Guia de Referência da playAPC

Versão 1.6

playAPC

Dedication

Dedico à todos os alunos de Ciência da Computação que querem fazer trabalhos divertidos na primeira matéria de programação.

Guia de Referência da playAPC: Versão 1.6

This electronic book was generated by Anthologize

ACKNOWLEDGEMENTS

Agradeço ao professor Ralha, que sempre ofereceu apoio e entusiasmo ao projeto, além do grande auxílio no desenvolvimento tanto na própria biblioteca como material de ensino; ao professor Zaghetto pela oportunidade de PIBIC e pela confiança ao aplicar a biblioteca em uma turma de APC, além de estar sempre sugerindo novas ideias de funcionalidades; e à todos os alunos de APC que retornaram comentários sobre o uso da biblioteca.

INTRODUÇÃO

Guia de referência da playAPC

A playAPC é uma biblioteca educacional de programação voltada para os alunos de Algoritmos e Programação de Computadores (*APC*) da UnB que tem como objetivo ilustrar, de maneira gráfica, os conceitos aprendidos durante o curso com aplicações simples. Com ela, é possível criar animações simples e desenhos estáticos.

A biblioteca consiste em um conjunto de funções para criação e manipulação de formas 2D e utilização de imagens. Utiliza a API OpenGL e GLFW.

A playAPC foi desenvolvida usando o conceito de Orientação a Objetos de C++, por praticidade de desenvolvimento e facilidade de encapsulamento. Entretanto, a playapc.h entrega ao aluno uma interface amigável com chamadas de funções simples para a criação de todos os objetos, utilizando o paradigma Imperativo.

Neste guia, você encontrará todas as funções da playAPC explicadas e exemplificadas, a fim de facilitar o uso da biblioteca

Requisitos

Esta biblioteca está configurada para executar juntamente com a GLFW 2.7 e qualquer versão da OpenGL

Download

Clique aqui para baixar a versão mais recente da playAPC.

Changelog

1.6 (15/03/2016)

- Mudança de nome

1.5.1 (11/03/2016)

- Sobrecarga do operadores +, -, =, ==, +=, -= para estruturas do tipo Ponto

1.5 (24/02/2016)

- Adicionado função CriaGrafico()
- Corrigido problemas de proporção

1.4.1 (04/09/2015)

- Adicionado função RetornaTecla()
- Consertado outros bugs menores

1.4 (12/08/2015)

- Modificação da função Move()
- Criação da função de redisionamento de limites da projeção
- Criação da geometria Elipse
- Integração com biblioteca SOIL
- Reformulação do Guia de Referência

1.3.1 (13/03/2015)

- Enumeração de geometrias válida
- Funções de criar geometrias retorna índice para cada geometria criada
- Função Pintar recebe nome de geometria e índice de geometria

1.3 (06/06/2014)

- Reformulação da animação (qualquer grupo pode ser animado)
- Inserção de super grupos de grupos
- Reformulação das operações de transformação (para suportar realizar a operação dado um grupo)

1.2 (23/05/2014)

- Renderização de animações simples (apenas o último grupo poderia ser animado)
- Reformulação das operações de transformação
- Input de tecla

1.1 (24/03/2014)

- Operações de transformação (move, gira, redimensiona)
- Inserção de grupos de geometrias

1.0 (11/01/2014)

- Renderização de objetos geométricos

COMO INSTALAR

Windows

Antes da instalação da bibliotecaImage not found

Considerações iniciais

Este guia de instalação considera os caminhos dos diretórios definidos por padrão no momento da instalação. Se atente a este detalhe antes de seguí-lo.

A playAPC foi testada no Windows 10, 8, 7 e XP.

Antes de iniciar o processo de instalação, desative o seu antivírus. Alguns antivírus consideram os binários da playAPC como falso-positivo.

Code::Blocks

A utilização de uma IDE não interfere na instalação de uma biblioteca. Para o curso de Algoritmo e Programação de Computadores da UnB, em geral opta-se pelo uso do Codeblocks no caso de usuários de Windows.

Na , baixe a versão *codeblocks-16.01mingw-setup.exe*, que é a opção que instala, juntamente com o Code::Blocks, o compilador MinGW. O compilador será instalado no diretório *C:\Program Files (x86)\Codeblocks\MinGW*

Após a instalação do Code::Blocks com o compilador, prossiga para **Instalação da biblioteca**.

Caso você opte por usar outra IDE ou gostaria de manter compiladores separados para cada aplicação (seja por organização ou outro motivo), sugiro a instalação do compilador MinGW por ser Open Source, mas nada impede de instalar outro compilador, uma vez que suas estruturas de pastas devem ser similares.

MinGW

Baixe o (o botão de download está no canto superior direito).

Na janela que abriu, **MinGW Installation Manager**, selecione **Basic Setup** e marque as opções:

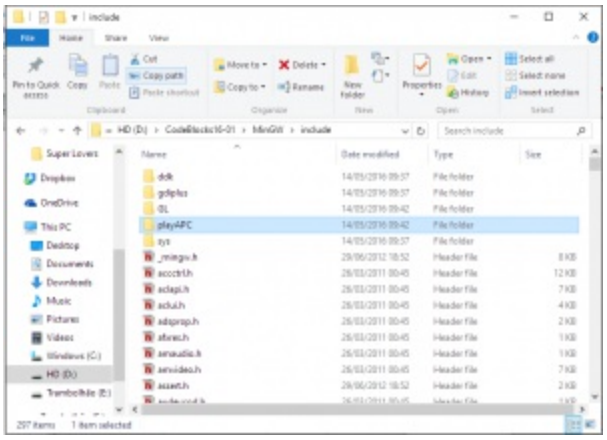
- mingw-developer-toolkit
- mingw32-base
- mingw32-gcc-g++
- msys-base

Na aba Installation, selecione a opção **Apply changes**. Os arquivos necessários serão baixados da rede e instalados no diretório *C:\MinGW*.

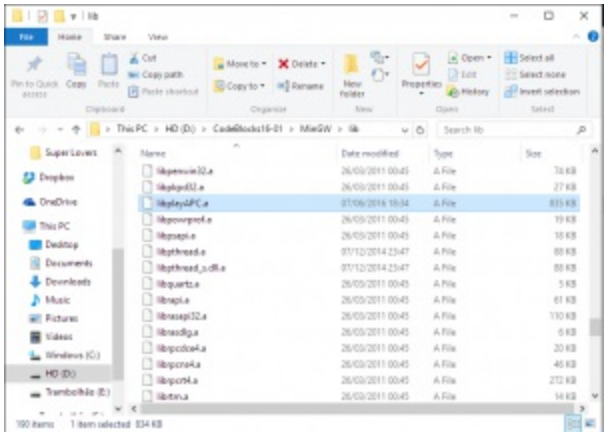
Instalação da biblioteca

Acesse a página de download dos arquivos da . Baixe somente o arquivo ***playAPC_windows.zip***. Este arquivo possui os arquivos binário da biblioteca, este tutorial de forma resumida (*tutorial_Windows.txt*) e um arquivo de teste chamado *teste.cpp*.

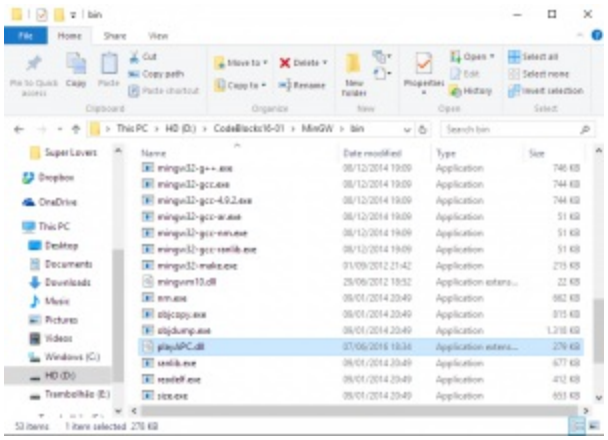
1. Ao terminar de baixar o arquivo zipado do Windows, haverão três pastas importantes
   - include
   - bin
   - lib
2. Abra a pasta do seu compilador (***C:\MinGW*** ou ***C:\Program Files (x86)\Codeblocks\MinGW***) e note que também haverão pelo menos as três pastas
   - include
   - bin
   - lib
3. Copie todo o conteúdo da pasta include da playAPC e cole na pasta include do MinGW. Faça o mesmo para o bin e o lib.
   - **nota:** na pasta include do MinGW **NÃO** é para ter uma outra pasta include (o mesmo vale pra bin e lib). Copie somente os *arquivos respectivos de cada pasta*.



Exemplo de como fica a pasta
C:\MinGW\include



Exemplo de como fica a pasta C:\MinGW\lib

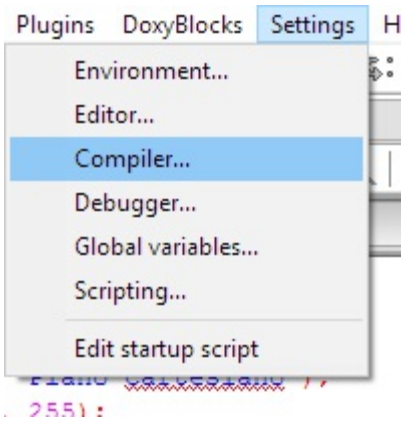


Exemplo de como fica C:\MinGW\bin

Após esse passo, a biblioteca está instalada.

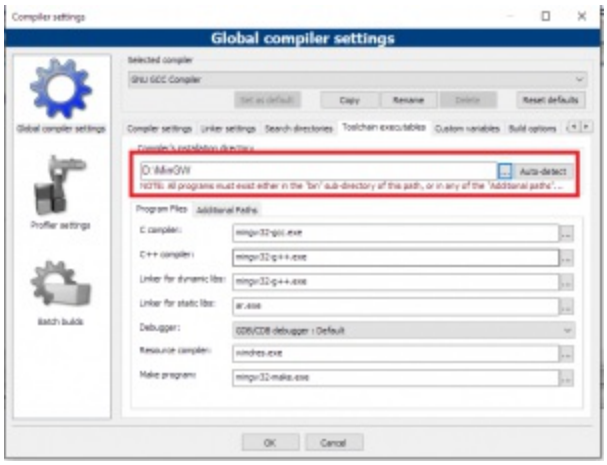
Configurando a biblioteca para ser utilizada no Code::Blocks

1. Na barra de menu do Code::Blocks, selecione Settings->Compiler…



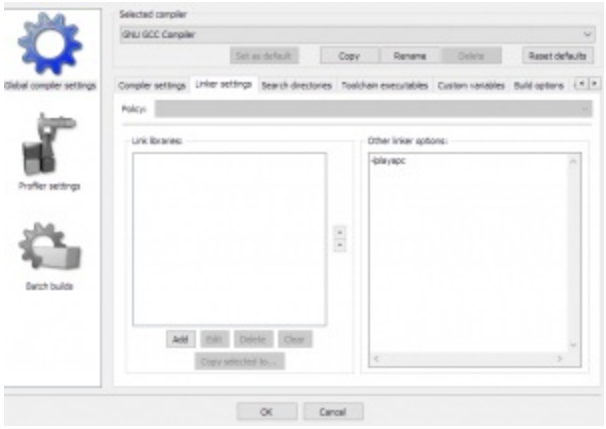
Definições do compilador

2. Se o seu MinGW está dentro do Code::Blocks (*C:\Program Files (x86)\Codeblocks\MinGW*), prossiga pro passo 3. Caso você tenha baixado o MinGW (*C:\MinGW*), em *Toolchain executables*, na caixa *Compiler's installation directory*, selecione o diretório de onde está instalado o MinGW



Definindo onde está instalado o MinGW

3. Em *Linker Settings*, na caixa *Other Linker Options*, escreva
   - -lplayapc



Linkando a playAPC

4. Aperte OK e feche o Code::Blocks antes de testar a playAPC.

Testando a playAPC

Para apenas verificar se a playAPC foi instalada com sucesso, dentro da pasta teste, há um arquivo

chamado teste.cpp.

