## Bibliotecas de Jogos

Uma biblioteca de jogos ou game engine pode ser vista como uma caixa de recursos e ferramentas para a construção de um jogo. Você pode criar um jogo sem uma biblioteca básica, assim como você pode criar uma mesa de madeira sem pregos, martelos, parafusos, chaves de fenda e serras, mas as vantagens que as ferramentas proporcionam justificam chamá-las de necessárias.

O nível dessas ferramentas varia: algumas engines se limitam a códigos, ou seja, constantes, variáveis, funções e classes relacionadas, mas outras contam com interfaces gráficas que possibilitam o desenvolvimento de um jogo sem codificação alguma. De qualquer forma, uma game engine precisa proporcionar, no mínimo, ferramentas para manipular sons, imagens (texto, imagens, etc), memória (dados) e controle (teclado, mouse, etc).

## Saga Game Library

Há duas finalidades que justificam o desenvolvimento de uma tecnologia que já existe: 1. O estudo da tecnologia, que por sua vez possibilita 2. O aperfeiçoamento da tecnologia. É certo que já existem muitas game engines, inclusive em C++, mas o estudo é o piso de todas as descobertas científicas, o que justifica e motiva o desenvolvimento de uma biblioteca de jogos didática. Esta é a nossa proposta: uma camada de orientação a objetos envolvendo a Allegro de uma forma simples e didática.

A engine, também conhecida com SAGA, ou simplesmente SGL, está estruturada em 7 camadas ou pacotes. São eles:

**1 - Sgl**

É o pacote mais geral, e que contém todos os outros. Quando a funcionalidade de uma classe não é específica ou é usada como ferramenta auxiliar em outras classes, ela é colocada nesse pacote.

* 1. **-**  **Sgl:** é o arquivo que contém os includes da Allegro e as definições dos pacotes. Além disso, contém o tipo String, como definido abaixo.

typedef std::string String;

* 1. **– AllegroStarter:** classe responsável por inicializar a Allegro e seus componentes, e também por desalocar os recursos quando o programa é fechado. Uma exceção é lançada caso algum dispositivo apresente problemas durante a inicialização. Também contém informações sobre a atual versão da Allegro. É baseada no padrão Singleton, e portanto existe somente uma instância da classe.
  2. **– Sgl\_exception:** é a classe gerenciadora das exceções que possam ocorrer durante a execução do programa. É uma especialização de std::exception.
  3. **– Color:** possui recursos de cores, que podem ser usados tanto para colorir a tela como para alterar a cor de uma determinada fonte de texto. Ela aceita dois construtores. Com o primeiro deles é possível definir cores no formato RGB. Para isso, o construtor recebe três parâmetros que variam de 0 a 255, um para a cor vermelha, outro para verde e outro para azul, respectivamente. O segundo construtor aceita strings no formato html, e também strings com o nome em inglês de uma cor, desde que ele já esteja pré-definido. Strings mal formatadas podem levar a resultados inesperados. Material de consulta: <http://www.w3.org/TR/2010/PR-css3-color-20101028/#svg-color>

A classe tem, ainda, métodos para conversão de formatos e para mudar a cor de um objeto. Exemplos:

Definição das cores vermelho e azul escuro usando o primeiro construtor.

**Color vermelho(255,0,0);**

**Color azulEsc(0,0,139);**

Definição das cores rosa passando um nome pré-definido e das cores verde escuro e coral no formato html. Note que o símbolo ‘#’ é opcional.

**Color rosa("pink");**

**Color verdeEsc("#006400");**

**Color coral("FF7F50");**

Convertendo formatos.

**String aux;**

**aux = vermelho.getName(); // saída: aux = red**

**aux = azulEsc.toHTML(); // saída: aux = #00008b**

Alterando a cor vermelho para vermelho escuro.

**int r = 139, g = 0, b = 0;**

**vermelho.toRGB(r,g,b);**