Avaliação de diferentes metodologias para o ensino de computação gráfica em nível superior

Arthur Carvalho Oliveira[1], **Bruno Augusto Nassif Travençolo**[1]

[1]Faculdade de Computação - Universidade Federal de Uberlândia (UFU)
Av. João Naves de Ávila, 2.121 - Bairro Santa Mônica
Uberlândia - MG - Brasil - CEP 38400-902

Resumo. *Este estudo realiza uma avaliação comparativa das metodologias de ensino dessa disciplina em cursos superiores considerados referência no exterior: MIT, Utah, British Columbia, e Carnegie Mellon. A pesquisa adotou uma metodologia empírica baseada na implementação prática de 14 projetos propostos por essas instituições. O desenvolvimento abrangeu desde a rasterização via software e transformações geométricas até simulações físicas e Ray Tracing. Os resultados evidenciaram perfis pedagógicos distintos, contrastando a curva de aprendizado de abstrações web (WebGL/Three.js) com a exigência arquitetural de ferramentas de baixo nível (C++/OpenGL). Como desdobramento dessa análise empírica, este estudo delineia as diretrizes de um plano de ensino híbrido para a disciplina. A proposta recomenda a adoção de tecnologias de mais baixo nível. A fim de mitigar a dificuldade inicial, sugere-se fornecer uma quantidade gradualmente decrescente de código-base no início de cada projeto, unindo assim a acessibilidade ao rigor técnico.*

1. Introdução

O ensino de Computação Gráfica é fundamental para a formação de profissionais de computação, especialmente diante de sua ampla aplicação em filmes, séries, jogos e outras mídias. Além de aproximar conceitos abstratos de algoritmos e estruturas de dados de aplicações práticas, a disciplina integra álgebra linear com soluções computacionais. Esse contato estimula o aluno a explorar metodologias e algoritmos para representar um mundo virtual, elevando a qualidade do ensino e gerando práticas pedagógicas que podem ser replicadas em outras instituições.

O objetivo principal deste trabalho é comparar diferentes metodologias de ensino de Computação Gráfica no ensino superior. Foram analisados cursos renomados, como o **6.837 - Computer Graphics** do MIT [MIT 2020], **CS4600** da University of Utah [University of Utah 2020], **CMU 15-462/662** da Carnegie Mellon University [CMU 2020], e o **CPSC 314** da University of British Columbia [UBC 2020], que são cursos de referência em Computação Gráfica e não têm apenas uma abordagem de ensino; enquanto [CMU 2020] e [MIT 2020] usam C++ e OpenGL com ferramentas auxiliares (como GLUT no MIT), a CMU possui uma biblioteca 3D própria chamada Scotty3D [CMU Graphics 2020], e [University of Utah 2020] e [UBC 2020] usam WebGL junto com JavaScript. Essa comparação evidencia a necessidade de pesquisa sobre o método de ensino mais eficaz em Computação Gráfica.

2. Trabalhos Relacionados

O trabalho [Suselo et al. 2017] focam em consolidar o conhecimento existente sobre o ensino de Computação Gráfica, identificando problemas comuns e soluções propostas

pela comunidade acadêmica. Foram analisados dados de quatro repositórios científicos bem conhecidos – ACM Digital Library, IEEExplore, SpringerLink e ScienceDirect. Após buscas e filtragem, 45 artigos relevantes foram selecionados e, a partir desses, foram categorizados quatro principais problemas de ensino de: Conhecimento insuficiente de matemática [Gálvez et al. 2008, Hui et al. 2012, Zhou et al. 2010] e programação básica [Lowther and Shene 2000, Papagiannakis et al. 2014]; dificuldades em compreender transformações, projeções e modelagem geométrica 3D [Elyan 2012]; dificuldades em resolver problemas lógicos [Hitchner and Sowizral 2000, Talton and Fitzpatrick 2007] e em estabelecer a conexão entre teoria, programação, aplicação e efeitos visuais [Stephenson and Taube-Schock 2009]; e os estudantes tornaram-se aprendizes passivos, não interagindo muito com colegas nem com o corpo docente [Peternier et al. 2006, Martí et al. 2006].