1. Abra o arquivo *teste.cpp* no Code::Blocks (clique duas vezes no arquivo).
2. Aperte a tecla F9.
3. Se aparecer uma janela mostrando um Plano Cartesiano num espaço de -100 a 100, então a playAPC foi instalada com sucesso

Linux

Antes da instalação da biblioteca

A playAPC foi testada em sistemas baseados no Debian, como Ubuntu e suas variações.

Antes de começar a instalar a biblioteca, é necessária baixar todas as dependências dela primeiro utilizando o comando no terminal sudo apt-get.

Abra o terminal e digite:

```
sudo apt-get update

sudo apt-get install g++

sudo apt-get install libglfw-dev
```

Após ter instalado todos estes pacotes, podemos prosseguir com a instalação.

Instalação da biblioteca

Acesse a página de download dos arquivos da . Baixe somente o arquivo ***playAPC_linux.zip***. Este arquivo possui os arquivos binário da biblioteca, este tutorial de forma resumida (*tutorial_Linux.txt*) e um arquivo de teste chamado *teste.cpp*.

1. Ao terminar de baixar o arquivo zipado, haverão duas pastas importantes
    - include
    - lib
2. No terminal, digite o comando 1 para criar a pasta playAPC no seu compilador
    1. *sudo mkdir /usr/local/include/playAPC*
3. Navegue até a pasta include da playAPC usando o comando 1 e depois execute o comando 2. Se tudo foi feito corretamente, os arquivos da pasta include devem sumir. Volte para pasta raiz da playAPC com o comando 3.
    1. *cd include/*
    2. *sudo mv * /usr/local/include/playAPC/*
    3. *cd ../*
4. Navegue até a pasta lib da playAPC com o comando 1 e depois execute o comando 2
    1. *cd lib/*
    2. *sudo mv * /usr/local/lib/*

Configurando o compilador

Verifique se seu compilador tem a pasta usr/local setada no PATH. Para isso, digite o comando:

```
env |grep PATH
```

Caso não encontre usr/local, faça os seguintes procedimentos:

1. Caso o terminal esteja aberto, feche-o e abra-o novamente.
2. Digite o comando 1 para abrir um arquivo no gedit.
    1. *gedit .bashrc*
3. Na última linha deste arquivo, adicione:
    - **export LD_LIBRARY_PATH=/usr/local/lib**
4. Clique em salvar, feche o gedit e digite o comando 1 no terminal
    1. *source .bashrc*
5. Feche o terminal para atualizar as variáveis de ambiente

Testando a playAPC

Para apenas verificar se a playAPC foi instalada com sucesso, dentro da pasta teste, há um arquivo chamado teste.cpp.

1. Abra o terminal e navegue até onde está o arquivo teste.cpp
2. Compile utilizando o comando 1 e execute com o comando 2
    1. *g++ teste.cpp -o teste -lplayAPC*
    2. *./teste*
3. Se aparecer uma janela mostrando um Plano Cartesiano num espaço de -100 a 100, então a playAPC foi instalada com sucesso

Mac OS X

Antes da instalação da biblioteca

Perceba que a instalação da playAPC em ambientes MAC está num processo ainda muito primitivo, portanto, não é garantido que funcione logo pela primeira vez.

A playAPC foi testada no MAC OS X Mavericks e El Captain.

Acesse a página de download dos arquivos da . Baixe somente o arquivo ***playAPC_mac.zip***. Este arquivo possui o código fonte da glfw 2.7.9, o código fonte da playAPC para o Mac, este tutorial de forma resumida (*tutorial_Mac.txt*) e um arquivo de teste chamado *teste.cpp*.

GCC

Verifique se possui o GCC instalado com o comando no terminal:

```
gcc --version
```

Se aparecer a versão do gcc, então prossiga com a instalação da GLFW.

Caso não apareça, deve surgir uma janela sobre *command line tools*. Clique em instalar caso apareça o botão ou procure na AppStore. É gratuito.

GLFW

Vá até a pasta glfw-2.7.9 pelo terminal e verifique se há um arquivo chamado **Makefile**. Compile a biblioteca usando o comando 1 e instale executando o comando 2.

1. make cocoa
2. sudo make cocoa-dist-install

Volte para a pasta raiz da playAPC para continuar com a instalação.

Instalação da biblioteca

1. Navegue até a pasta src da playAPC usando o comando 1.
    1. cd playAPC/src
2. Compile a biblioteca usando o comando 1
    1. make
3. Instale a biblioteca no seu desktop provisoriamente usando o comando 1
    1. sudo make dist-install

Testando a playAPC

Para apenas verificar se a playAPC foi instalada com sucesso, navegue até a pasta teste pelo terminal.

Compilando pela linha de comando

Para compilar o arquivo teste.cpp pelo terminal, escreva:

```
g++ teste.cpp -o teste -I ~/Desktop/Public/playAPC/include -L ~/Desktop/Public/pl
```

Compilando usando makefile

Para compilar o arquivo teste.cpp usando o makefile, escreva:

```
make FILE=teste
```

Perceba que se quiser reaproveitar este makefile para compilar outros programas que usam a biblioteca playAPC, troque o **teste** pelo nome do seu arquivo.cpp

Para executar o programa, escreva no terminal:

```
./teste
```

Se aparecer uma janela 400×400 mostrando um Plano Cartesiano, a playAPC foi instalada com sucesso.

FUNÇÕES ESSENCIAIS

Renderização

Para conseguir exibir na tela todas as geometrias definidas ao longo do programa, a playAPC precisa realizar todo um processamento da CPU (definições da geometria) para a GPU (como elas devem ser tratadas) e enfim a tela do seu computador (como elas são exibidas). Para este processo, é possível utilizar uma das duas funções: *Desenha()* ou *Desenha1Frame()*.

Não há nada contra usar funções como *printf()* ou outras funções de outras bibliotecas entre a chamada de *AbreJanela()* e *Desenha()/Desenha1Frame()*, mas **evite** usar funções de **input que não sejam da playAPC**, como *scanf()*. O motivo é que funções como essa geram interrupções no programa que *congela* todos os outros processamentos, inclusive o processamento gráfico.

Nota 1: A janela de contexto pode ser fechada caso o usuário clique em fechar ou aperte a tecla ESC.

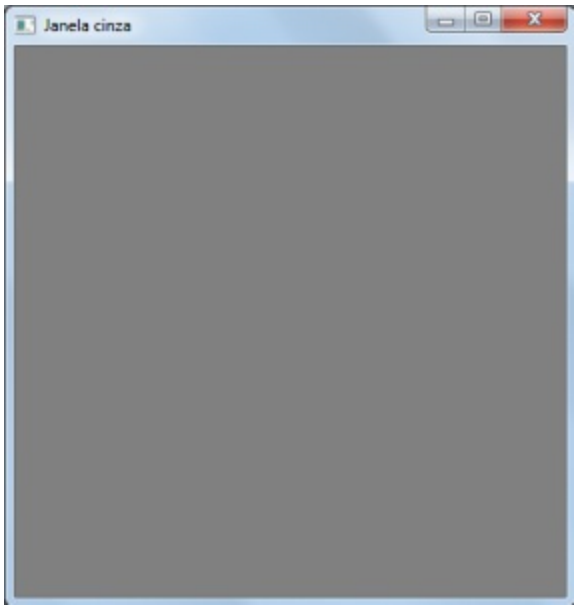
Nota 2: O loop de renderização da função *Desenha()* é realizado numa taxa de 60 frames por segundo.

Renderização de uma cena completa

```
void Desenha()
```

A função continua em execução até a janela ser fechada, impedindo chamada de outras funções.

Não use funções da playAPC **após** a chamada desta função, ou acontecerá *falha de segmentação*.



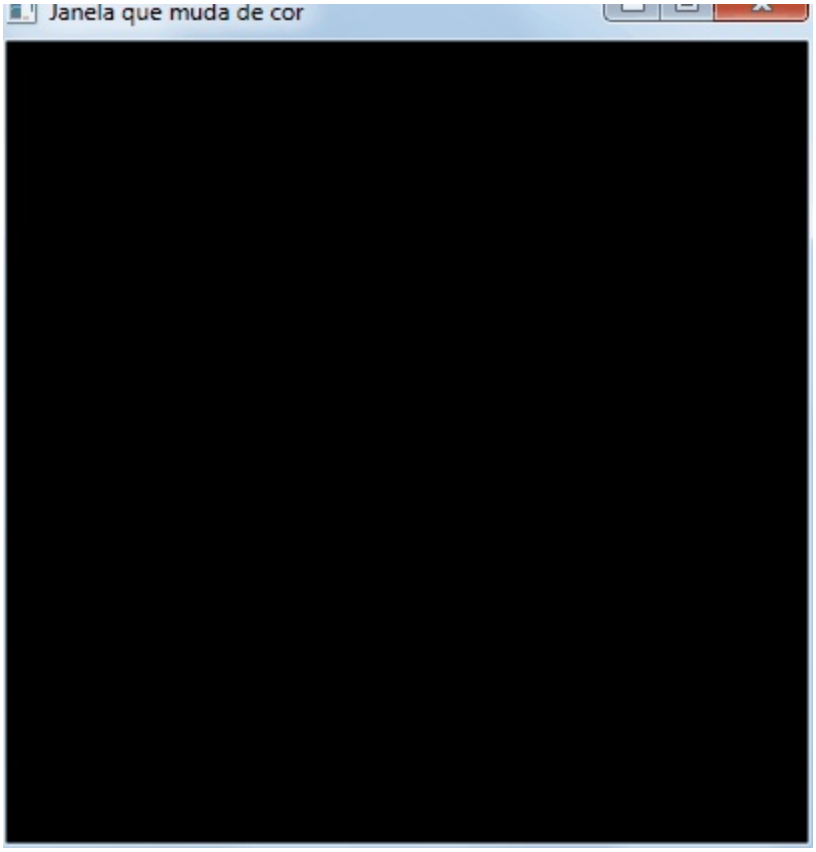
Renderização de um frame

```
int Desenha1Frame()
```

Retorno

1 se a janela continua aberta e *0* se a janela foi fechada.

Esta função não possui controle de fps pois renderiza apenas uma cena, sendo por conta do programador definir uma.



Abrir janela de contexto

Antes de começar a criar as geometrias e usar todas as outras funções da playAPC, é necessário inicializar a OpenGL. Essa inicialização é feita somente abrindo a janela de contexto, a qual também vai alocar na memória todos os endereços relacionados a renderização da cena e inicializar as variáveis da playAPC.

Ou seja, esta função deve estar **antes de qualquer outra função** da playAPC.

Nota: esta função só pode ser chamada uma única vez durante a execução do programa.

