Universidade Federal de Mato Grosso do Sul
Faculdade de Computação

Computação Gráfica 2021

Trabalho – Parte 0

Paulo A. Pagliosa

A **Parte 0** do trabalho é um exercício *opcional* e tem por propósito apenas incentivá-lo a conhecer e experimentar o arcabouço a partir do qual você desenvolverá as demais partes do trabalho. Se você optar pela entrega da **Parte 0**, você receberá uma nota de zero a dez de acordo com a lista de objetivos no final desta descrição, mas que não será considerada no cômputo de sua média de aproveitamento final.

1 Passo 1: Código-fonte

O primeiro passo é obter o código-fonte, disponível no diretório `cg` do repositório `https://git.facom.ufms.br/pagliosa/cg.git` (você deve ter uma conta na Facom para poder acessar o repositório), e copiá-lo em algum diretório local em seu computador de trabalho. Feito isso, seu diretório de trabalho conterá um diretório chamado `cg`, contendo, por sua vez, um diretório chamado `common` e um diretório chamado `p0`. Não se preocupe por enquanto com o diretório `common`. Esse contém código de terceiros, a ser explicado em aula, e o código-fonte de uma biblioteca comum, como sugere o nome do diretório, a todas as partes do trabalho. Essa biblioteca permite, entre outras funcionalidades, que você crie janelas gráficas com outra biblioteca chamada GLFW[1], desenhe nelas usando OpenGL, trate os eventos de entrada do usuário (a partir de dispositivos como teclado e mouse) e crie interfaces gráficas com o usuário (GUIs) com uma biblioteca chamada Dear ImGui[2].

O código que interessa para esta parte do trabalho está no diretório `p0`. Este contém dois outros diretórios, chamados `assets` e `build`, e os arquivos `P0.h`, `P0.cpp` e `Main.cpp`, todos explicados em aula. Em `assets` há os arquivos `p0.vs` e `p0.fs`, mas ignore-os por enquanto. No diretório `build` há um diretório `vs2019` contendo um projeto do Visual Studio 2019. Assim, a estrutura do diretório `p0` em seu diretório de trabalho será

```
assets
  p0.fs
  p0.vs
build
  vs2019
    p0.sln
    p0.vcxproj
    p0.vcxproj.filters
Main.cpp
P0.cpp
P0.h
```

[1] `http://www.glfw.org`.
[2] `https://github.com/ocornut/imgui`.

2 Passo 2: Tarefas

A seguir, abra, no Visual Studio 2019, o arquivo `p0.sln` e construa o arquivo executável na configuração de *release* (este aparecerá no diretório `p0` com o nome `p0.exe`). Então, execute o programa. Você verá em sua tela uma janela gráfica com um triângulo no centro e uma interface com o usuário como na Figura 1.