O trabalho de [Balreira et al. 2017] focam em responder à pergunta sobre o que está sendo ensinado nos cursos introdutórios de computação gráfica ao redor do mundo. Foram usados como base para pesquisa 20 cursos de graduação de nível superior com base na contribuição de artigos para a SIGGRAPH (*Special Interest Group on Computer Graphics and Interactive Techniques*), Siggraph Asia e Eurographics (*European Association for Computer Graphics*), considerando que o nível de pesquisa conduzido se transferiria para o ensinamento do tópico na universidade. A coleta foi realizada usando como fonte de informações dados públicos online.

3. Desenvolvimento

Esta seção apresenta a avaliação das metodologias adotadas pelas instituições de referência. A análise está estruturada de forma cronológica, avançando semanalmente pelos conteúdos ministrados em cada curso. Como todas as disciplinas avaliadas possuem um forte foco prático baseado na aprendizagem por projetos (PBL), optou-se por realizar a implementação de parte desses trabalhos. Essa resolução prática foi fundamental para propiciar uma análise mais aprofundada e realista acerca do nível de dificuldade exigido e dos tópicos técnicos priorizados por cada instituição.

A atribuição dos rótulos de dificuldade (Fácil, Médio, Difícil e Muito difícil) para cada projeto foi fundamentada em uma metodologia de avaliação empírica. Essa classificação levou em consideração a experiência direta de implementação, baseando-se em dois fatores principais: o nível de complexidade técnica exigida e o tempo necessário para a resolução de cada trabalho proposto.

A Tabela 1 mostra um resumo dos projetos desenvolvidos. É feita uma análise da dificuldade de execução do projeto.

Tabela 1. Comparativo dos projetos desenvolvidos.

Projeto	**Conteúdos**	**Descrição**	**Dificuldade**
Compositing Images (UTAH)	*Alpha blending*, manipulação de *buffers*, hierarquia 2D.	Implementação de manipulação de *buffers* de cor e composição de camadas com *alpha blending*. Foca no entendimento de transparência e ordem de renderização.	**Fácil**

Tabela 1 – Continuação da página anterior

Projeto	Conteúdos	Descrição	Dificuldade
OpenGL Mesh Viewer (MIT)	*Parsing* de OBJ, *pipeline* de renderização com OpenGL legado e cálculo de normais.	Desenvolvimento *parser* de arquivos OBJ e processamento de normais baseado em cada face de triângulos. Foca na transição de dados para a GPU e iluminação básica.	**Fácil**
Curves and Surfaces (MIT)	Curvas de Bézier, *B-Splines* e superfícies de revolução.	Implementação algorítmica de avaliação de curvas paramétricas (Bézier e *Splines*). Envolve o estudo de continuidade geométrica e interpolação de superfícies.	**Muito difícil**
Transformations (UTAH)	Transformações, matrizes homogêneas, coordenadas 2D.	Aplicação prática de álgebra linear para manipulação de cena. Foca no domínio da composição de matrizes homogêneas e transformações de coordenadas.	**Fácil**
Hierarchical Modeling & SSD (MIT)	*Forward kinematics*, *skinning* (SSD) e árvore de cena.	Implementação de animação baseada em esqueleto (*skinning*). Foca no cálculo de matrizes relativas e na deformação ponderada de malhas para personagens.	**Difícil**
DrawSVG (CMU)	Rasterização, coordenadas baricêntricas, *anti-aliasing* e *mipmapping*.	Construção de um *pipeline* de rasterização em *software*. Foca em *anti-aliasing* via *supersampling* e em filtros de textura.	**Médio**
Intro (UBC)	*Shaders* GLSL (*vertex/fragment*) e *pipeline* básico utilizando Three.js.	Desenvolvimento de *shaders* iniciais em GLSL e integração com Three.js. Foca no entendimento básico dos estágios de renderização na GPU.	**Fácil**
Physically-based simulation (MIT)	Integração numérica (Euler/*Trapezoidal*), EDOs e malhas deformáveis.	Implementação de simuladores dinâmicos baseados em física. Foca na modelagem de forças físicas (molas e gravidade).	**Médio**
Curves (UTAH)	Curvas de Bézier e avaliação paramétrica de Bézier na GPU.	Migração do cálculo geométrico de curvas para a GPU. Foca na avaliação de equações paramétricas diretamente no *vertex shader* para ganho de *performance*.	**Médio**
Transformations (UBC)	*Forward kinematics* e transformações em hierarquias articuladas.	Desenvolvimento de animações procedurais. Foca no controle de movimentos articulados e utilizando os conceitos de *forward kinematics*.	**Fácil**
Ray casting (MIT)	Interseção de primitivas, coordenadas baricêntricas e geração de raios.	Implementação matemática das interseções raio com geometria básicas. Foca na renderização por *pixel* de uma cena através do traçado de raios.	**Difícil**