Janela de contexto

```
void AbreJanela(float largura, float altura, const char* titulo)
```

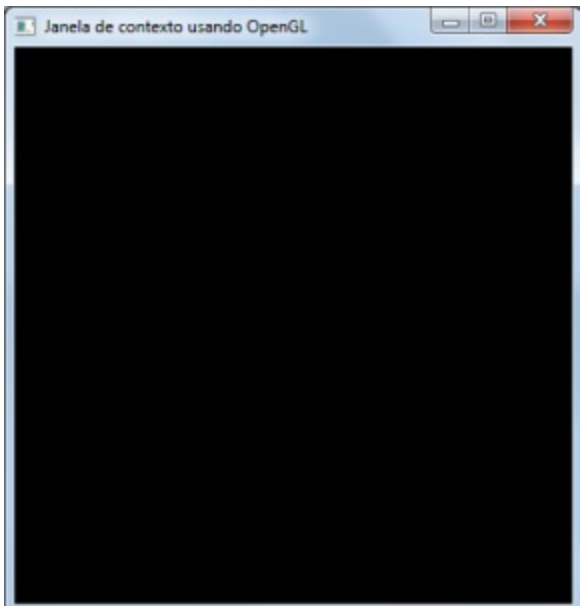
Parâmetros

largura: Largura da janela

altura: Altura da janela

titulo: Nome da janela

Exemplo



GEOMETRIAS

Gráfico

```
int CriaGrafico(short int index, Ponto *p, int verTipo)
```

Parâmetros

index: quantidade de pontos/tamanho do vetor

**p*: vetor de pontos

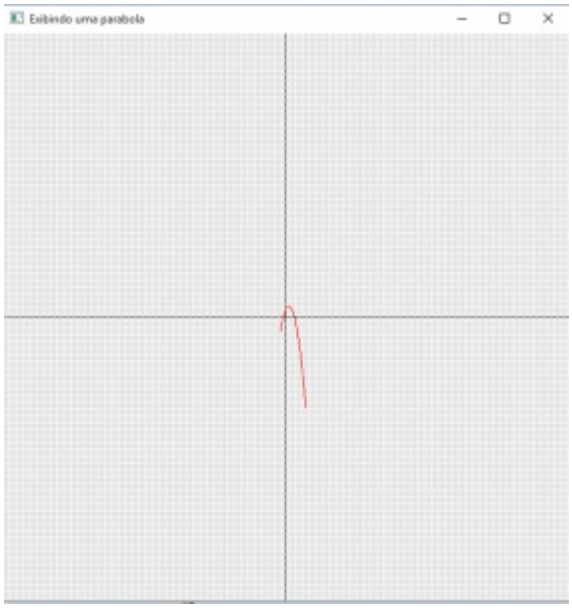
verTipo: tipo de visualização do Gráfico. Atualmente, este parâmetro pode receber os seguintes valores:

Valor	Descrição
0	Não há redimensionamento da janela de exibição
1	A janela se redimensiona de modo a caber todo o gráfico na tela. Podem ocorrer distorções.
2	A janela se redimensiona de modo a caber todo o gráfico na tela. Não ocorre distorções

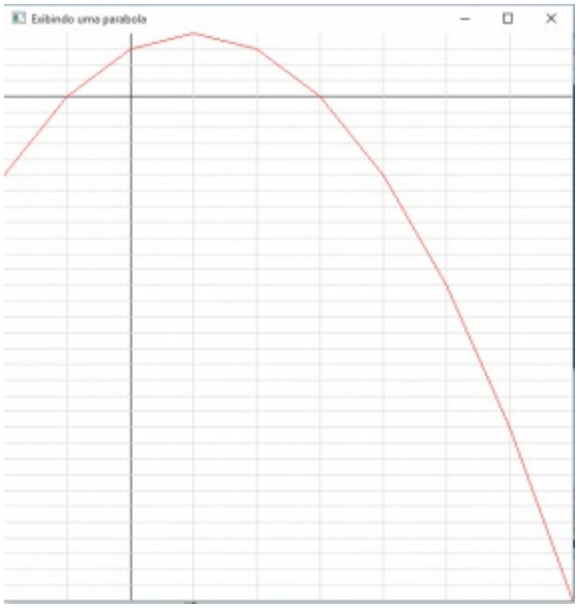
Retorno

Índice da geometria do tipo GRAFICO.

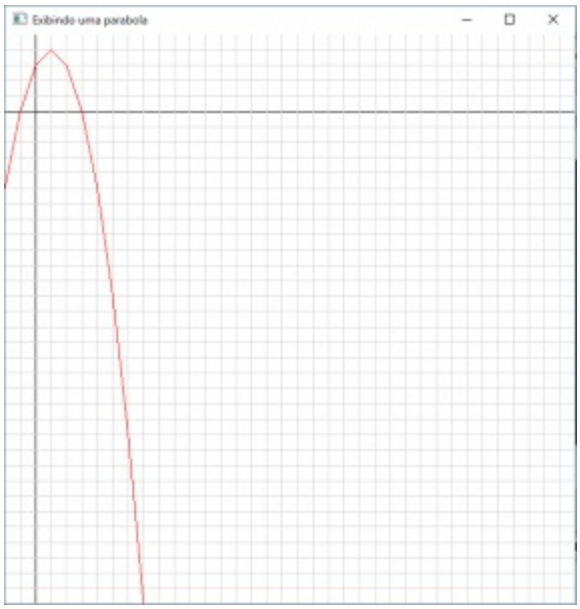
Exemplo 1: Alterando valor de *verTipo* de CriaGrafico



Tipo 0



Tipo 1



Tipo 2

Exemplo 2: Usando retorno de CriaGrafico

Elipse

```
int CriaElipse(float a, float b, Ponto meio)
```

Os parâmetros de criação de elipse seguem a seguinte relação:

Image not found

Onde:

2a: maior eixo da elipse

2b: menor eixo da elipse

O: ponto central

Parâmetros

a: metade do maior eixo

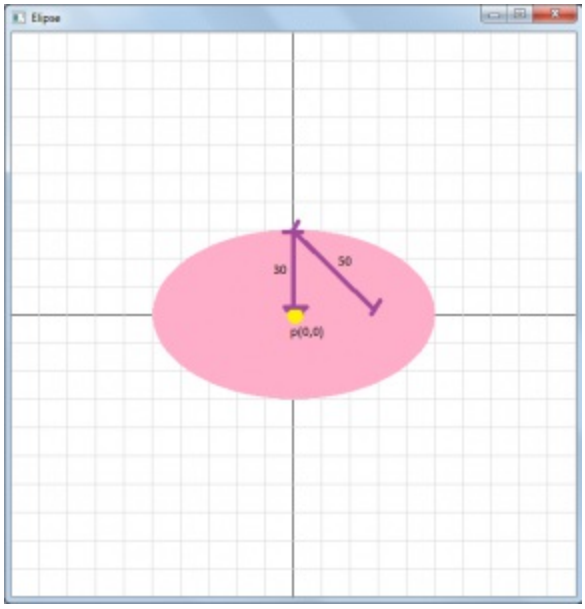
b: metade do menor eixo

meio: ponto central da elipse

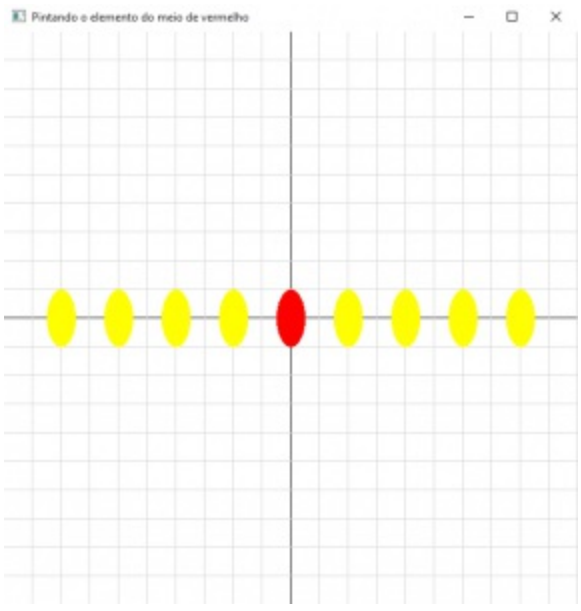
Retorno

Índice da geometria do tipo ELIPSE.

Exemplo 1: Entendendo parâmetros de CriaElipse



Exemplo 2: Usando retorno de CriaElipse



Circunferência

```
int CriaCircunferencia(float raio, Ponto meio)
```

Parâmetros

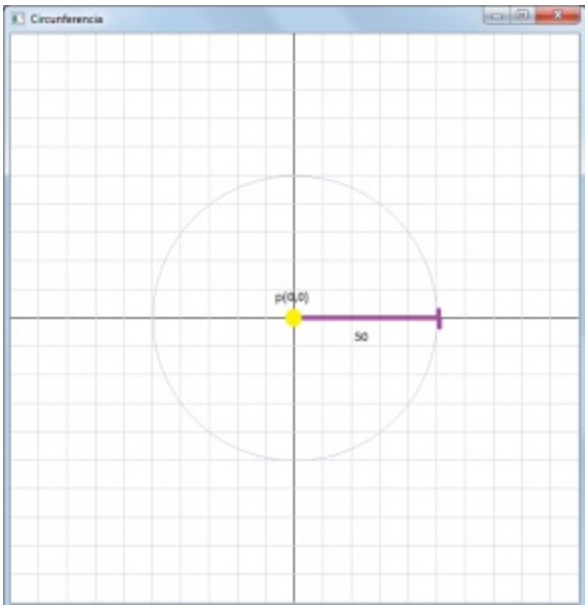
raio: raio da circunferência

meio: coordenadas do centro do circunferência

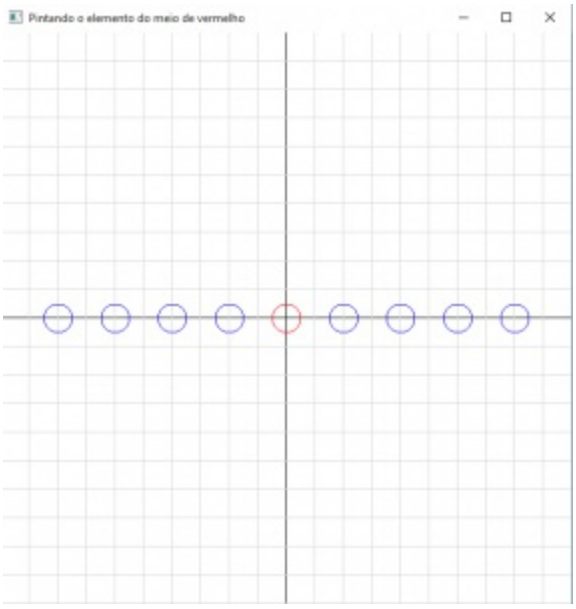
Retorno [Image not found]

Índice da geometria do tipo CIRCUNFERENCIA.

Exemplo 1: Entendendo parâmetros de CriaCircunferencia



Exemplo 2: Usando retorno de CriaCircunferecia



Círculo

```
int CriaCirculo(float raio, Ponto meio)
```

Parâmetros [Image not found]

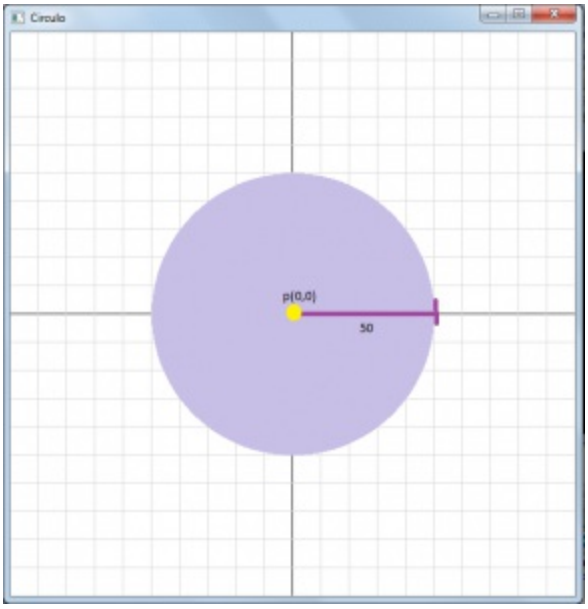
raio: raio do círculo

meio: coordenadas do centro do círculo

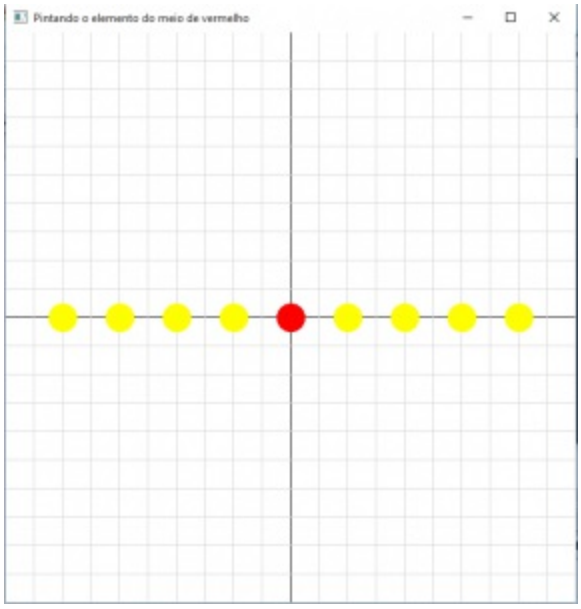
Retorno

Índice da geometria do tipo CIRCULO.

