Projeto Interativo III

Marcelo Hashimoto

Última Atualização: 21 de fevereiro de 2014

1 Introdução

Neste Projeto Interativo, cada grupo deve desenvolver um **jogo na plataforma Allegro, baseado em visão computacional**. Além do *código* desse jogo, o grupo deve entregar um *artigo* e um *pôster* nos quais detalha seus aspectos técnicos. Por fim, uma *apresentação* deve ser elaborada para a Semana de Projetos.

2 Especificação

O jogo deve ser desenvolvido na linguagem **C**, versão **99**. A *flag* `-std=c99` garante que o compilador `gcc` considere estritamente essa versão. Além da biblioteca padrão, somente aquelas mencionadas nas Subseções 2.2 e 2.3 podem ser utilizadas.

2.1 Ambiente de Execução

O código deve ser compilado e executado sem nenhum erro no **Terminal** do sistema operacional **Mac OSX** instalado nas máquinas do laboratório **I443**.

Cabe observar que esse ambiente *não é obrigatório durante o desenvolvimento*, apenas durante a avaliação.

2.2 Plataforma Allegro

A interface gráfica deve ser desenvolvida na plataforma **Allegro** [1], versão **5.x**. O *Manual de Referência* [3] e os *Tutoriais de API* [2] dessa plataforma contêm todas as informações necessárias.

Adicionalmente, o arquivo `exemplo.c` contém um exemplo simples de programa em Allegro integrado com a plataforma de visão computacional descrita na Subseção 2.3. Recomenda-se compilar, executar e compreender totalmente cada linha de código desse exemplo antes de iniciar qualquer desenvolvimento próprio.

2.3 Visão Computacional

A interação do usuário deve ser feita **principalmente através de câmeras conectadas ao computador**. O arquivo `camera.c` contém a implementação de uma biblioteca que permite o acesso às imagens dessas câmeras através da plataforma *OpenCV* [5]. Para utilizar essa biblioteca, o arquivo de cabeçalho `camera.h` deve ser incluído no código. Essa inclusão adiciona o tipo de dados `camera`.

```
typedef struct {
  unsigned char ***quadro;
  int largura, altura;
  CvCapture *capture;
} camera;
```

Os campos `largura` e `altura` representam a resolução em *pixels* da imagem de uma câmera. O campo `quadro` representa essa imagem como uma matriz tridimensional de caracteres sem sinal. São esses três campos que devem ser analisados pelo programa. O campo `capture` é auxiliar e deve ser ignorado.

A primeira dimensão da matriz representa a altura, a segunda dimensão representa a largura e a terceira representa os canais *vermelho*, *verde* e *azul* de cada pixel. Para ilustrar essa definição, considere um apontador `camera *cam`. Sabemos que os valores

```
cam->quadro[5][9][0]
cam->quadro[5][9][1]
cam->quadro[5][9][2]
```

representam respectivamente a quantidade de vermelho, verde e azul do pixel na posição vertical 5 e posição horizontal 9. Cabe lembrar que cada quantidade, sendo um caractere sem sinal, está entre 0 e 255.

Para utilizar esse tipo de dados apropriadamente, o cabeçalho também adiciona as seguintes funções básicas.

- `camera *camera_inicializa(int i);`

  Aloca, inicializa e devolve o endereço de uma variável do tipo `camera`. O valor dessa variável representa a i-ésima câmera conectada ao computador. A contagem inicia do zero: $i = 0$ corresponde à primeira câmera, $i = 1$ corresponde à segunda câmera e assim em diante.

- `void camera_finaliza(camera *cam);`

  Finaliza e libera os recursos utilizados pela câmera representada por `cam`.

- `void camera_atualiza(camera *cam);`

  Atualiza a câmera representada por `cam`, escrevendo uma nova imagem no campo `quadro`. A chamada dessa função e a subsequente lógica de visão computacional devem ser executadas imediatamente antes das funções de desenho do Allegro.

- `void camera_copia(camera *cam, unsigned char ***matriz, ALLEGRO_BITMAP *bitmap);`

  Copia a imagem representada por `matriz` para um *bitmap* do Allegro. Espera-se que as resoluções de ambos sejam iguais à de uma imagem da câmera representada por `cam` e que a matriz tenha o mesmo formato que o campo `quadro`, mas a matriz não precisa necessariamente *ser* esse campo.

Por fim, também adiciona as seguintes funções de conveniência.

- `unsigned char ***camera_aloca_matriz(camera *cam);`

  Aloca e devolve o endereço de uma matriz com o mesmo formato do campo `quadro` de `cam`.

- `void camera_libera_matriz(camera *cam, unsigned char ***matriz);`

  Libera uma matriz com o mesmo formato do campo `quadro` de `cam`.

Cabe mencionar que **a OpenCV não pode ser utilizada diretamente**, apenas através dessa biblioteca. Além disso, interação através de mouse e teclado é permitida apenas para *calibrar algoritmos* e *fechar janelas*.

2.4 Estrutura do Artigo

O artigo deve ser escrito em LaTeX e seu objetivo é defender a qualidade dos algoritmos implementados, além de disponibilizar informação suficiente para reproduzi-los de maneira precisa. Espera-se que contenha os seguintes tópicos.

1. **Resumo.** Deve descrever de maneira sucinta o jogo, as principais características dos algoritmos e os resultados. O propósito do resumo é apresentar uma visão geral que omite detalhes mas estabelece a adequação do trabalho.

2. **Introdução.** Deve contextualizar o trabalho, definindo o jogo e identificando os algoritmos necessários para sua interface.

