C++ Orientado a la programación de Videojuegos

Temario

Este temario es únicamente un roadmap de lo que creo que debe quedar bien claro en cuanto a C++ se refiere. Como puede verse no incluye nada específico sobre videojuegos. La idea es explicar los conceptos básicos de videojuegos y hacer videojuegos sencillos, como un Pong, un arcanoid, Tetris, el Juego de La Vida, etc. Pero poder hacer juegos bien requiere de una experiencia en el lenguaje que seguramente en un principio se carece. Así que en las clases se dará C++11 desde el principio, explicando todos los conceptos para que os quede bien claro como funciona. Mi objetivo no es que seáis expertos en C++, sino que entendáis que es lo que hacéis, y seais conscientes de lo que hacen vuestros programas, el compilador, y el lenguaje. Por supuesto cuanto más demos de juegos mejor, y todos los ejemplos, prácticas, etc los haré sobre juegos o cosas relacionadas con ellos.
El punto 5 se considera C++ avanzado, así que no pasaría nada si no llego a darlo del todo, pero hasta el punto 4 se considera muy importante (Importante quiere decir que si no tienes perfectamente claro hasta ahí no sabes C++).

1. **Sistema Operativo, Hardware, etc: ¿Donde y cómo se ejecuta mi programa?**
    1. CPU, RAM, cache
        - CPU: Registros, ALU, etc
        - Jerarquía de memoria
        - Rendimiento: La CPU es gratis, la memoria no.
        - Proceso. Memoria de un proceso.
    2. Datos y tipos de datos: ¿Cómo interpreto este churro de bytes?
        - ¿Que significa el valor 0x00000007?
        - Punteros: ¿Por que son tan complicados si realmente es la misma tontería?
    3. Paso por valor y paso por referencia: Abstracciones software
2. **Repaso de "C++ FDI": Lo que se supone que sabéis de C.....++**
    1. Sintaxis C++: `for, while, if...`
3. **Lo que hay que saber antes de empezar a usar C++**

1. Namespaces:
    - C namespaces: `typedef struct`
    - C++ namespaces
        - Why `using namespace std` is considered a bad practice?
        - When and how to use namespaces?
        - unqualified.ids, qualified-ids, nested-name-specifiers
        - ADL: ¿Por qué `std::cout << "HOLA!!!" << std::endl` funciona, cuando no debería?

2. Storage class specifiers
    - El modelo de compilación de C y C++
        - Translation units
        - Declaración vs definición
        - Enlazado

    - Specifiers

3. Programación Orientada a Objetos
    - Class memory layout
    - `class` vs `struct`
    - El principio de responsabilidad única
    - Encapsulación: Pensar primero, encapsular después
    - Constructores: Initialization list.
    - Constructor de copia/move
    - Operador de asignación
    - Destructor
    - The Rule Of Three/Five
    - The Rule Of Zero

4. RAII
    - Recursos. Gestión de recursos.
    - smart pointers

5. Operators overloading
    - Operadores unarios vs operadores binarios

6. Herencia
    - Class memory layout (Revisited). Empty base class optimization.
    - La herencia no es tu única herramienta, úsala solo cuando tenga sentido.
    - Herencia múltiple: Una herramienta potente pero peligrosa
        - Problema del diamante

7. Polimorfismo
    - ”Java sabe a que función llamar”: No, la magia no existe
    - vtable. vptr. Class memory layout (Revisited, again…)
    - Destructor virtual
    - Muy bonito si, pero, ¿y el rendimiento?

4. **Templates**

    1. Templates
        - ¿Que son y como funcionan?
        - ¿Por qué solo puedo escribir templates en los archivos de cabecera?

    2. Sintaxis en profundidad
        - Type parameters
            - `typename` vs `class`
        - Non-type parameters
        - Template template parameters
        - Template-dependent names: ¿Cuando tengo que usar `typename` o `template`?

    3. Syntaxis (Bis)
        - Optional template parameters
        - Variadic templates

    4. Template specialization
        - Template specialization
        - Partial template specialization

    5. Function templates
        - Template argumment deduction
        - En C++ todo son templates, hasta ahora estábais engañados: ¿Que es `std::endl`?