Exemplo 1: Entendendo parâmetros de CriaCirculo



Usando retorno de CriaCirculo



Polígono

Para a criação de Polígono, há na realidade duas funções que criam esta geometria e seu funcionamento é parecido. Em ambas, o usuário passa uma série de Pontos definindo os **vértices** da geometria.

As arestas do polígono são construídas tendo em base a ordem dos vértices passados.

nota: a playAPC só renderiza geometrias ***convexas***.

Criação de polígono passando ponto a pontoImage not found

```
int CriaPoligono(short int qtd, Ponto p1, Ponto p2,...)
```

Parâmetros

qtd: quantidade total de pontos que serão passados

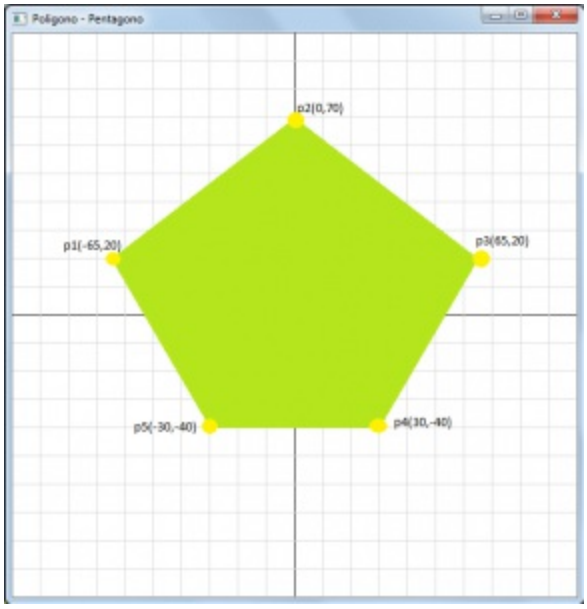
...: variáveis do tipo Ponto que definem os vértices da geometria.

A quantidade de variáveis do tipo Ponto passadas por esta função deve variar de acordo com o valor de *qtd*.

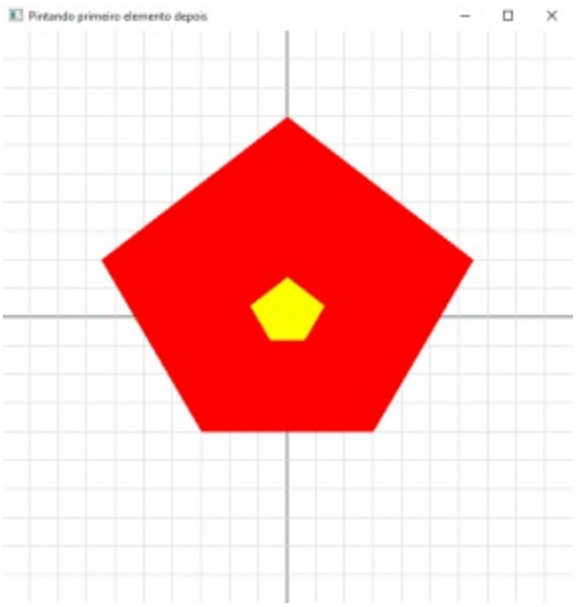
Retorno

Índice da geometria do tipo POLIGONO.

Exemplo 1: Entendendo parâmetros de CriaPoligono



Exemplo 2: Usando retorno de CriaPoligono



Criação de polígono passando vetorImage not found

```
int CriaPoligonoVetor(short int index, Ponto *p)
```

Parâmetros

index: tamanho do vetor

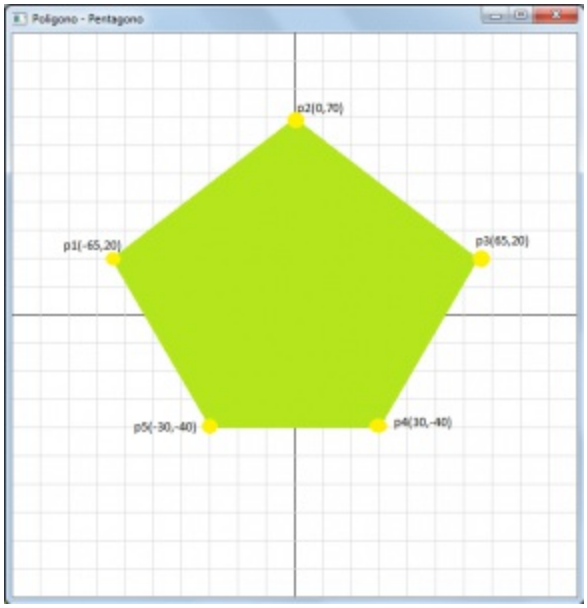
**p*: vetor de Pontos

O tamanho do vetor de Pontos deve ser exatamente igual ao valor de *index*.

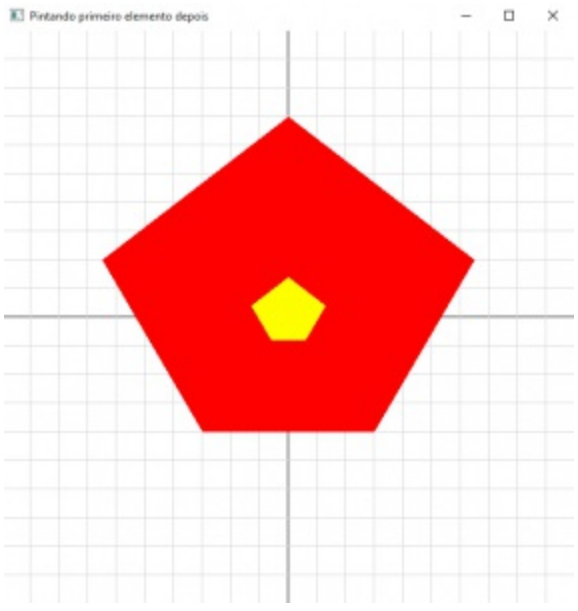
Retorno

Índice da geometria do tipo POLIGONO.

Exemplo 1: Entendendo parâmetros de CriaPoligonoVetor



Exemplo 2: Usando retorno de CriaPoligonoVetor



Retângulo

```
int CriaRetangulo(float base, float altura, Ponto cantoesq)
```

Parâmetros

base: tamanho da base do retângulo

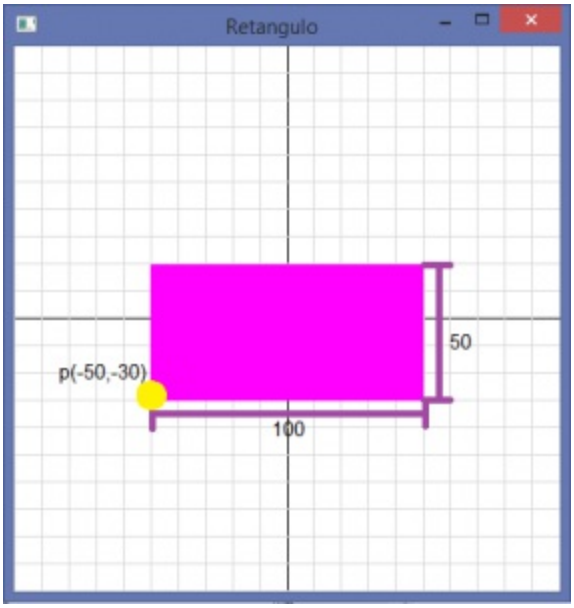
altura: altura do retângulo

cantoesq: ponto da base inferior esquerda do retângulo

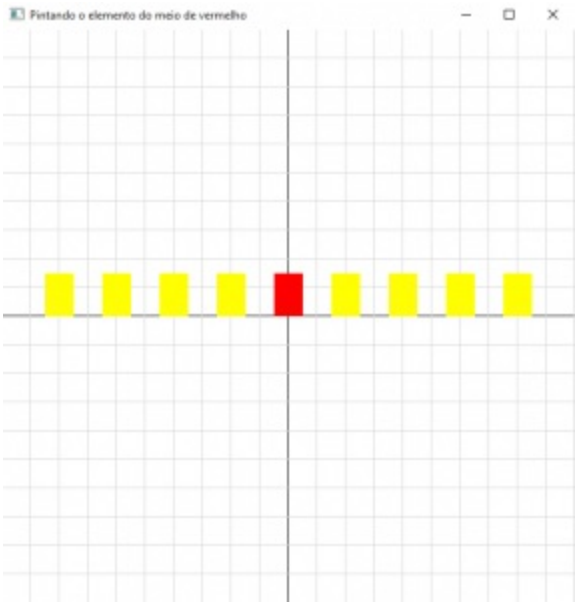
Retorno

Índice da geometria do tipo RETANGULO.

Exemplo 1: Entendendo parâmetros de CriaRetangulo



Exemplo 2: Usando retorno de CriaRetangulo



Quadrado

```
int CriaQuadrado(float lado, Ponto cantoesq)
```

Parâmetros

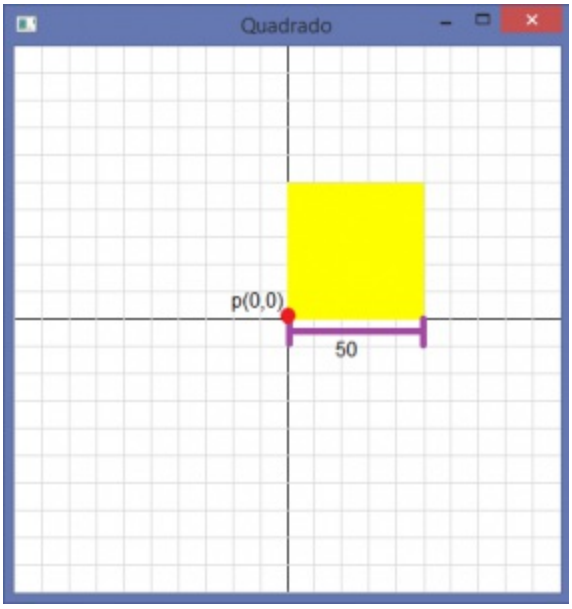
lado: tamanho do lado do quadrado

cantoesq: ponto da base inferior esquerda do quadrado

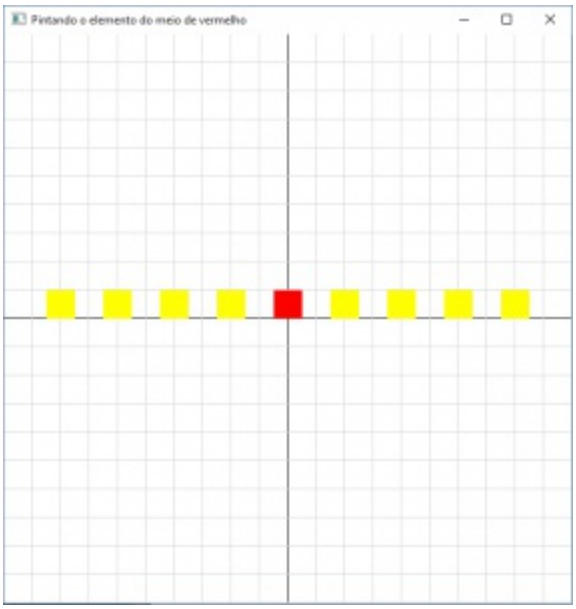
Retorno

Índice da geometria do tipo QUADRADO.

Exemplo 1: Entendendo parâmetros de CriaQuadrado



Exemplo 2: Usando retorno de CriaQuadrado



Triângulo

`int CriaTriangulo(float base, float altura, Ponto cantoesq)`

Função de criação de triângulos isósceles.

