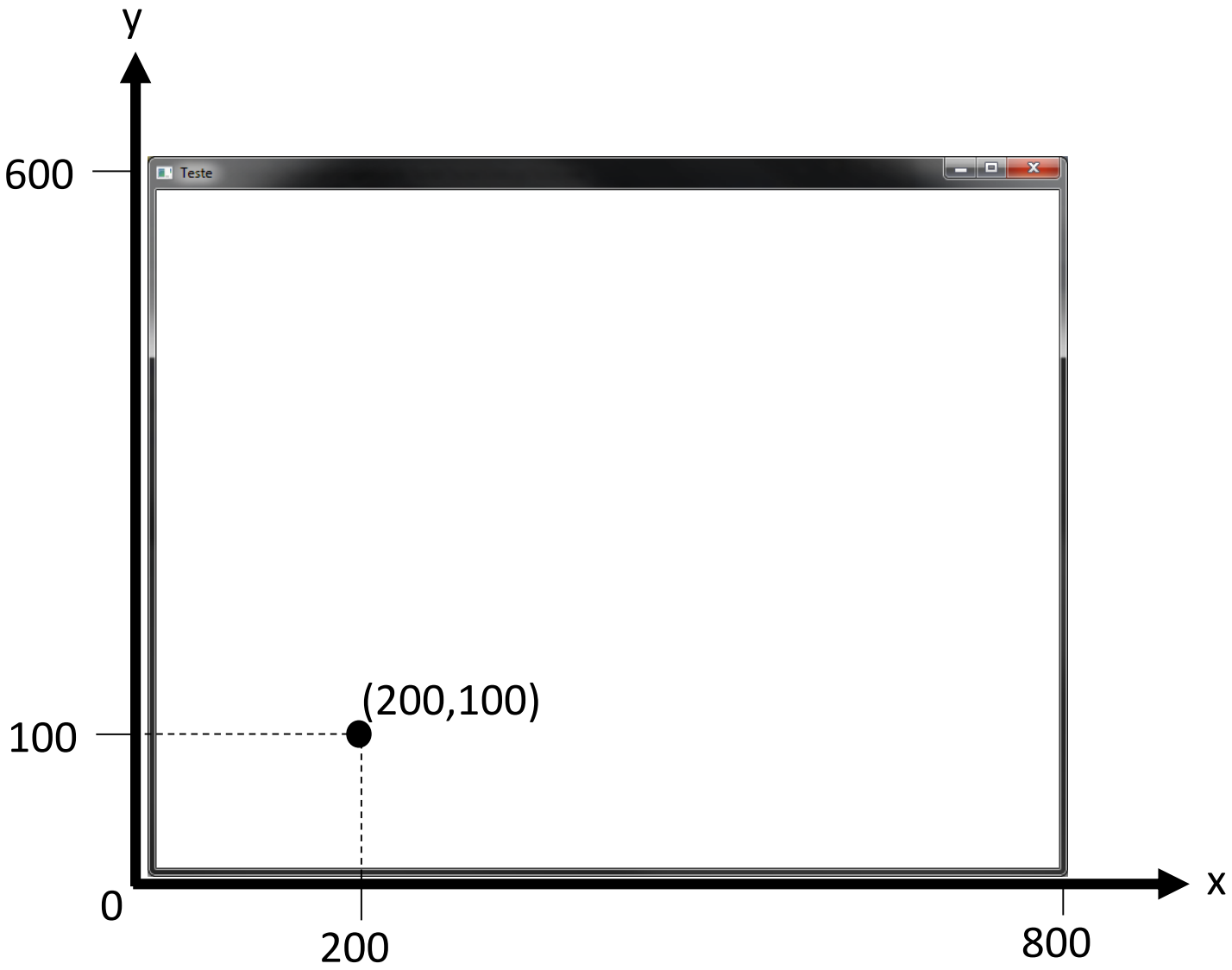
- **Conjunto de funções** para criação e manipulação de formas geométricas, imagens, áudio, janelas...
- Baseada na API **OpenGL**.
- Pode ser usada para criação de **jogos 2D**, **simulações**, **animações** e outros aplicativos.
- **Desenvolvida especialmente para esse curso!**

