Visão Geral do Processing

Programa Alô Mundo em Processing

Uma proposta de programa "Alô Mundo" em Processing pode ser o código abaixo:

```
line(17, 28, 75, 95)
```

Digite este exemplo e pressione o botão Executar, que é um ícone que se parece com o botão Reproduzir de qualquer dispositivo de áudio ou vídeo. Seu código aparecerá em uma nova janela, com fundo cinza e uma linha preta da coordenada (15, 25) a (70, 90). A coordenada (0, 0) é o canto superior esquerdo da janela de exibição. Com base neste programa para alterar o tamanho da janela de exibição e definir a cor de fundo, digite o código abaixo:

```
size(600, 600)
background(194, 68, 0)
stroke(255)
line(160, 30, 280, 360)
```

Para o código acima, foi definida uma janela de 600x600 pixels, como cor do plano de fundo fora definida a cor laranja-escuro, e desenha uma linha na cor branca, tendo um traço na mesma com o valor 255.

Alô Mouse

O código abaixo contém uma lista de instruções, tal estrutura é chamada de **esboço estático**. Um esboço estático contém funções, e estas relizam tarefas ou criam uma única imagem sem nenhuma animação ou interação. Caso seja necessário interativade em programas, estes podem ser desenhados como uma série de quadros, para tal adiciona-se as funções **setup()** e **draw()**, conforme o exemplo do código abaixo:

```
def setup():
      size(500, 500)
      stroke(255)
      background(190, 64, 0)
def draw():
      line(140, 100, mouseX, mouseY)
```

No código exemplificado a função **setup**() é executada uma vez, e a outra função **draw()** é executada como um laço, repetidamente. Assim, a função **setup()** realiza a inicialização do programa, ou como uma configuração inicial, uma preparação do ambiente para a execução repetida da função **draw()**.

```
def setup():
     size(400, 400)
     stroke(255)

def draw ():
     line(150, 25, mouseX , mouseY )

def mousePressed ():
     background(192, 64, 0)
```

Criando imagens do seu trabalho

Caso se deseje distribuir a imagem e não o projeto, o ideal é criar uma imagem de saída salva em arquivo.

Se você não quiser distribuir o projeto real, talvez queira criar imagens de sua saída. As imagens são salvas com a função **saveFrame()**.

saveFrame("arqsaida.png")

Para fazer o mesmo para uma sequência numerada, use # (marcas de hash) onde os números devem ser colocados:

saveFrame("saída-####.png")

Para uma saída de alta qualidade, você pode gravar geometria em arquivos PDF em vez de na tela, conforme descrito na seção posterior sobre a função size().

Mais sobre a Função size()

A função **size()** define as variáveis globais largura e altura. Para objetos cujo tamanho depende da tela, sempre use as variáveis largura e altura ao invés de um número. Isso evita problemas quando a linha size() é alterada.

```
size(400, 400)

# A maneira errada de especificar o meio da tela
ellipse(200, 200, 50, 50);

# Sempre no meio, não importa como a linha size() mude
ellipse(width/2, height/2, 50, 50);
```

Nos exemplos anteriores, a função `size()` especificava como argumentos apenas a largura e a altura para a janela a ser criada. Para além dos referidos argumentos é possível usar outro argumento para modificar a forma como o renderizador lida com a API de processamento para ocorrer uma saída específica (quer seja para saída em placa gráfica ou arquivo PDF). Vale destacar que o renderizador trabalha bem com gráficos vetoriais 2D com alta qualidade, porém demanda velocidade, particularmente ao lidar diretamente com pixels, a performance é lenta. Abaixo seguem descrições sobre outros modos de desenho que o Processing pode trabalhar:

```
size(400, 400, P2D)
```

O renderizador P2D usa OpenGL para renderização mais rápida de gráficos bidimensionais, enquanto usa as APIs gráficas mais simples do Processing e a fácil exportação de aplicativos do ambiente de desenvolvimento do Processing.

```
size(400, 400, P3D)
```

O renderizador P3D também usa OpenGL para renderização mais rápida. Ele pode desenhar objetos tridimensionais e objetos bidimensionais no espaço, bem como iluminação, textura e materiais.

```
size(600, 600, PDF, "output.pdf")
```

O renderizador de PDF desenha toda a geometria em um arquivo em vez da tela. Para usar PDF, além de alterar sua função `size()`, você deve selecionar e Importar a Biblioteca e, em seguida, PDF no menu Esboço. Este é um primo do renderizador padrão, mas grava diretamente em arquivos PDF.

Carregando e Exibindo Dados

Um dos aspectos exclusivos da API de processamento é a maneira como os arquivos são tratados. As funções *loadImage*() e *loadStrings*() esperam encontrar um arquivo dentro de uma pasta chamada *data*, que é um subdiretório da pasta sketch.

As funções de manipulação de arquivos incluem *loadStrings*() , que lê um arquivo de texto em uma matriz de objetos String, e *loadImage*() que lê uma imagem em um objeto *PImage*, o contêiner para dados de imagem em Processing.

```
# Exemplos de carregamento de um arquivo de texto e uma imagem JPEG
# da pasta de dados de um esboço.
lines = loadStrings("something.txt")
img = loadImage("picture.jpg")
```

A função *loadStrings*() retorna um array de objetos string Python, que é atribuído a uma variável chamada lines ; presumivelmente será usado mais tarde no programa com este nome. A razão pela qual loadStrings cria um array é que ele divide o arquivo something.txt em suas linhas individuais. A função a seguir retorna um objeto PImage , que é atribuído a uma variável chamada img .