Parâmetros

base: tamanho da base do triângulo

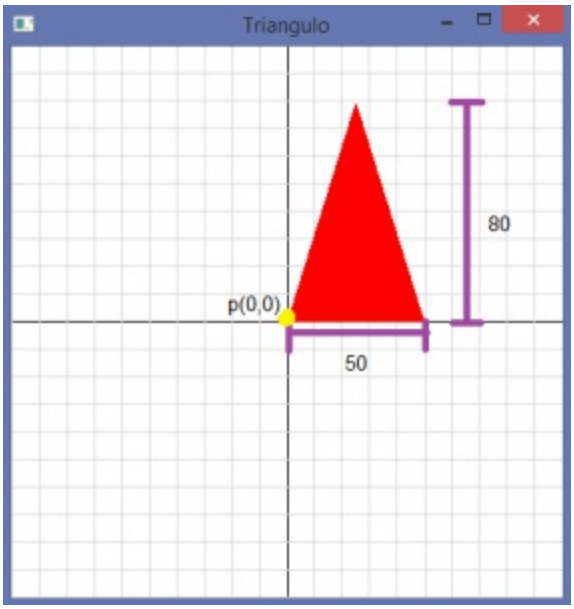
altura: altura do triângulo

cantoesq: ponto da base inferior esquerda do triângulo

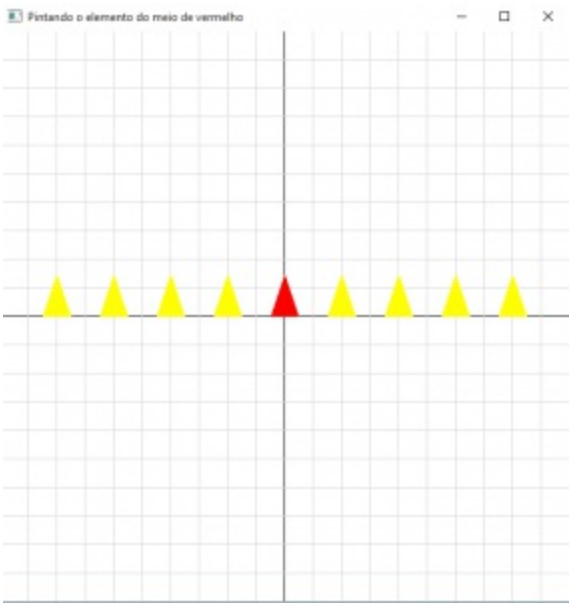
Retorno

Índice da geometria do tipo TRIANGULO.

Exemplo 1: Entendendo parâmetros de CriaTriangulo



Exemplo 1: Usando retorno de CriaTriangulo



Reta

```
int CriaReta(Ponto p1, Ponto p2)
```

Parâmetros

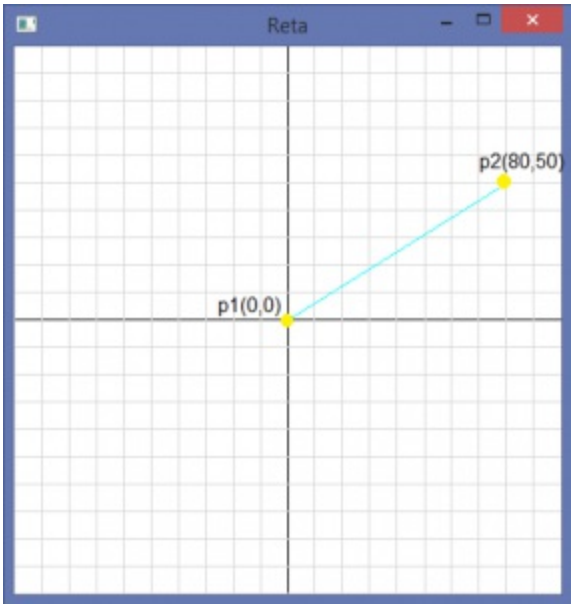
p1: coordenadas do primeiro ponto

p2: coordenadas do segundo ponto

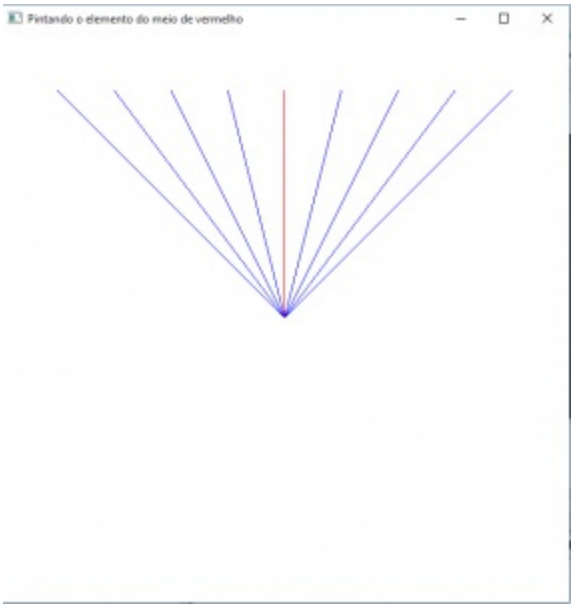
Retorno

Índice da geometria do tipo RETA.

Exemplo 1: Entendendo parâmetros de CriaReta



Exemplo 2: Usando retorno de CriaReta



Ponto

```
int CriaPonto(Ponto p)
```

Um ponto é representado por um pixel.

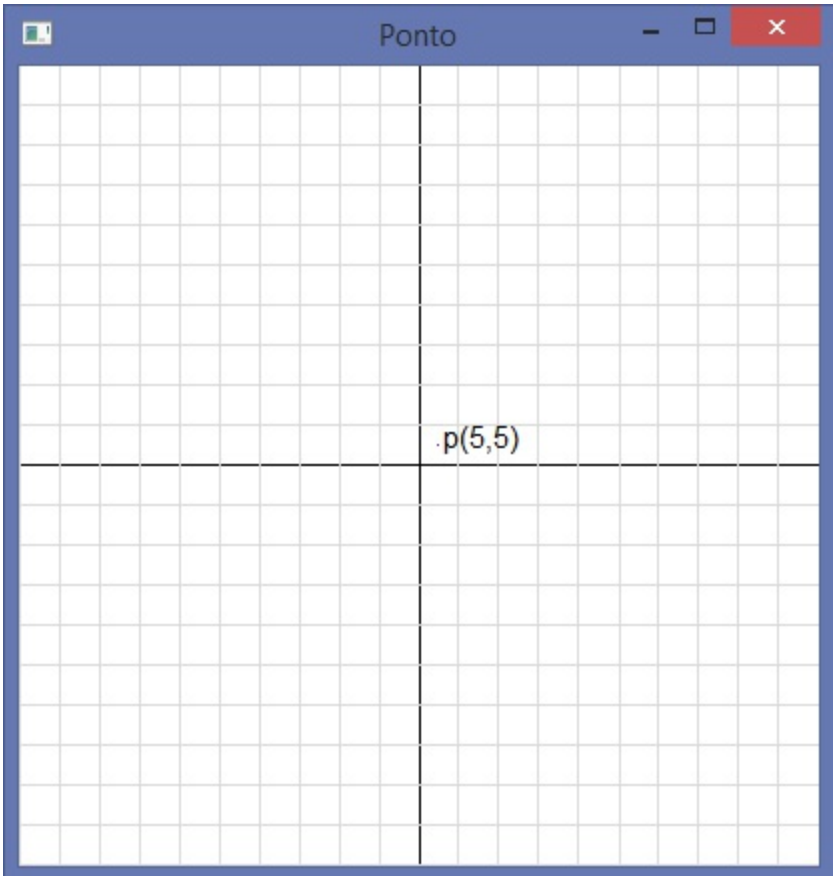
Parâmetros

p: coordenadas do ponto

RetornoImage not found

Índice da geometria do tipo PONTO.

Exemplo 1: Entendendo parâmetros de CriaPonto



Exemplo 2: Usando retorno de CriaPonto

TRANSFORMAÇÕES

Redimensionamento

Atualmente, há dois modos de usar a função *Redimensiona()*. Em ambos os modos, a relação de cada de cada ponto para a realização da transformação segue a seguinte regra:

Seja Image not found a coordenada do eixo x original do ponto, Image not found a coordenada do eixo y original do ponto, Image not found a coordenada resultado do eixo e Image not found a coordenada resultante do eixo y.

Image not found

Onde Image not found e Image not found são o fator de redimensionamento.

Os parâmetros da função *Redimensiona()* na playAPC são justamente o fator de redimensionamento em *escala decimal*.

Nota: caso os fatores de redimensionamento sejam negativos, ocorrerá uma reflexão.

Redimensionamento do último grupo definido

```
void Redimensiona(float x, float y)
```

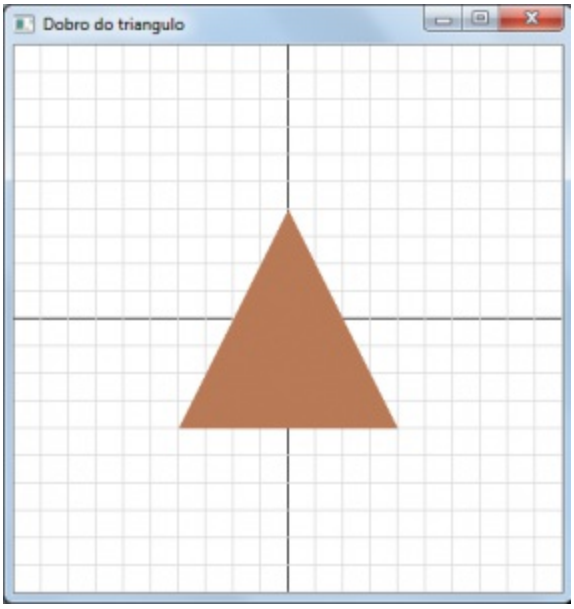
Rotaciona o último grupo dado uma *escala em x* e uma *escala em y*.

Parâmetros

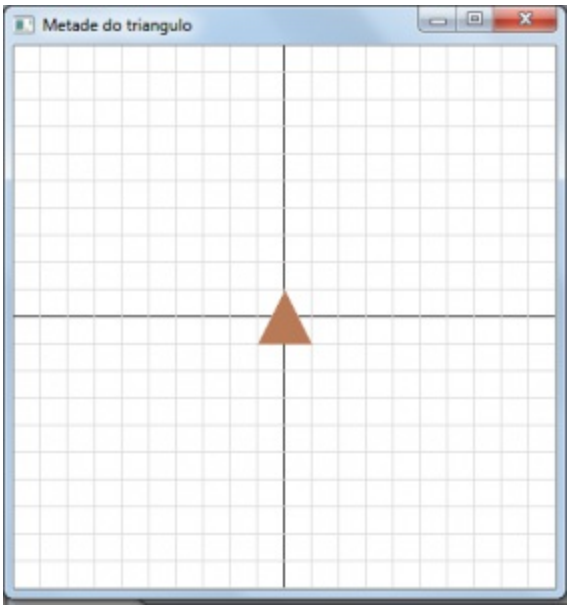
x: Escala de redimensionamento em x

y: Escala de redimensionamento em y

Exemplo 1



Exemplo 2



Redimensionamento de um grupo específico

```
void Redimensiona(float x, float y, int index)
```

Redimensiona todas as geometria grupo *index* dado as escalas de redimensionamento *x* e *y*.