Coordenadas de Tela

- Sistema de Coordenadas Cartesiano
- Duas dimensões (2D)
- Coordenas X e Y



Coordenadas de Tela



Desenhando Imagens

- **Para desenhar uma imagem na tela é necessário:**
  - **(1)** Criar uma variável do tipo **Image**.
  - **(2)** Carregar a imagem do arquivo usando o comando **LoadPNGImage**.
  - **(3)** Desenhar efetivamente a imagem na tela usando o comando **DrawImage2D**.

Desenhando Imagens

- **(1) Criar uma variável do tipo Image:**

```
Image minha_imagem;
```

OBS: Sempre declare as variáveis Image como **variáveis globais**.

Exemplo:

```
#include "Graphics.h"
using namespace GraphicsLib;

Graphics graphics;
Image minha_imagem1;
Image minha_imagem2;

int main(void)
{
...
```

Variáveis Image declaradas no inicio do programa. Antes e fora da função principal ou outras funções.

Desenhando Imagens

- **(2) Carregar a imagem do arquivo usando o comando LoadPNGImage:**

```
minha_imagem.LoadPNGImage("Mario.png");
```

Exemplo:

```
int main(void)
{
...
  minha_imagem.LoadPNGImage("Mario.png");
...
}
```

Carrega a imagem do arquivo **Mario.png** para a variável minha_imagem.

OBS: Cada imagem deve ser carregada **apenas uma vez**. Por isso, nunca carregue a imagem diretamente de dentro do Loop Principal.

Desenhando Imagens

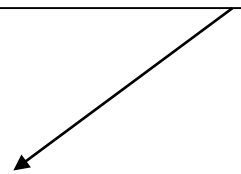
- **(3) Desenhar efetivamente a imagem na tela usando o comando DrawImage2D.**

```
graphics.DrawImage2D(200, 200, 256, 256, minha_imagem);
```

Exemplo:

```
void MainLoop()
{
...
  graphics.DrawImage2D(200, 200, 256, 256, minha_imagem);
...
}
```

Desenha a imagem “minha_imagem” na posição (200, 200) com tamanho (256, 256) na tela.



Desenhando Imagens

- **Também é possível definir a posição e tamanho das imagens em variáveis armazenadas dentro do objeto Image. Para isso, deve-se:**

1) Criar uma variável do tipo Image:

```
Image minha_imagem;
```

2) Carregar a imagem do arquivo usando o comando LoadPNGImage:

```
minha_imagem.LoadPNGImage("Mario.png");
```

3) Definir a posição da imagem com o comando SetPosition:

```
minha_imagem.SetPosition(100,100,256,256);
```

4) Desenhar a imagem na tela com o comando DrawImage2D:

```
graphics.DrawImage2D(minha_imagem);
```

Desenhando Imagens

- **Carregando uma Imagem:**

```
void Image.LoadPNGImage(char *filename);
```

Exemplo:

```
Image mario;
mario.LoadPNGImage("Mario.png");
```

Declaração da variável do tipo Image que vai armazenar a imagem

Carrega o arquivo “Mario.png” para a variável “mario”

Desenhando Imagens

- **Desenhando Imagens na Tela:**

```
void DrawImage2D(int x, int y, int width, int height, Image image);
void DrawImage2D(Image image);
void DrawImage2D(int x, int y, int width, int height, int crop_x,
       int crop_y, int crop_width, int crop_height, Image image);
```

Exemplo:

```
graphics.DrawImage2D(200, 200, 256, 256, mario);
```

Desenha a imagem “mario” na posição (200, 200) com tamanho (256, 256) na tela.