Tabela 1 – Continuação da página anterior

Projeto	Conteúdos	Descrição	Dificuldade
Triangular Meshes (UTAH)	Transformações MVP, texturização de malhas e uso de WebGL.	Renderização em tempo real de malhas texturizadas. Foca no processamento do sistema *Model-View-Projection* (MVP) e no gerenciamento de memória na GPU.	**Médio**
Shading (UTAH)	Modelo Blinn-Phong e BRDF.	Implementação do modelo de iluminação Blinn-Phong. Foca no cálculo das componentes difusa e especular calculado no espaço da câmera via *shaders*.	**Médio**
Ray tracing (MIT)	Sombras, reflexão/refração, BVH e paralelização *multi-thread*.	Expansão do renderizador para suportar efeitos de iluminação global. Foca na simulação física da luz e na otimização arquitetural via particionamento espacial (BVH) e *threads*.	**Muito difícil**

4. Conclusão

O objetivo central deste trabalho foi avaliar metodologias de ensino de Computação Gráfica no ensino superior, identificando os tópicos, ferramentas e abordagens predominantes em cursos de referência mundial.

A análise de dificuldade revelou perfis distintos. Os projetos do MIT são os mais exigentes matematicamente, cobrindo superfícies de revolução, simulação física com EDOs e ray tracing com estrutura BVH. UTAH se destaca por uma progressão mais gradual, aumentando progressivamente a complexidade e a dificuldade de cada projeto; CMU aprofunda os fundamentos algorítmicos da rasterização por meio do projeto DrawSVG, capacitando o aluno a implementar um pipeline gráfico completo operando estritamente em software (CPU); e UBC oferece a entrada mais acessível via Three.js, apostando em projetos lúdicos e de alto apelo visual que engajam o aluno pelo resultado imediato, servindo como uma excelente porta de entrada para a área. Conclui-se que não existe uma metodologia superior absoluta, mas sim diferentes balanceamentos entre rigor técnico, escopo e acessibilidade.

Embora as evidências confirmem a inexistência de uma metodologia universalmente superior, a vivência prática ao longo desta pesquisa permite delinear que um modelo pedagógico híbrido apresenta-se como o mais eficaz.

Nesse contexto, nota-se o grande valor de adotar, desde o primeiro projeto, um ambiente de programação de alto controle e baixo nível. Essa base seria formada por C++ e OpenGL, preferencialmente com o apoio de bibliotecas como o GLFW para gerenciar o loop de renderização. Para reduzir a dificuldade inicial com essas ferramentas, a disciplina utilizaria a entrega de um código inicial estruturado. Assim, nas primeiras etapas, a complexidade tecnológica fica isolada pela infraestrutura já fornecida pelo curso, permitindo que o aluno foque exclusivamente na matemática e nos conceitos. Na progressão ideal da matéria, a quantidade de código entregue pronta diminui a cada novo projeto. Conforme o aluno consolida o seu aprendizado, a responsabilidade de programar a arquitetura do sistema passa gradualmente para ele. É nesse momento que o rigor e o controle direto sobre o hardware, oferecidos pelo C++ e OpenGL, tornam-se essenciais e são compreendidos pelo estudante.