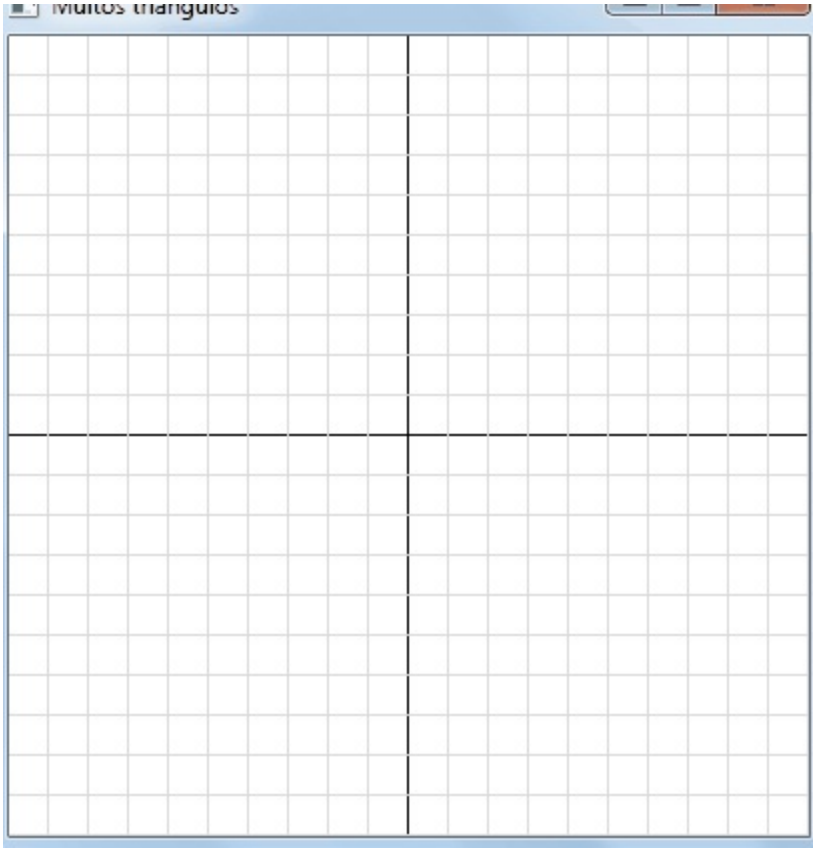
Parâmetros

x: escala de redimensionamento no eixo x

y: escala de redimensionamento no eixo y

index: índice do grupo

Exemplo



Rotação

Atualmente, há dois modos de usar a função *Gira()*. Em ambos os modos, a relação de cada de cada ponto para a realização da transformação segue a seguinte regra:

Seja Image not found a coordenada do eixo x original do ponto, Image not found a coordenada do eixo y original do ponto, Image not found a coordenada resultado do eixo e Image not found a coordenada resultante do eixo y.

Image not found

Onde Image not found é o ângulo de rotação.

Os parâmetros da função *Gira()* na playAPC é justamente o ângulo em **graus**.

Translação do último grupo definido

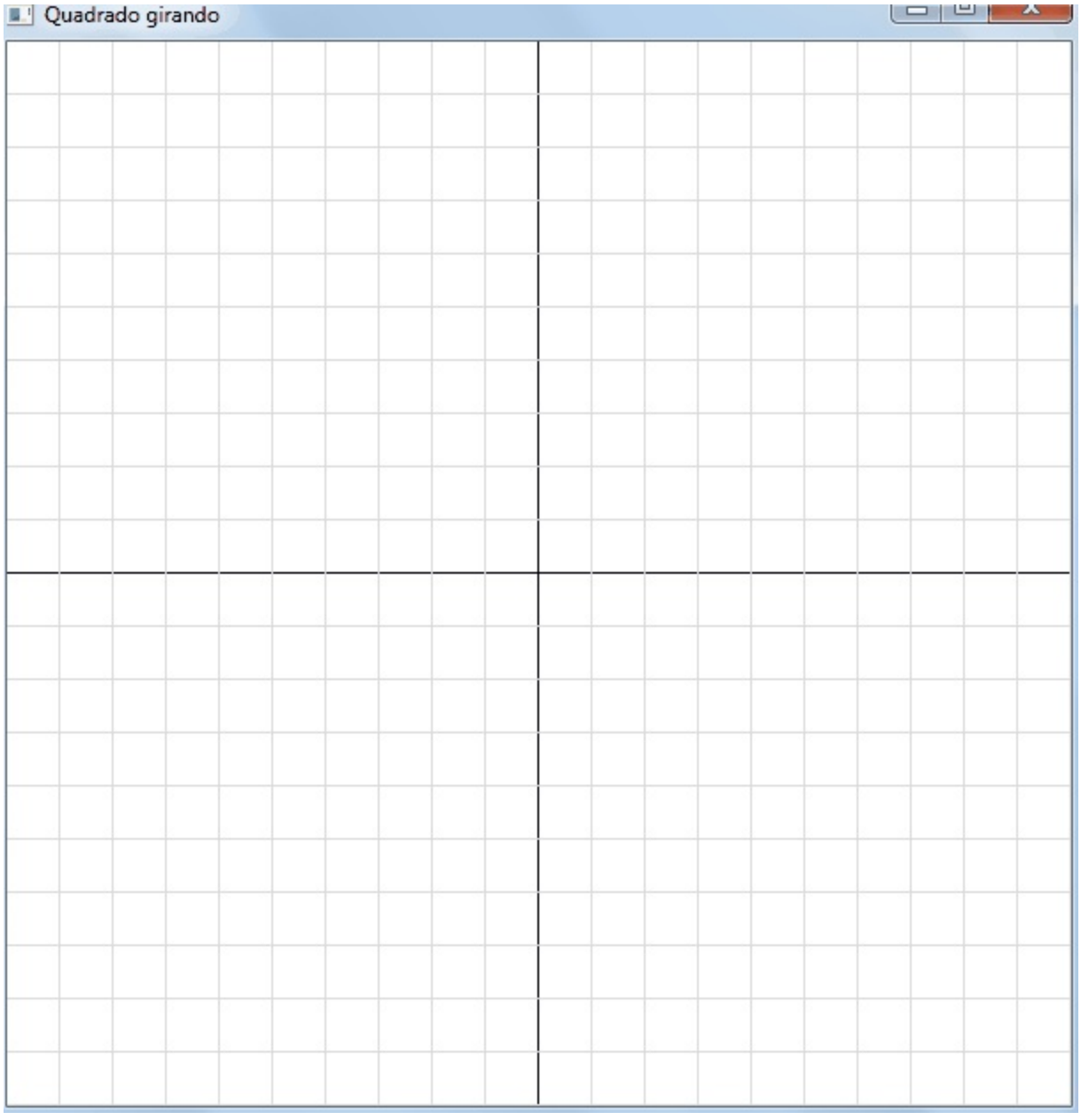
```
void Gira(float theta)
```

Rotaciona o último grupo dado um ângulo *theta*.

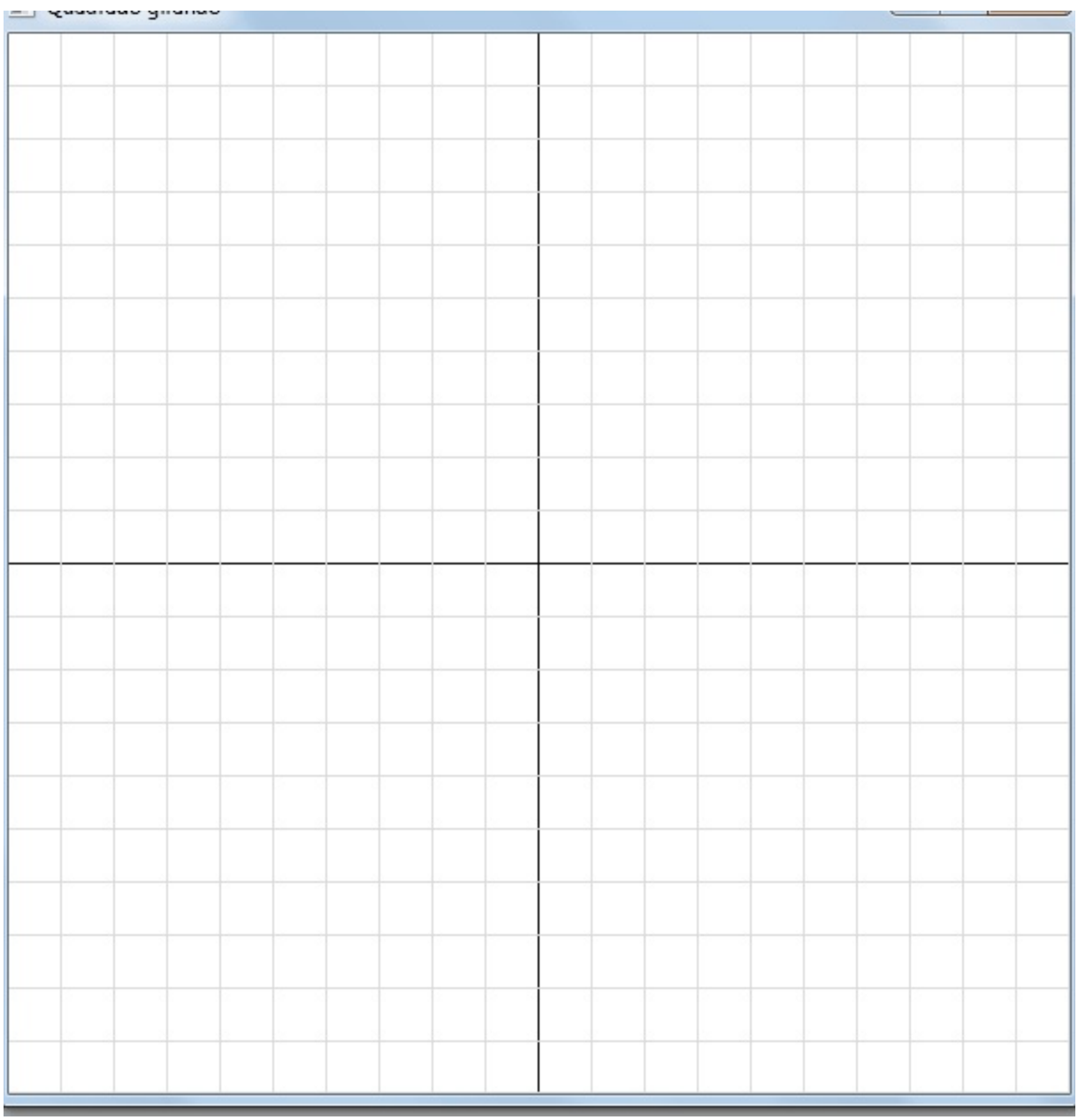
Parâmetros

theta: ângulo em graus de rotação

Exemplo 1



Exemplo 2



Translação de um grupo específico

```
void Gira(float theta, int index)
```

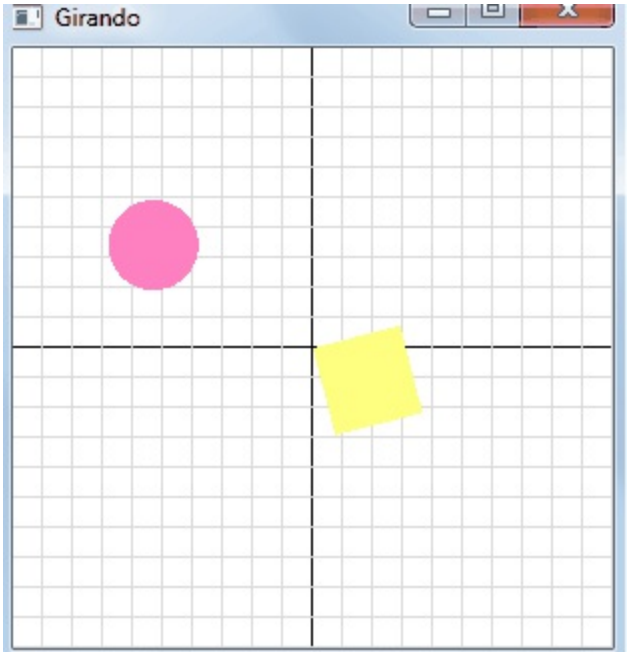
Rotaciona todas as geometria grupo *index* dado um ângulo *theta*.

Parâmetros

theta: ângulo em graus de rotação

index: índice do grupo

Exemplo



Translação

Atualmente, há dois modos de usar a função *Move()*. Em ambos os modos, a relação de cada de cada ponto para a realização da transformação segue a seguinte regra:

Seja Image not found a coordenada do eixo x original do ponto, Image not found a coordenada do eixo y original do ponto, Image not found a coordenada resultado do eixo x e Image not found a coordenada resultante do eixo y.

Image not found

Onde Image not found e Image not found são o incremento dada a posição original do ponto.

Os parâmetros da função *Move()* na playAPC são justamente as coordenadas resultantes.