Desenhando Imagens

- **Definindo a Posição uma Imagem:**

```
void Image.SetPosition(int x, int y, int width, int height);
```

Exemplo:

```
mario.SetPosition(200, 200, 256, 256);
```

Define a posição da imagem "mario" na posição (200, 200) com tamanho (256, 256)

Desenhando Imagens

- **Observações importantes sobre imagens:**
  - **Somente são aceitas imagens no formato PNG**. Mas isso não é uma limitação, o formato PNG é um dos melhores formatos para esse tipo de aplicação. A principal vantagem é que ele permite o uso de **transparência** nas imagens.
  - Cerifique-se de que as imagens que serão lidas estão **dentro da pasta do seu projeto do Visual Studio**. Se preferir armazena-las em outras pastas você deve fornecer o caminho completo para o diretório onde as imagens estão para o comando LoadPNGImage.
  - Se a sua imagem estiver em **outro formado** (JPG, GIF, BMP...) você deve convertê-la para o formato PNG antes de carrega-la.

Tratando Entradas do Teclado

- Para poder tratar os eventos gerados pelo teclado (**teclas sendo pressionadas**) é necessário criar uma função para essa tarefa.

- Essa função deve ter a seguinte sintaxe:

```
void KeyboardInput(int key, int state, int x, int y))
{
   /* Bloco de Comandos */
}
```

- Também é **necessário indicar** que essa é a sua função para tratar eventos de teclado:

```
graphics.SetKeyboardInput(KeyboardInput);
```

Tratando Entradas do Teclado

- Dessa forma, sempre que uma tecla normal do teclado for pressionada a função **KeyboardInput** será executada e o parâmetro key indicará qual tecla foi pressionada. O parâmetro state indicará se a tecla foi pressionada ou liberada.
- **Exemplo:**

```
void KeyboardInput(int key, int state, int x, int y))
{
  if ((key == 'f')&&(state == KEY_STATE_DOWN))
  {
    graphics.SetFullscreen(true);
  }
  if ((key == KEY_RIGHT)&&(state == KEY_STATE_DOWN))
  {
    posicao_personagem_x = posicao_personagem_x + 2;
  }
  if ((key == KEY_ESC)&&(state == KEY_STATE_DOWN))
  {
    exit(0);
  }
}
```

Se a letra f for pressionada

Coloca o programa em tela cheia

Se a seta da direita for pressionada

Incrementa em +2 uma variável que representa a posição de um personagem

Se a tecla esc for pressionada

Fecha o programa

Códigos das Teclas Especiais

- KEY_LEFT
- KEY_UP
- KEY_RIGHT
- KEY_DOWN
- KEY_PAGE_UP
- KEY_PAGE_DOWN
- KEY_HOME
- KEY_END
- KEY_INSERT
- KEY_ESC
- KEY_ENTER

- KEY_BACKSPACE
- KEY_LEFTCTRL
- KEY_RIGHTCTRL
- KEY_LEFTSHIFT
- KEY_RIGHTSHIFT
- KEY_LEFTALT
- KEY_RIGHTALT
- KEY_TAB
- KEY_F1
- KEY_F2
- KEY_F3

- KEY_F4
- KEY_F5
- KEY_F6
- KEY_F7
- KEY_F8
- KEY_F9
- KEY_F10
- KEY_F11
- KEY_F12

Estados das teclas:

- KEY_STATE_DOWN
- KEY_STATE_UP

Tratando Cliques do Mouse

- Para poder tratar os eventos gerados pelo mouse (**cliques do mouse**) é necessário criar uma função para essa tarefa.

- Essa função deve ter a seguinte sintaxe:

```
void MouseClickInput(int button, int state, int x, int y)
{
   /* Bloco de Comandos */
}
```

- Também é **necessário indicar** que essa é a sua função para tratar eventos de clique do mouse:

```
graphics.SetMouseClickInput(MouseClickInput);
```

Tratando Cliques do Mouse

- Dessa forma, sempre que um botão do mouse for pressionado a função **MouseClickInput** será executada e o parâmetro button indicará qual botão foi pressionado. Os parâmetros x e y indicam a posição na tela em que mouse estava quando o clique foi realizado.