3. **Revisão da literatura.** Deve relacionar o trabalho com outros, citando aqueles que foram utilizados como base e destacando as características originais.

4. **Desenvolvimento.** Deve descrever os aspectos técnicos do trabalho.

5. **Resultados.** Deve discutir resultados obtidos pelo trabalho e detalhar experimentos que os originaram.

6. **Considerações finais.** Deve estabelecer conclusões a partir dos resultados e formular conjecturas que poderiam ser confirmadas por evoluções do trabalho. O propósito das considerações finais é apresentar uma visão honesta que enfatiza as conclusões positivas mas não ignora as negativas.

7. **Bibliografia.** Deve enumerar as fontes de informação consultadas durante o desenvolvimento do trabalho. Espera-se que essas fontes sejam confiáveis e persistentes. Prefere-se, por exemplo, artigos acadêmicos de periódicos estabelecidos em vez de páginas da Internet.

Cabe observar que esses tópicos não precisam necessariamente corresponder a seções. De fato, espera-se que a descrição dos aspectos técnicos seja suficientemente complexa para ser particionada em múltiplas seções.

2.5 Pôster e Apresentação

O objetivo do pôster e da apresentação é demonstrar o programa e apresentar de maneira sucinta os principais pontos do artigo. O pôster também deve ser escrito em LaTeX.

3 Entrega

Cada grupo deve criar um repositório no *GitHub* [4] chamado `BCC-1s14-PI3-NOMEDOTRABALHO` e incluir os usuários `mhsenac`, `ehsenac` e `dmsenac` como colaboradores desse repositório. Além do código, que pode ser organizado como o grupo desejar, a raiz deve conter uma pasta chamada `banca` com os seguintes arquivos.

- **autores.pdf**

  Lista os nomes completos dos membros do grupo.

- **instrucoes.pdf**

  Descreve o procedimento para configurar, compilar e executar o programa.

- **artigo.pdf**

  Contém o artigo associado ao trabalho.

- **poster.pdf**

  Contém o pôster associado ao trabalho.

O repositório deve ser preenchido ao longo do semestre, de acordo com as seguintes datas.

- **20/02: Definição do Grupo (1 a 4 membros)**
- **29/05: Artigo Preliminar**
- **05/06: Código, Pôster e Apresentação**
- **12/06: Artigo Final**

Em cada uma dessas datas, a avaliação considerará como entregue *a última versão antes das 23:50*.

4 Orientação

Em toda aula, qualquer grupo pode fazer uma solicitação de *prévia*, na qual o professor responsável pela disciplina avalia uma versão incompleta da apresentação e identifica perguntas que podem ser feitas.

Adicionalmente, qualquer grupo pode fazer uma solicitação de *revisão*, na qual o professor avalia uma versão incompleta do artigo e sugere modificações. Essa, porém, deve ser feita com uma semana de antecedência.

5 Avaliação

5.1 Geral

- **Usabilidade.** A interface deve ser *fluida*, *intuitiva* e *divertida*. A visão computacional não pode aparentar ser simplesmente uma ferramenta que o desenvolvedor foi obrigado a utilizar.
- **Adequação.** A qualidade do projeto, particularmente do que foi feito *além* dos requisitos mínimos, deve ser consistente com o que se espera de alunos do 3^o período do Bacharelado em Ciência da Computação.

5.2 Código

Além da qualidade dos algoritmos, serão considerados os seguintes critérios.

- **Documentação.** Inclua comentários relevantes, que ajudem o leitor a compreender.
- **Clareza.** Escreva código auto-explicativo, que evite naturalmente um excesso de comentários.
- **Organização.** Particione funções e arquivos, seguindo uma metodologia lógica.

Cabe observar que **o código deve ser original**. Trechos de outros autores *podem ser estudados e adaptados se necessário* mas *não podem ser simplesmente copiados e colados*. Caso não seja capaz de explicar todos os detalhes de implementação, o grupo será reprovado *independentemente das notas de artigo e apresentação*.

5.3 Artigo

Além da adequação do texto, serão considerados os seguintes critérios.

- **Norma culta.**
- **Padrão científico.**

Cabe observar que **o artigo deve ser original**. Trechos de outros autores podem ser incluídos apenas quando *explicitamente apresentados como citação* e *adequadamente acompanhados de referência*. Caso seja constatado algum tipo de plágio, o grupo será reprovado *independentemente das notas de código e apresentação*.

5.4 Apresentação

A banca lerá a versão preliminar do artigo antes da apresentação e observará os seguintes critérios.

1. **Desenvoltura.**
2. **Conhecimento.**

Cabe observar que, embora as tarefas necessárias possam ser divididas entre os membros de um grupo, **todo conhecimento adquirido individualmente deve ser compartilhado**, pois a banca poderá formular perguntas sobre *qualquer parte* do projeto para *qualquer membro* do grupo. Uma resposta individual insatisfatória será considerada na avaliação *mesmo que o restante do grupo seja capaz de produzir respostas satisfatórias*.

Referências

[1] *Allegro*. http://alleg.sourceforge.net/.

[2] *Allegro 5 API Tutorials*. http://wiki.allegro.cc/index.php?title=Allegro_5_API_Tutorials.

[3] *Allegro 5 Reference Manual*. http://alleg.sourceforge.net/a5docs/5.0.10/.

[4] *GitHub*. https://www.github.com/.

[5] *OpenCV*. http://www.opencv.org/.

[6] R. C. Gonzalez e R. E. Woods. *Processamento Digital de Imagens (3a. Edição)*. Pearson, 2007.