Curso Prático de Introdução à Programação em Linguagem C++ Aplicada ao Desenvolvimento de Jogos em 3D

MARCELO DE OLIVEIRA JOHANN
MARCUS KINDEL
CINTYA BORTOLAZZO

Pontifícia Universidade Católica do Rio Grande do Sul - Campus Universitário II
Faculdade de Administração. Contabilidade e Informática - Departamento de Informática
{johann, kindel, cintya}@pucrs.campus2.br

Abstract

This paper describes an introductory course on object-oriented programming applied to the development of computer games with 3D modeling. The use of games as the goal allows us to approach both basic concepts and complex practical problems of programming, while keeping the students interested on the subject. A survey of standard techniques for game development are presented with running examples at the course, although the objective is not to cover all aspects of computer graphics, artwork, and other advanced techniques required for finished products. The paper describes the motivation, objectives and the structure of this 64 hours undergrad course, as well as the resources used to implement it.

Keywords: *programming, object orientation, C++, education, games*

1 Introdução

Jogos de computador apresentam uma necessidade acentuada de poder computacional. Os recursos visuais, sonoros, e de controle desejados estão sempre além da tecnologia disponível. Por esta razão, os programadores de jogos sempre extraíram o máximo da tecnologia existente, por exemplo, usando modos de operação especiais das placas de vídeo, programando em linguagem *assembly* para ter código mais otimizado, e principalmente usando tabelas de resultados pré-computados. Algumas dessas técnicas são dificilmente dominadas por programadores iniciantes ou mesmo por aqueles já experientes em outras áreas.

Os primeiros jogos disponíveis utilizavam apenas modelos bidimensionais ou quase unidimensionais, onde o protagonista pode apenas mover-se para a direita ou esquerda. Com a evolução tecnológica, muitos jogos atualmente empregam o que há de mais moderno em *hardware* e *software* para apresentar universos virtuais em 3 dimensões (3D) com uma infinidade de recursos. Os processadores gráficos de placas de vídeo para computadores domésticos possuem a capacidade de desenho de milhões de faces em 3D (triângulos) por segundo, usando efeitos de textura, iluminação, neblina, fazendo recorte, transformações, cálculo de visibilidade e transparência automaticamente. Esse fator simplifica muito a programação de jogos, e com o auxílio de bibliotecas de *software* denominadas de *engines*[3,4], torna-se possível desenvolver aplicações visualmente ricas em níveis de abstração mais altos. Acredita-se que esta contínua evolução irá eliminar a grande necessidade de economizar recursos dos primeiros jogos, e que os profissionais do futuro devam estar preparados para trabalhar com modelos de jogos complexos em 3D.

Do ponto de vista de *software*, um jogo é bastante semelhante a processos de simulação e sistemas de tempo real. Um processo de simulação deve acompanhar a geração e

execução de eventos no tempo, testando condições segundo as regras do domínio em questão. Um sistema de tempo real tem como principal característica o fato de que deve atender a determinadas restrições de temporização externa. Do ponto de vista de projeto, as tecnologias mais importantes de programação são hoje intensivamente utilizadas na implementação de jogos. São praticamente indispensáveis: orientação a objetos, *design patterns*[2], *templates*[1], *containers*[1], iteradores[1], ponteiros seguros, tratadores de eventos, sistemas de janelas, programação concorrente, comunicação, bases de dados, etc...

Além do uso de *hardware* e domínio da tecnologia de *software*, há o projeto gráfico e semântico do jogo, que tendem a requerer a maior parte do esforço de projeto principalmente em jogos com modelos tridimensionais. Em nosso caso, todo este trabalho de arte e roteiro não é de interesse para a realização do curso.

2 Motivação

Conceitualmente, um bacharel em Ciência da Computação é um profissional muito superior a um simples programador especializado em codificar pequenos algoritmos em uma determinada linguagem. Entretanto, a maioria do conteúdo computacional é expresso sob forma de programas, mesmo em casos onde este seja posteriormente implementado em *hardware*. Assim, é de fundamental importância que um bacharel tenha destreza em programação utilizando as mais variadas técnicas e linguagens disponíveis. Apesar desta necessidade básica, o aprendizado de programação não é trivial, leva muito tempo para ser desenvolvido, e muitas vezes não atinge um grau de maturidade mesmo para alunos que se formam em nossas Universidades. É conhecida a dificuldade de alunos principiantes para compreender técnicas básicas de algoritmos e programação, e as necessidades e vantagens de técnicas mais avançadas como orientação a objetos ou outras abstrações se eles não tiveram uma boa experiência. Por esta razão, os professores universitários estão continuamente em busca de novas técnicas de ensino, e principalmente de novos problemas que motivem os alunos, partindo da aplicação, da necessidade, para que eles entendam a utilidade de tão complexos mecanismos, como os disponíveis em linguagens como C++[1].