Para adicionar um arquivo à pasta de dados de um sketch do Processing, use a opção de menu Sketch → Add File, ou arraste o arquivo para a janela do editor do PDE. A pasta de dados será criada se ainda não existir.

Para visualizar o conteúdo da pasta de esboços, use a opção de menu Sketch → Show Sketch Folder (Ver pasta de Sketchs/Esboços). Isso abre a janela de esboço no navegador de arquivos do seu sistema operacional.

Bibliotecas adicionam novos recursos
Uma biblioteca é uma coleção de código em um formato especificado que facilita o uso no Processing. As bibliotecas foram importantes para o crescimento do projeto, porque permitem que os desenvolvedores tornem novos recursos acessíveis aos usuários sem precisar torná-los parte da API de processamento principal.

Várias bibliotecas principais vêm com Processing. Elas podem ser vistas na seção Bibliotecas da referência online (também disponível no menu Ajuda no PDE). Essas bibliotecas podem ser vistas em http://processing.org/reference/libraries/

Um exemplo é a biblioteca de exportação de PDF. Esta biblioteca possibilita escrever arquivos PDF diretamente do Processing. Esses arquivos gráficos vetoriais podem ser dimensionados para qualquer tamanho e produzir em resoluções muito altas.

Para usar a biblioteca PDF em um projeto do Modo Python, escolha Sketch → Import Library → pdf. Isso adicionará a seguinte linha ao topo do esboço:

```
add_library('pdf')
```

Esta linha instrui o Processing a usar a biblioteca indicada ao executar o sketch.

Agora que a biblioteca PDF foi importada, você pode usá-la para criar um arquivo. Por exemplo, a linha de código a seguir cria um novo arquivo PDF chamado lines.pdf no qual você pode desenhar.

```
beginRecord(PDF, "linhas.pdf")
```

Cada função de desenho, como line() e ellipse() , agora desenhará na tela e também no PDF.

Outras bibliotecas fornecem recursos como leitura de imagens de uma câmera, envio e recebimento de comandos MIDI e OSC, controle sofisticado de câmera 3D e acesso a bancos de dados MySQL.

Esboço e script

Os esboços de processamento são compostos de uma ou mais guias, com cada guia representando um pedaço de código. O ambiente é projetado em torno de projetos que são algumas páginas de código e geralmente de três a cinco guias no total. Isso abrange um número significativo de projetos desenvolvidos para testar e prototipar ideias, geralmente antes de incorporá-las a um projeto maior ou criar um aplicativo mais robusto para uma implantação mais ampla.

Processing reúne nossa experiência na construção de software desse tipo (esboços de trabalhos interativos ou visualização orientada por dados) e simplifica as partes que achamos que deveriam ser mais fáceis, como começar rapidamente e isolar novos usuários de problemas como os associados à configuração estruturas de programação complicadas.

Não comece tentando construir uma catedral

Se você já estiver familiarizado com programação, é importante entender como o Processing difere de outros ambientes e linguagens de desenvolvimento. O projeto Processing incentiva um estilo de trabalho que constrói código rapidamente, entendendo que ou o código será usado como um esboço rápido ou as ideias estão sendo testadas antes de desenvolver um projeto final. Isso pode ser mal interpretado como heresia da engenharia de software. Talvez não estejamos longe de “hackear”, mas isso é mais apropriado para as funções em que o Processing é usado. Por que forçar estudantes ou programadores casuais a aprender sobre contextos gráficos, threading, e funções de manipulação de eventos antes que eles possam mostrar algo na tela que interage com o mouse? O mesmo vale para desenvolvedores avançados: por que eles sempre precisam começar com as mesmas duas páginas de código sempre que iniciam um projeto?

Em outro cenário, a capacidade de testar as coisas rapidamente é uma prioridade muito maior do que a estrutura de código sofisticada. Normalmente você não sabe qual será o resultado, então você pode construir algo uma semana para tentar uma hipótese inicial e construir algo novo na próxima com base no que foi aprendido na primeira semana. Para isso, lembre-se das seguintes considerações ao começar a escrever código com Processing:

- Tenha cuidado ao criar estruturas desnecessárias em seu código. À medida que você aprende a encapsular seu código em classes, é tentador criar classes cada vez menores, porque os dados sempre podem ser destilados ainda mais. Você precisa de aulas no nível de moléculas, átomos ou quarks? Só porque os átomos ficam menores não significa que precisamos trabalhar em um nível mais baixo de abstração. Se uma classe tem meia página, faz sentido ter seis subclasses adicionais, cada uma com meia página? A mesma coisa poderia ser realizada com uma única classe que é uma página e meia no total?
- Considere a escala do projeto. Nem sempre é necessário criar software de nível empresarial no primeiro dia. Explore primeiro: descubra o código mínimo necessário para ajudar a responder suas perguntas e satisfazer sua curiosidade.

O argumento não é evitar reescrever continuamente, mas sim atrasar o trabalho de engenharia até que seja apropriado. O limite para onde começar a engenharia de um software é muito mais tarde do que para projetos de programação tradicionais, porque há um tipo de arte no processo inicial de iteração rápida.

Claro, uma vez que as coisas estão funcionando, evite o desejo de reescrever por si só. Uma reescrita deve ser usada ao abordar um problema completamente diferente. Se você conseguiu acertar o prego na cabeça, você deve refatorar para limpar nomes de funções e interações de classe. Mas uma reescrita completa do código já finalizado é quase sempre uma má ideia, não importa o quão "feio" possa parecer.