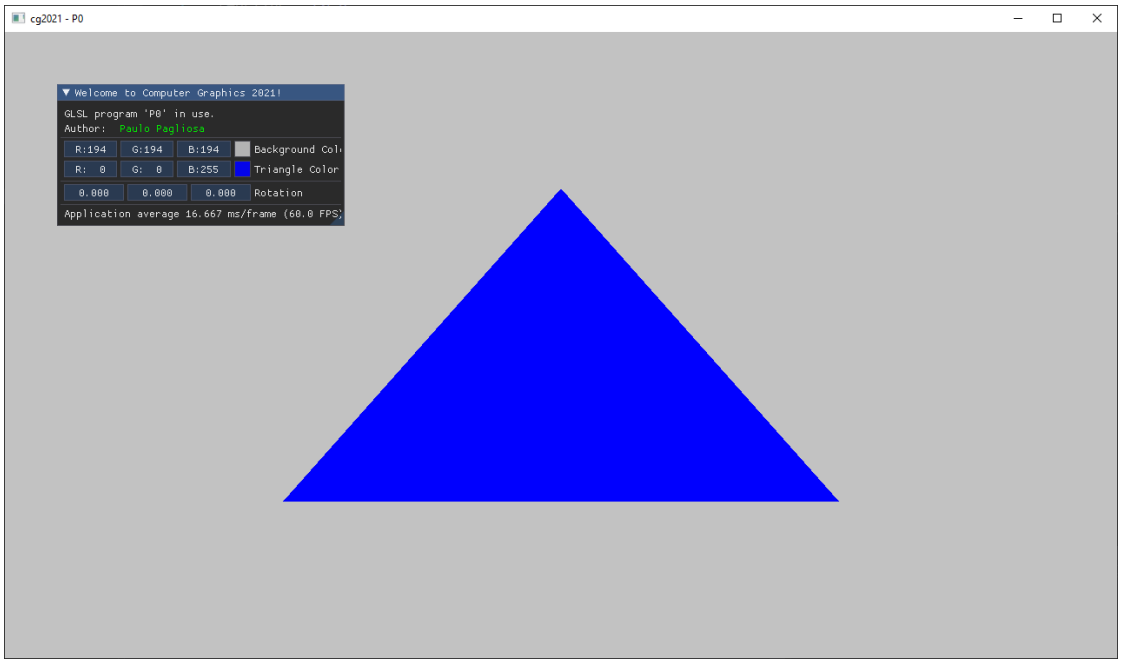

Figura 1: Programa `p0.exe` (no Windows).

Experimente os controles da GUI. Você pode modificar a cor de fundo da janela (`Background Color`) e a cor do triângulo (`Triangle Color`). Você pode também arrastar o mouse nas caixas de entrada de `Rotation` para rotacionar o triângulo em torno dos eixos x, y e z de um sistema de coordenadas Cartesianas chamado *sistema global de coordenadas* (veremos mais sobre isso no Capítulo 2).

As duas atividades de programação dessa parte do trabalho serão modificações no arquivo `P0.cpp`, mais especificamente no método `P0::gui()`, e consistem em:

A1 Alterar a interface gráfica para exibir seu nome (e o nome de seu colega de grupo, se for caso), como autor(es) do programa.

A2 Adicionar à interface com o usuário controles para especificar, a exemplo da rotação, os componentes x, y, e z da posição (`Position`) e da escala (`Scale`) a ser aplicada ao triângulo, os quais são argumentos da mensagem da linha 31 do arquivo.

3 Passo 3: README

Todas as partes do trabalho deverão ter, no diretório correspondente, um arquivo texto chamado `README`. Este deve conter o(s) nome(s) do(s) autor(es) e uma descrição de como gerar (caso o Visual Studio 2019 não seja utilizado) e executar o programa. Por exemplo, quais são os argumentos da linha de comando, se o programa deve ser executado de um diretório específico, etc. Em adição, devem estar descritas quais as atividades você

conseguiu ou não implementar, parcial ou totalmente (ou se implementou quaisquer outras extensões por iniciativa própria), além de um breve manual do usuário (por exemplo, se é preciso usar alguma tecla para alguma funcionalidade para a qual não há ajuda textual na interface gráfica). Para essa parte do trabalho, `p0/README` conterá somente a identificação do(s) autor(es) e se as atividades **A1** e **A2** acima foram ou não realizadas.

4 Passo 4: Entrega do programa

O trabalho deve ser entregue via AVA em arquivo único compactado (somente um arquivo por grupo), chamado `p0.zip` (nome(s) do(s) autor(es) vão no arquivo `README`). Preste atenção, vamos repetir para não deixar dúvidas: a submissão deve ser feita em um arquivo compactado do tipo `zip` cujo nome deve ser `p0.zip`. Este deve conter:

- o diretório `p0` com o código-fonte completo e com o arquivo executável `p0.exe`, mas **sem** quaisquer arquivos intermediários de backup ou resultantes da compilação, tanto em `p0` como em seus subdiretórios; e
- o arquivo `README`.

Não serão considerados e terão nota zero programas plagiados de qualquer que seja a fonte, mesmo que parcialmente, exceção feita ao código fornecido pelo professor. O trabalho deve ser desenvolvido em grupos de **dois** estudantes. No caso de turma com número ímpar de estudantes, será aceito somente **um** trabalho com um único autor.

5 Lista de objetivos

Cada parte do trabalho terá uma lista de objetivos que servirá como base para sua avaliação. A verificação dos objetivos da **Parte 0** levará em conta se:

- ☐ O arquivo `p0.zip` foi submetido com os arquivos corretos.
- ☐ `README` contém as informações solicitadas.
- ☐ O(s) nome(s) do(s) autor(es) aparece(m) na interface com o usuário.
- ☐ A interface com o usuário foi estendida com controles para especificação da posição e da escala do triângulo.