Em resumo, o trabalho fornece um mapeamento técnico e prático das metodologias de ensino adotadas por instituições de referência na área. Os resultados geram uma base documentada para orientar a reestruturação curricular de disciplinas de Computação Gráfica, servindo como material de apoio prático para coordenadores e docentes da área.

Referências

Balreira, D. G., Walter, M., and Fellner, D. W. (2017). What we are teaching in introduction to computer graphics. In *Proceedings of the European Association for Computer Graphics: Education Papers*, EG '17, page 1–7, Goslar, DEU. Eurographics Association.

CMU (2020). 15-462/662 computer graphics, fall 2020.

CMU Graphics (2020). Scotty3d.

Elyan, E. (2012). Enhanced interactivity and engagement: Learning by doing to simplify mathematical concepts in computer graphics and animation. In *Proceedings of the 2012 IEEE Global Engineering Education Conference (EDUCON)*, pages 1–8.

Gálvez, A., Iglesias, A., and Corcuera, P. (2008). An introductory computer graphics course in the context of the european space of higher education: A curricular approach. In Bubak, M., van Albada, G. D., Dongarra, J., and Sloot, P. M. A., editors, *Computational Science – ICCS 2008*, pages 715–724, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hitchner, L. and Sowizral, H. (2000). Adapting computer graphics curricula to changes in graphics. *Computers & Graphics*, 24:283–288.

Hui, P., Liu, S., Xiu, J., and Zhai, R. (2012). Ability-oriented teaching reform for computer graphics. In *Proceedings of the 7th International Conference on Computer Science & Education (ICCSE)*, pages 1908–1911, Melbourne, Australia.

Lowther, J. and Shene, C.-K. (2000). Rendering + modeling + animation + postprocessing = computer graphics. *Journal of Computing Sciences in Colleges*, 16:20–28.

Martí, E., Gil, D., and Julià, C. (2006). A pbl experience in the teaching of computer graphics. *Computer Graphics Forum*, 25(1):95–103.

MIT (2020). 6.837 computer graphics, fall 2020.

Papagiannakis, G., Papanikolaou, P., Greassidou, E., and Trahanias, P. (2014). glGA: an OpenGL Geometric Application Framework for a Modern, Shader-based Computer Graphics Curriculum. In Bourdin, J.-J., Jorge, J., and Anderson, E., editors, *Eurographics 2014 - Education Papers*. The Eurographics Association.

Peternier, A., Thalmann, D., and Vexo, F. (2006). Mental vision: A computer graphics teaching platform. In Pan, Z., Aylett, R., Diener, H., Jin, X., Göbel, S., and Li, L., editors, *Technologies for E-Learning and Digital Entertainment*, pages 223–232, Berlin, Heidelberg. Springer Berlin Heidelberg.

Stephenson, B. and Taube-Schock, C. (2009). Quickdraw: bringing graphics into first year. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, SIGCSE '09, page 211–215, New York, NY, USA. Association for Computing Machinery.

Suselo, T., Wünsche, B., and Luxton-Reilly, A. (2017). The journey to improve teaching computer graphics: A systematic review.

Talton, J. O. and Fitzpatrick, D. (2007). Teaching graphics with the opengl shading language. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, page 259–263, New York, NY, USA. Association for Computing Machinery.

UBC (2020). Cs-314 computer graphics.

University of Utah (2020). Cs-4600 computer graphics, fall 2020.

Zhou, J., Ye, M., and Huang, C.-L. (2010). Reform of computer graphics teaching method. In *2010 International Forum on Information Technology and Applications*, volume 3, pages 222–225.