Transladando o último grupo definido

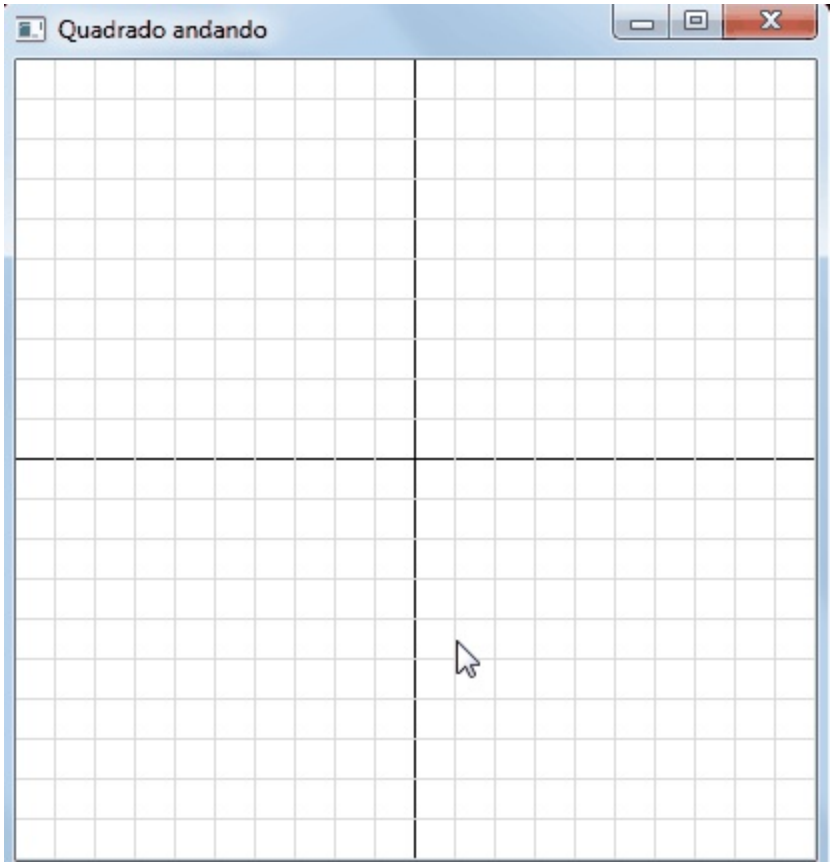
```
void Move(Ponto p)
```

Translada todas as geometrias do último grupo criado para uma posição *p* do plano cartesiano

Parâmetros

p: coordenadas do plano cartesiano

Exemplo



Transladando um grupo específico

```
void Move(Ponto p, int index)
```

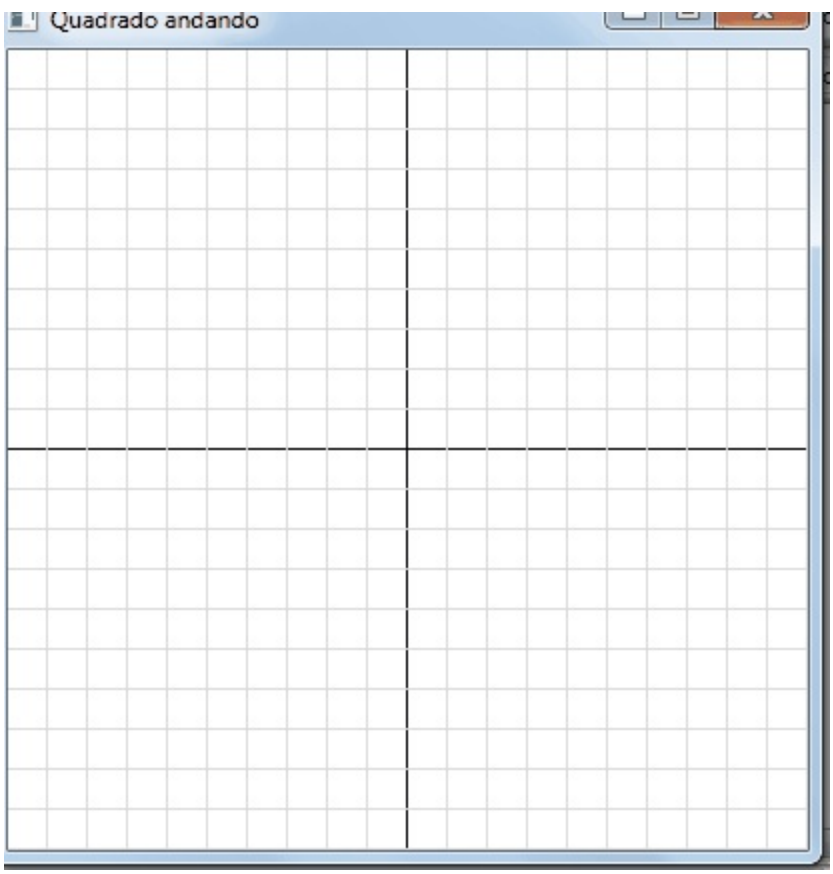
Translada todas as geometrias de um grupo *index* criado para uma posição *p* do plano cartesiano

Parâmetros

p: coordenadas do plano cartesiano

index: índice do grupo

Exemplo


Quadrado andando

EXTRAS

Alterar limites de renderização

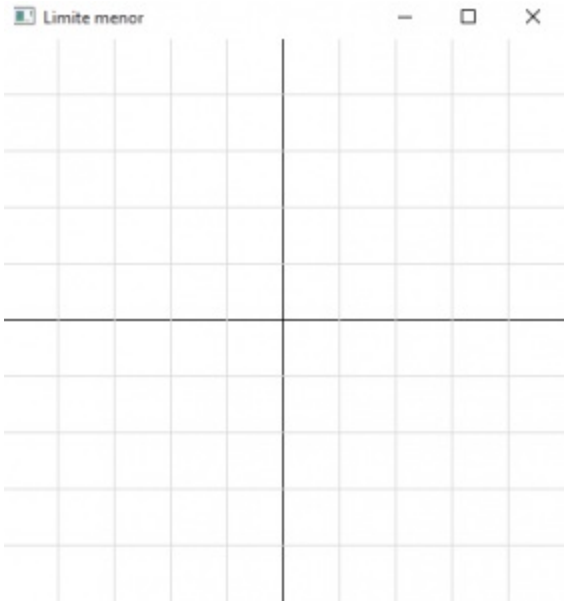
```
void MudaLimitesJanela(int limite)
```

Esta é a **única função** da playAPC que, se for utilizada, deve ser chamada **antes** da função *AbreJanela*. Ela altera os limites das coordenadas da playAPC. Caso esta função não seja chamada, os limites são de -100 à 100, tanto no eixo horizontal quanto no eixo vertical.

Parâmetros

limite: valor dos limites horizontais e verticais dos eixos de coordenadas da playCB.

Exemplo



Exibir Plano Cartesiano

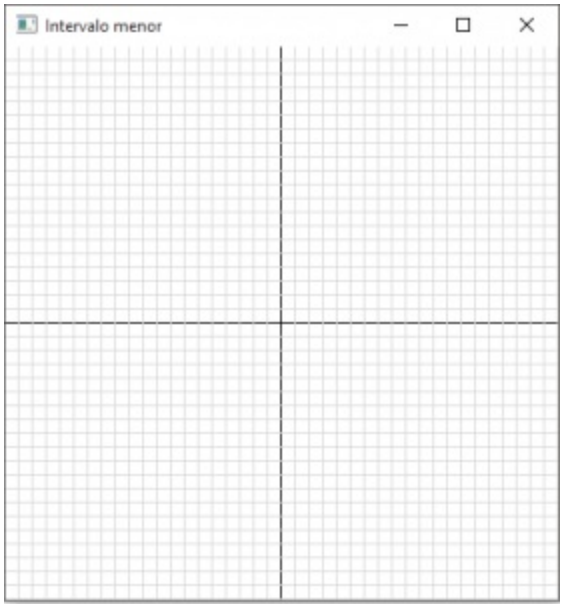
```
void MostraPlanoCartesiano(int intervalo)
```

Exibe linhas das coordenadas da playAPC. Seu centro é localizado em (0,0) e é identificado pelas linhas pretas. As linhas cinzas indentificam o intervalo das coordenadas.

Parâmetros

intervalo: valor do intervalo que cada linha cinza tem. Valor teve ser **maior que** 0.

Exemplo



Limpar desenho

```
void LimpaDesenho()
```

Destrói todos os elementos da cena, incluindo grupos e o plano cartesiano. Porém, não altera a cor do fundo nem os limites dos eixos.

Exemplo

Criar um grupo

Um grupo na playAPC é um conjunto de geometrias que, caso seja aplicada alguma das transformações disponíveis , todos os elementos daquele grupo também sofrerão da mesma transformação.

Todas as geometrias definidas **após** a chamada da função *CriaGrupo()* pertencerão ao índice desse grupo. Cada grupo é separada por chamadas de funções de *CriaGrupo()*. As transformações aplicadas no grupo terão como referência a **primeira** geometria.

nota: um grupo deve ter necessariamente no mínimo uma geometria.

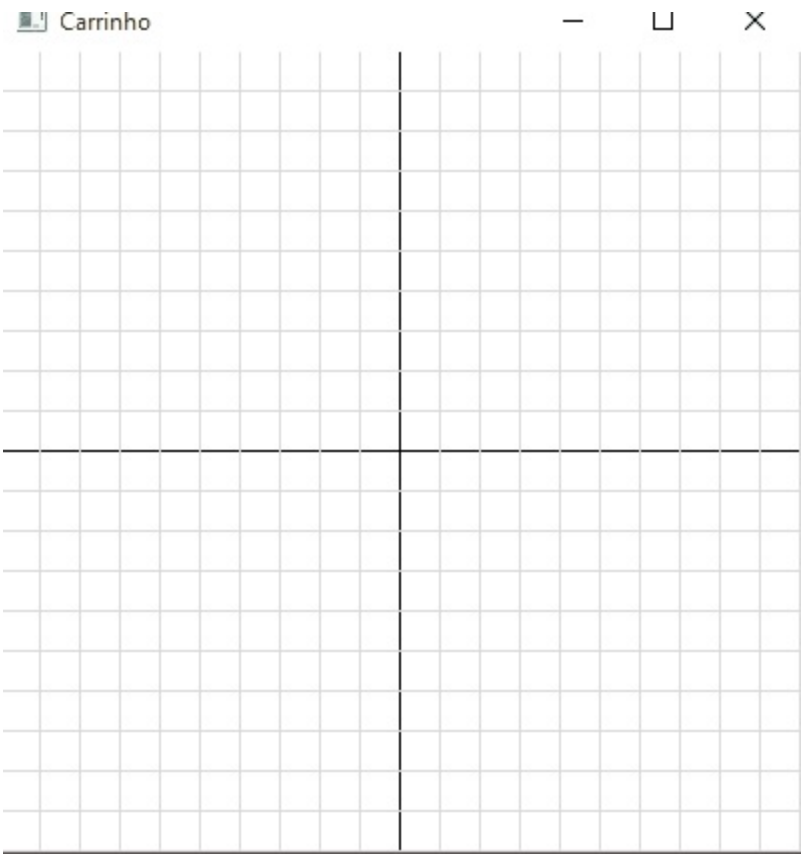
```
int CriaGrupo()
```

Função que agrupa geometrias.

Retorno

Índice do grupo que será determinado após a sua chamada.