A aplicação de jogos tem esta característica especial de atração para os jovens estudantes. A maioria dos estudantes já jogou ou é acostumada a jogar regularmente jogos de computador e *vídeo games*. E como todos os jovens, os alunos têm curiosidade e criatividade potenciais para trabalharem em assuntos relacionados aos seus maiores interesses, ou àquilo em que vêem atração, não fazendo parte de suas obrigações. Segundo pesquisa realizada no Campus de Uruguaiana, a grande maioria dos alunos demonstrou forte interesse em aprender técnicas usadas para a construção de jogos. E estas técnicas incluem, obrigatoriamente, a maior parte do conteúdo de programação que se deseja ensinar. Assim, o potencial de aprendizado por ela oferecido é muito animador, e espera-se que seu emprego venha a sanar deficiências deixadas por outras abordagens.

3 Objetivos

Gerais:

- Ensinar e praticar a programação orientada a objetos em C++;
- Ensinar as tecnologias para desenvolvimento de jogos com modelos 3D.

Específicos:

- Desenvolver capacidade de definir classes e programar usando objetos;
- Apresentar motores 3D disponíveis e como se os usa;
- Conhecer a arquitetura e temporização de um jogo;
- Conhecer e saber como usar as técnicas de: modelagem e visualização em 3D, movimentação, tratamento de eventos, detecção de colisões, física, animação, representação de superfícies e terrenos;
- Dar visão da indústria e mercado de jogos, objetivos e fatores de sucesso.

4 Metodologia de Ensino

O curso é prático, em laboratório, com computadores individuais, todo baseado em exemplos. Os exemplos iniciais são bem simples, com o menor número de linhas de código que permita criar e exibir objetos tridimensionais. Cada exemplo é experimentado e alterado de diversas formas pelos alunos para dar aos mesmos o domínio do programa, de seu funcionamento, e do que está expresso na linguagem de programação. Os desafios lançados aos alunos, como, por exemplo, completar um determinado objeto que está pela metade, estimulam sua criatividade e desejo de programar cenas cada vez mais complexas, assim dominando as técnicas ensinadas e permitindo ao mesmo tempo terem suas próprias conclusões e aprimoramento de suas técnicas de programação. A partir destes exemplos, os demais conceitos são inseridos um a um, construindo ambientes mais complexos, até incluírem movimentos, controle, e se assemelharem a jogos simples. O processo é repetido várias vezes com novos objetivos.

Após estas experiências, são propostos problemas mais genéricos de jogos, como movimento simultâneo de diversos objetos, e então apresentadas algumas opções de arquitetura do programa para tratar esta simulação em tempo real. Dá-se prioridade ao estudo de uma arquitetura que atende às necessidades, para que também com ela os alunos se acostumem, embora saibam que há outras formas possíveis de estruturar o programa. Durante todo este processo, a orientação a objetos é usada intensivamente, já que os ambientes com que se trabalha são 100% baseados nela. Por esta razão os exemplos simples iniciais são conduzidos, passo a passo, para firmar o domínio básico de orientação a objetos. Assuntos mais avançados ou exceções, como herança múltipla e funções virtuais, são evitados, tanto quanto possível. O objetivo principal é dar poder seguro de expressão com orientação a objetos, tornando-a uma maneira intuitiva de pensar. Finalmente, são abordados os assuntos mais avançados da tecnologia de jogos, como uso de animações, detecção de colisões e sistemas de física, descrições de superfície e terrenos e uso de diferentes níveis de detalhe, também com uso de exemplos práticos, embora não haja tempo para construir um jogo completo que envolva todos estes recursos.

5 Conteúdo Programático

Os tópicos abordados no curso são, basicamente, os descritos abaixo. A critério dos professores, os itens poderão ser simplificados, estudados com mais profundidade ou novos tópicos poderão ser inseridos, dependendo do progresso dos alunos no decorrer do curso. Seguem os tópicos previstos, organizados por semana:

Sem. 1 – Introdução, ambiente, caracterização do motor **Gizmo3D**, triângulos, câmera, texturas

Sem. 2 – Vértices e arestas, transformações, arquivo de descrição de objetos

Sem. 3 – Funções *callback OnIdle* e *OnTick*

Sem. 4 – Controles de teclado e de movimentos

Sem. 5 – Primeiros jogos – **estoque**

Sem. 6 – Movimentos, Tempo e Arquitetura

Sem. 7 – Movimentos, Tempo e Arquitetura

Sem. 8 – Rede, Mensagens, Jogo de duplas

Sem. 9 – Estrutura do **Crystal Space**

Sem. 10 – Arquivo de mapa, animações

Sem. 11 – Representação de terrenos

Sem. 12 – Detecção de colisões

Sem. 13 – Efeitos diversos (som, neblina, representações procedurais, etc.)