- **Exemplo:**

```
void MouseClickInput(int button, int state, int x, int y)
{
  if ((button == MOUSE_LEFT_BUTTON)&&(state == MOUSE_STATE_DOWN))
  {
     destino_x = x;
     destino_y = y;
  }
}
```

Se o botão esquerdo foi pressionado

As variáveis destino_x e destino_y recebem a posição x e y do mouse no momento do clique, ou seja, onde o usuário clicou.

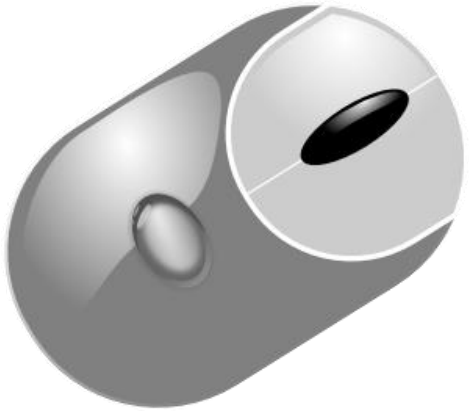
Códigos da Teclas do Mouse

Botões:

- MOUSE_LEFT_BUTTON
- MOUSE_MIDDLE_BUTTON
- MOUSE_RIGHT_BUTTON

Estados:

- MOUSE_STATE_DOWN
- MOUSE_STATE_UP



Tratando o Movimento do Mouse

- Para poder tratar os eventos de movimento gerados pelo mouse é necessário criar uma função para essa tarefa.

- Essa função deve ter a seguinte sintaxe:

```
void MouseMotionInput(int x, int y)
{
   /* Bloco de Comandos */
}
```

- Também é **necessário indicar** que essa é a sua função para tratar eventos de movimento do mouse:

```
graphics.SetMouseMotionInput(MouseMotionInput);
```

Tratando o Movimento do Mouse

- Dessa forma, sempre que o mouse for movimentado pelo usuário a função **MouseClickInput** será executada e os parâmetros x e y indicaram a posição do mouse na tela.

- **Exemplo:**

```
void MouseMotionInput(int x, int y)
{
  mouse_x = x;
  mouse_y = y;
}
```

As variáveis mouse_x e mouse_y recebem a posição x e y do mouse, ou seja, o local onde o usuário está com o cursor do mouse.

Tratando Cliques do Mouse Sobre uma Imagem

- Para poder tratar os eventos de clique do mouse sobre uma determinada imagem é necessário definir uma função para essa tarefa.

- A função para tratar esse evento deve ter a seguinte sintaxe:

```
void MouseClickMinhaImagem(int button, int state, int x, int y)
{
   /* Bloco de Comandos */
}
```

- Também é necessário indicar que essa é a sua função para tratar eventos de clique do mouse sobre a imagem em questão usando o comando SetOnClick:

```
MinhaImagem.SetOnClick(MouseClickMinhaImagem);
```

Tratando Cliques do Mouse Sobre uma Imagem

- Dessa forma, sempre que o usuário clicar sobre a imagem “MinhaImagem”, a função MouseClickMinhaImagem será executada e o parâmetro button indicará qual botão foi pressionado. Os parâmetros x e y indicam a posição na tela relativa a imagem em que mouse estava quando o clique foi realizado.

- **Exemplo:**

```
void MouseClickMinhaImagem(int button, int state, int x, int y)
{
  carregando_imagem = true;
}
```

Tratando Cliques do Mouse Sobre uma Imagem

- **Importante:** Para poder usar este evento é necessário que a posição da imagem tenha sido definida com o comando SetPosition.

- **Exemplo:**

```
Image minha_imagem;

void MouseClickMinhaImagem(int button, int state, int x, int y)
{
  clicou_na_imagem = true;
}

int main(void)
{
  ...
  minha_imagem.LoadPNGImage("Marvin.png");
  minha_imagem.SetPosition(0,100,256,256);
  minha_imagem.SetOnClick(MouseClickMarvin);
  ...
}
```

Exercícios

Lista de Exercícios 06 – Imagens e Interação

http://www.inf.puc-rio.br/~abaffa/iue1503/