5. **La Biblioteca Estándar**

    1. Streams: `<iostream>`, `<sstring>`, etc
    2. Contenedores: `<vector>`, `<unordered_map>`, etc
    3. C++ es un lenguaje de muy alto nivel (Más que Java): `<iterator>`, `<algorithm>`, `<functional>`, etc
    4. Y sigue evolucionando: `<chrono>`, `<thread>`

6. **“Ya se Kung Fu” “Demuéstralo”**

    1. El compilador es más listo que tu, deja que él haga el trabajo
    2. CRTP
    3. Static-polymorphism
    4. Barton-Nackman trick
    5. Traits and mixins. Component-based dessigns.
    6. Policy-based dessigns
    7. Typelists
    8. DSELs
        - Blitz++, expression templates
        - Boost::spirit
        - Multidimensional Analog Literals

9. Extreme Template Metaprogramming
    - Turbo
    - Boost::mpl

Bibliografía:

Antes de nada debéis entender una cosa: Cuando aquí pongo *“básico”*, *“medio”*, y *“avanzado”*, es que de verdad se consideran de nivel básico, medio, o avanzado; no me refiero al nivel que vayamos a dar en éste curso.
Creo que ya somos todos mayorcitos, y estaría bien que se nos expliquen las cosas como verdaderamente son. Ese es uno de los objetivos de éstas clases.
De echo, podeis comprobar que ésta lista es prácticamente idéntica a la lista de libros recomendados en stackoverflow.

Esta lista no incluye todos los libros que os he pasado, incluye solo los que creo son más importantes. La lista es una guía de aprendizaje; una lista con los libros que creo se deberían leer (En ese orden) para aprender bien.

C++ básico:

- ***”Thinking in C++”*** : Es un libro escrito para programadores de C que quieren pasar a C++. Dado que lo que enseñan en la facultad es básicamente C (Mal C), éste libro es perfecto para vosotros.
  Explica detalladamente todos y cada uno de los conceptos del lenguaje.
  Aunque es de hace algunos años y puede que haya cosas que ahora no sean correctas del todo, sigue siendo un muy buen libro.
  Está disponible en la biblioteca de la facultad.

- ***”The C++ Programming Language”*** : Es el libro del creador del lenguaje, Bjarne Stroupstrup.
  Aunque no es un libro para aprender, es un libro de referencia y demostración del lenguaje, contiene buenas anotaciones y explicaciones para programadores noveles.
  Mi recomendación es que lo leáis en paralelo junto con “Thinking”, hay cosas que puede que estén mejor explicadas en uno, y viceversa.
  En la biblioteca de la facultad está disponible la tercera edición, pero no os recomiendo que lo cojáis. Es de hace bastantes años, y ya me parece suficiente que tengáis un libro anticuado, más es demasiado. El que os he pasado en PDF es la última versión. Por supuesto si os pica la curiosidad podéis cojer ambos para comparar y ver como ha evolucionado el lenguaje.

C++ medio

Estos libros están pensados para gente que ya tiene asentadas las bases del lenguaje, entiende como éste funciona, y ya ha hecho cosillas con él.

- ***"Effective C++"*** , ***"More Effective C++"*** : Son un par de libros con una serie de puntos del estilo *"you should do"* o *"you shouldn't do"*, explicados en profundidad. Muy recomendados una vez que "sabes" C++.

C++ Avanzado

Aquí nos adentramos en los rincones oscuros de C++. Como programador normal de C++ no es absolutamente necesario conocer ésto, pero si quieres saber lo potente (En cuanto a rendimiento y expresividad) puede llegar a ser C++, tienes que conocer éstas cosas.

- ***"Modern C++ Dessign"*** : Tras la publicación de "Dessign Patterns", de la Banda de Los Cuatro Payasos (Perdón, la Banda de Los Cuatro), aparecieron un montón de libros que básicamente se resumían en *"Vale tíos, que os habéis fumado? Todas esas chorradas de diseño están muy bien, pero eso HAY QUE IMPLEMENTARLO DE ALGUNA MANERA. Imaginar diseños desde las nubes está muy bien, pero en la práctica no funciona."*
  Este es uno de esos libros. Es un libro que cubre metodologías de diseño con C++, implementaciones reales y eficientes de patrones de diseño, y discusiones sobre por qué las chorradas de GoF en la realidad no funcionan.
  Dejándo a un lado el tema del diseño, las tecnologías y metodologías que muestra para resolver esos problemas muestran las capacidades de C++, que en muchos aspectos ningún otro lenguaje tiene.
  Es el libro que te abre los ojos a lo que C++ puede llegar a hacer.

Bonus

Además de esos libros, os he incluido una serie de PDFs que cubren tres aspectos: APIs gráficas de bajo nivel, sistemas gráficos (Sistemas de ventanas), y teoría y algoritmos sobre computación gráfica en general.
Las APIs gráficas y los sistemas de ventanas los he puesto para que seáis conscientes de lo que hay debajo cuando usais programillas en Windows y Linux, y de todo lo que te abstrae las APIs de programación de juegos de medio y alto nivel.