Sem. 14 – Sistemas de Física

Sem. 15 – Superfícies curvas; elaboração de um projeto de jogo;

Sem. 16 – Tipos, Objetivos[5] e Mercado de Jogos; início da implementação de um jogo;

6 Infra-estrutura

Para realização do curso, é utilizado um laboratório com 20 máquinas (Pentium III 1.3GHz e Pentium IV1.7GHz), todas com 128 MB RAM, placa aceleradora gráfica e sistema operacional Linux Red Hat 7.3. Os *engines* utilizados são o **Gizmo3D**[3] e **Crystal Space**[4]. As licenças para uso desses sistemas são: livre para Crystal Space (LGPL) e comercial ou livre (para fins não comerciais) para o Gizmo3D. Tanto um como outro sistema oferecem quase a totalidade das características descritas a seguir:

Plataformas Linux e Win32, tratamento de exceções, algumas *design patterns*, *templates* para dicionários, listas, filas, vetores dinâmicos; clonagem, *threads* e *mutexes*, *RTTI*, *rendering* de múltiplas passagens, incluindo iluminação, texturas, sombras, neblina, transparência; superfícies paramétricas com níveis de detalhe automáticos, seleção e detecção de colisões, controle de animações, e sistemas de bases de dados para leitura de alguns formatos de imagens, animações e mapas.

Na primeira metade do curso os alunos trabalham com o Gizmo3D, pois este possibilita a visualização dos resultados com apenas poucas linhas de código. Isso favorece aos alunos a compreensão de cada mecanismo apresentado. Na segunda metade do curso, já com o domínio de algumas técnicas fundamentais, poderão trabalhar com o Crystal Space, *engine* que oferece construções mais poderosas. Este não é utilizado primeiramente porque a estrutura dos programas é mais complexa, e os alunos demorariam para dominar as abstrações nele utilizadas.

7 Conclusões

No tempo de submissão deste artigo, a primeira edição do curso se encontra em sua segunda semana de funcionamento, seguindo o cronograma correto. Por esta razão, os resultados em termos de avaliação de aprendizado e trabalhos práticos implementados são escassos. A Fig. 1 apresenta dois modelos 3D criados por alunos após a quarta aula. Já a Fig. 2 apresenta um pequeno jogo completo, o Controle de Estoque, usado como exemplo no início da quinta semana, o qual define novas classes e já usa mais recursos da linguagem C++.

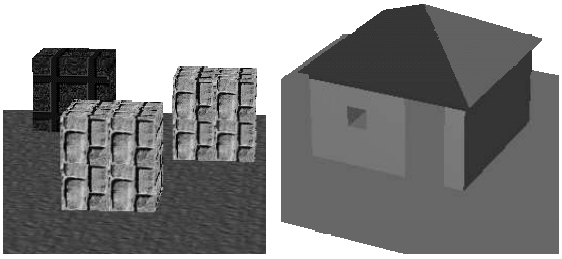

Figura 1 – Modelos simples criados por alunos com textura e iluminação

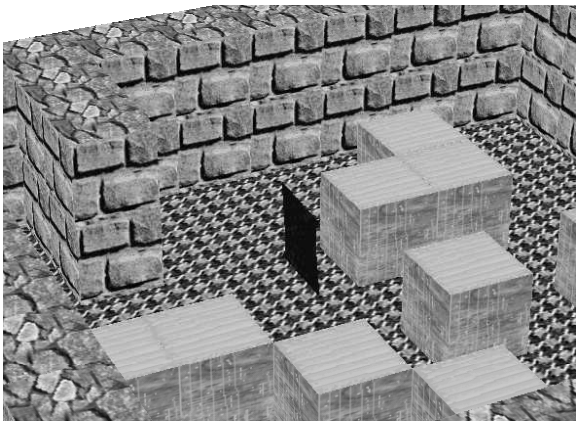

Figura 2 – Jogo "Controle de Estoque", com funcionalidade completa

8 References

1. Bjarne Stroustrup. C++ Programming Language. Reading: Addison Wesley. 1997.
2. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns – elements of reusable object-oriented software. Reading: Addison Wesley, 1995.
3. ToolTech Software. Gizmo3D, http://www.tooltech-software.com (21/08/2002).
4. Tyberghein, J., Sunshine, E. (Adm.). Crystal Space http://sourceforge.net/projects/crystal (21/08/2002).
5. Crawford, Chris. The Art of Computer Game Design. Washington State University at Vancouver - Department of History, 1982.