Exemplo



Apagar um grupo

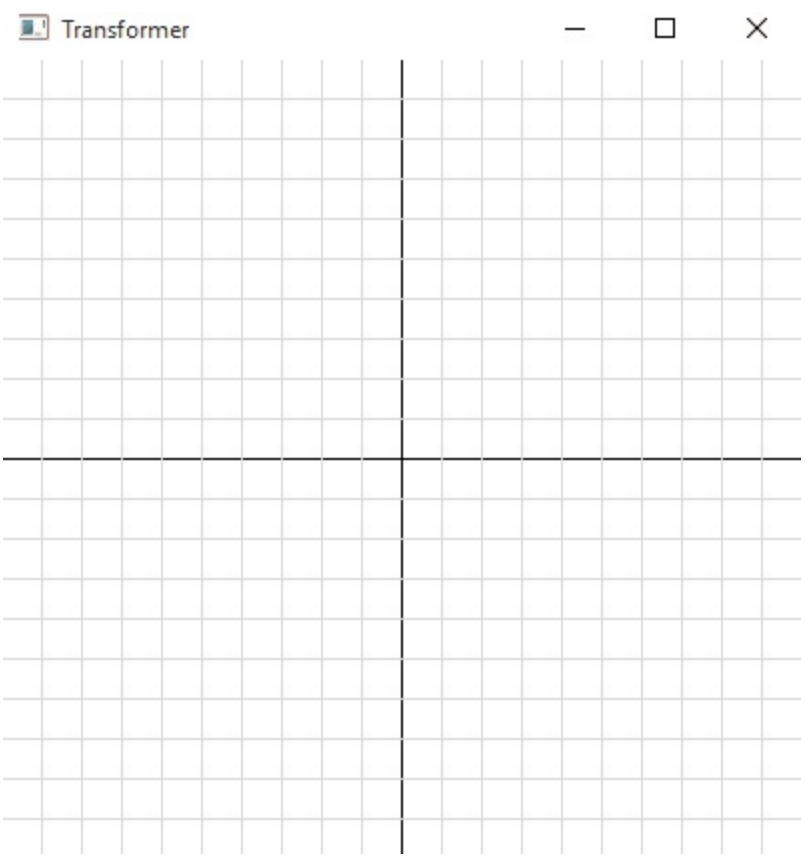
```
void ApagaGrupo(int index)
```

Função que apaga um grupo que foi anteriormente definido. O *index* deve ser o valor de retorno da função *CriaGrupo()*.

Parâmetros

index: índice do grupo

Exemplo



Capturar última tecla pressionada

```
int RetornaTecla()
```

A tecla que foi pressionada pelo usuário. Seus valores podem ser:

Valor	Descrição
GLFW_KEY_*n*	Teclas alfanuméricas ($n \in (0..9)$ ou $n \in (A..Z)$)
GLFW_KEY_SPACE	Espaço
GLFW_KEY_ESC	Escape
GLFW_KEY_F*n*	Function key ($n \in (0..25)$)
GLFW_KEY_LEFT	Seta para esquerda
GLFW_KEY_UP	Seta para cima
GLFW_KEY_DOWN	Seta para baixo
GLFW_KEY_RIGHT	Seta para direita
GLFW_KEY_LCONTROL	Control esquerdo
GLFW_KEY_RCONTROL	Control direito
GLFW_KEY_LALT	Alt esquerdo
GLFW_KEY_RALT	Alt direito
GLFW_KEY_TAB	Tabulador
GLFW_KEY_ENTER	Enter

GLFW_KEY_BACKSPACE	Backspace
GLFW_KEY_INSERT	Insert
GLFW_KEY_DEL	Delete
GLFW_KEY_PAGEUP	Page up
GLFW_KEY_PAGEDOWN	Page down
GLFW_KEY_HOME	Home
GLFW_KEY_END	End
GLFW_KEY_KP_*n*	Teclas numéricas do keypad ($n \in (0..9)$)
GLFW_KEY_KP_DIVIDE	Tecla dividir do keypad (÷)
GLFW_KEY_KP_MULTIPLY	Tecla multiplicar do keypad (×)
GLFW_KEY_KP_SUBTRACT	Tecla subtrair do keypad (–)
GLFW_KEY_KP_ADD	Tecla adição do keypad (+)
GLFW_KEY_KP_EQUAL	Tecla igual do keypad (=)
GLFW_KEY_KP_NUMLOCK	Tecla Numlock do keypad (=)
GLFW_KEY_CAPS_LOCK	Caps lock
GLFW_KEY_SCROLL_LOCK	Scroll lock
GLFW_KEY_PAUSE	Pause
GLFW_KEY_MENU	Menu

Exemplo

Verificar se tecla foi pressionada

```
int ApertouTecla(int tecla)
```

Parâmetros

tecla: tecla para ser verificada se foi pressionada uma vez ou não. Pode receber os seguintes valores:

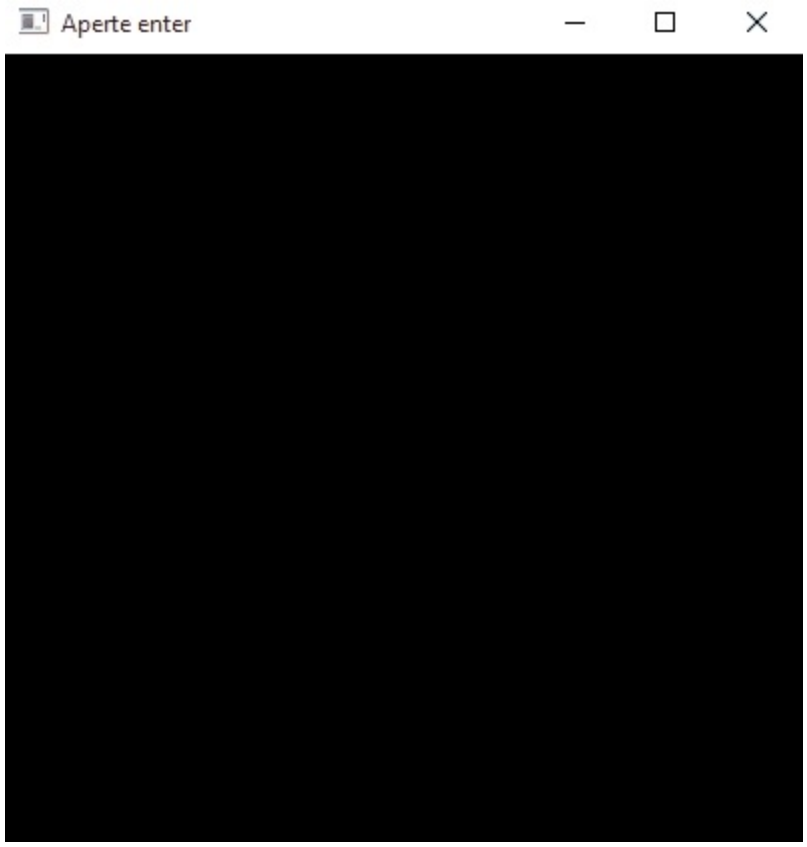
Valor	Descrição
GLFW_KEY_*n*	Teclas alfanuméricas ($n \in (0..9)$ ou $n \in (A..Z)$)
GLFW_KEY_SPACE	Espaço
GLFW_KEY_ESC	Escape
GLFW_KEY_F*n*	Function key ($n \in (0..25)$)
GLFW_KEY_LEFT	Seta para esquerda
GLFW_KEY_UP	Seta para cima
GLFW_KEY_DOWN	Seta para baixo
GLFW_KEY_RIGHT	Seta para direita
GLFW_KEY_LCONTROL	Control esquerdo
GLFW_KEY_RCONTROL	Control direito
GLFW_KEY_LALT	Alt esquerdo
GLFW_KEY_RALT	Alt direito
GLFW_KEY_TAB	Tabulador
GLFW_KEY_ENTER	Enter
GLFW_KEY_BACKSPACE	Backspace
GLFW_KEY_INSERT	Insert
GLFW_KEY_DEL	Delete
GLFW_KEY_PAGEUP	Page up
GLFW_KEY_PAGEDOWN	Page down
GLFW_KEY_HOME	Home
GLFW_KEY_END	End
GLFW_KEY_KP_*n*	Teclas numéricas do keypad ($n \in (0..9)$)
GLFW_KEY_KP_DIVIDE	Tecla dividir do keypad (÷)
GLFW_KEY_KP_MULTIPLY	Tecla multiplicar do keypad (×)
GLFW_KEY_KP_SUBTRACT	Tecla subtrair do keypad (–)
GLFW_KEY_KP_ADD	Tecla adição do keypad (+)
GLFW_KEY_KP_EQUAL	Tecla igual do keypad (=)
GLFW_KEY_KP_NUMLOCK	Tecla Numlock do keypad (=)

GLFW_KEY_CAPS_LOCK	Caps lock
GLFW_KEY_SCROLL_LOCK	Scroll lock
GLFW_KEY_PAUSE	Pause
GLFW_KEY_MENU	Menu

Retorno

0 se a tecla não foi pressionada naquela iteração ou *1* se ela foi pressionada.

Exemplo



Carregar uma imagem

Para poder carregar uma imagem na playAPC, primeiro é necessário abrir e prepará-la para ser carregada no programa. Os tipos de imagens suportadas são:

- png (podem possuir transparência)
- jpg
- bmp

Nota: dependendo do tamanho e qualidade da imagem, esta função pode vir a demorar para carregar a imagem, especialmente as que possuem transparência.

```
int AbreImagem(const char *src)
```

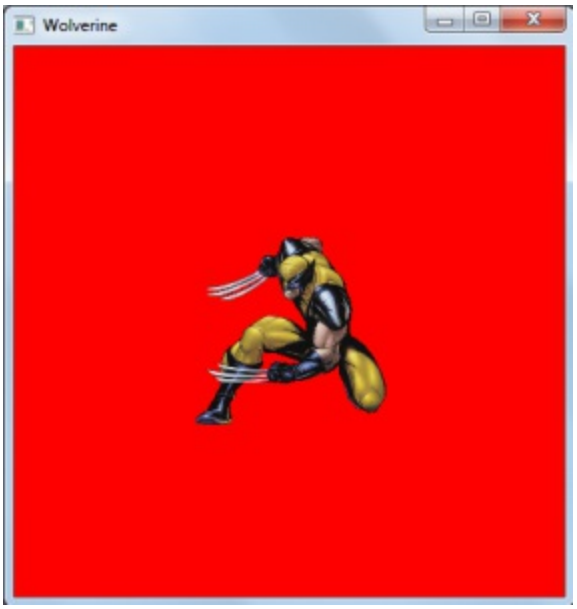
Parâmetros

src: nome da imagem

Retorno

Índice da imagem para ser associada a uma geometria

Exemplo



Associar uma imagem a uma geometria

Após abrir e carregar uma imagem na biblioteca usando *AbreImagem()*,é possível renderizar esta imagem em uma geometria qualquer, com **exceção** do Polígono, Reta e Ponto.

Preferivelmente, para que a imagem não sofra nenhum corte ou deformação, as geometrias ideais para renderizar uma imagem costumam ser **Quadrados** ou **Retângulos**.

Ao colocar a imagem na geometria, cada pixel de cor da imagem é multiplicado pela cor da geometria para resultar na nova cor daquele pixel. Ou seja:

Seja Image not found uma matriz definida por Image not found
Onde Image not found indica quantidade de vermelho que, Image not found indica a quantidade de verde e Image not found a quantidade de azul que o pixel contém. Os valores de Image not found, Image not found e Image not found variam de 0 à 1, onde 0 significa ausência e 1 significa presença. Para o cálculo da nova cor, é realizado o seguinte processo:

```
Deforma imagem de acordo com a geometria
```

```
Para cada pixel da geometria
        pixel novo = pixel da cor original Image not found pixel correspondente d
Fim-para
```

Portanto, para que a imagem não tenha nenhuma mudança inesperada de cor, é preferível que a cor da geometria seja **branco**.

A transparência da imagem é ativada se a imagem possue esse canal.

```
void AssociaImagem(int textura)
```

Associa uma imagem com a última geometria criada.

Parâmetros

textura: índice de retorno da função *AbreImagem()*.

Alterar espessura da borda de uma geometria

```
void Grafite(int espessura)
```

Seu valor máximo vai depender da capacidade gráfica da sua placa de vídeo. Caso o valor seja negativo ou 0, não há alteração.

A cor da borda será a mesma da geometria.

A ideia da espessura é contonar a geometria com uma linha.

Parâmetros

espessura: quantidade de vezes que a geometria deve ser contornada

Exemplo