Definição de linguagem de programação

A linguagem de programação é essencial no desenvolvimento de jogos. Toda a mecânica do jogo é interligada através de uma linguagem de programação. Podemos utilizar praticamente qualquer linguagem de programação, no entanto a escolha da linguagem adequada tornará a eficiência de desenvolvimento e o desempenho do jogo melhores.

C++

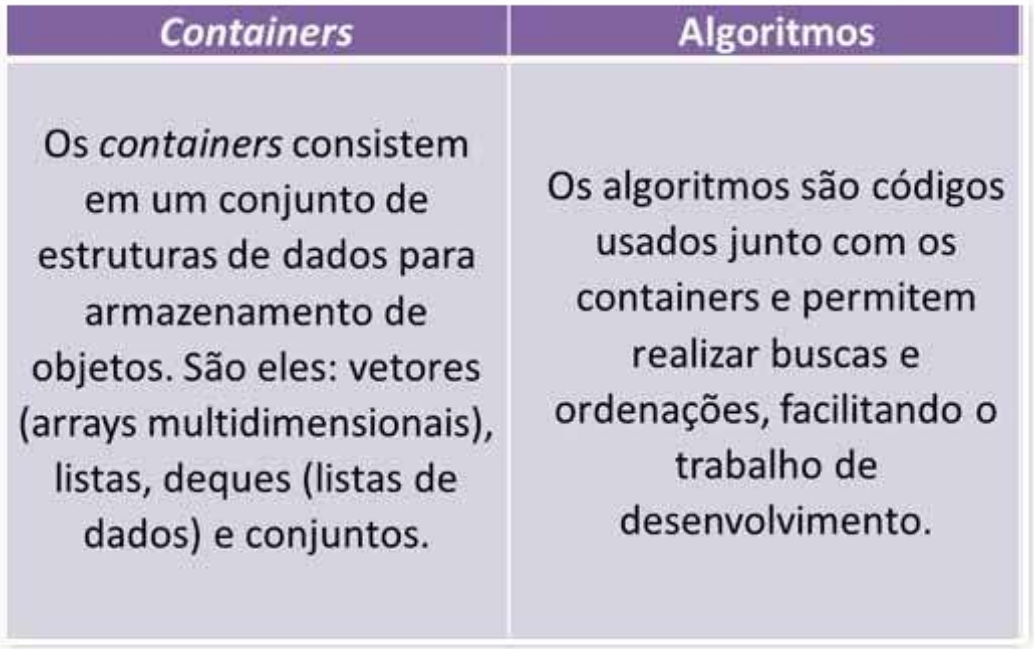
A linguagem de programação C++ é uma das mais utilizadas, atualmente, no desenvolvimento de *games*. Isso decorre da vasta utilização da linguagem C, que surgiu na década de 1980 para o desenvolvimento de jogos.

Quais são as vantagens de utilizar a linguagem C++?
A utilização da linguagem C++ é um fator é o desempenho, é um ponto crítico no mundo dos *games*. É necessário fazer o processamento da cena do jogo a tempo de mostrar um frame (quadro) na tela. Pelo fato de esta linguagem ser de baixo nível, ou seja, ter proximidade com as instruções do sistema operacional e do *hardware*, ela possibilita a programação de gerenciamento da memória e a utilização de instruções simples que são de rápido processamento. A customização de certas soluções é possibilitada por esta linguagem, sendo preferível as linguagens de alto nível neste ponto.

Além das características de baixo nível herdadas da linguagem C, a C++ oferece os recursos para programação orientada a objetos. Esta característica é útil para trabalhar com projetos complexos, pois fornece clareza para organizar a arquitetura da solução e para a compreensão do código. A linguagem C++ fornece isso tudo sem redução de desempenho.

As bibliotecas agregam recursos à linguagem, eliminando esforços de programação. As APIs gráficas Open GL e DirectX também fornecem bibliotecas em C++. Um dos recursos padrão do C++ é a STL Standard Template Library, uma biblioteca que fornece

dois tipos de funcionalidades: os containers e algoritmos.



Containers Os containers consistem em um conjunto de estruturas de dados para armazenamento de objetos. São eles: vetores (arrays multidimensionais), listas, deques (listas de dados) e conjuntos. Algoritmos Os algoritmos são códigos usados junto com os containers e permitem realizar buscas e ordenações, facilitando o trabalho de desenvolvimento.

A linguagem C++ possui, além das bibliotecas-padrão, outras bibliotecas comerciais, como, por exemplo, a Boost, que possui recursos semelhantes ao STL.

A linguagem de programação C++ tem uma padronização definida pela ISO (*International Standard Organization*). Atualmente, está na versão ISSO/IEC 14482:2014, a C++14. Dica: procure pelas mudanças realizadas entre a C++11 e a C++14.

Rabin (2012, p. 189) acrescenta que, apesar de ser uma linguagem muito utilizada, ainda tem que melhorar no quesito desenvolvimento de jogos. A seguir, veremos algumas fraquezas da C++.

Quando utilizamos essa linguagem, temos que nos preocupar com a programação de detalhes que poderiam ser abstraídos caso utilizássemos uma linguagem de alto nível. Por exemplo, temos que cuidar da alocação de memória e dos ponteiros para a memória.

Programando em C++, focamos muito em detalhes que dizem respeito a pontos críticos de programação de jogos. E como isso consiste em apenas 10% do desenvolvimento de jogos, podemos considerar a utilização de outra linguagem para os 90%, em vez do C++. Uma alternativa a isso é associar a C++ à utilização de linguagens de *script*.

A C++ é uma linguagem bastante complicada de trabalhar devido ao seu baixo nível de codificação. Isso é função da sua herança da linguagem C. Além do mais, alguns recursos que o C++ não possui acabam tomando tempo de programação que poderia ser direcionado para o desenvolvimento do jogo. A serialização é um deles. Este mecanismo consiste em o objeto ter noção da sua estrutura para com isso poder guardar seu estado. A serialização nada mais é do que a capacidade de ler os dados de um objeto de alguma mídia. Isso é utilizado para salvar um jogo, não vem nativo no C++, enquanto que outras linguagens já têm este mecanismo nativo.

O C++ é a linguagem utilizada como base da engine Unreal, da Unity 3D, da Blender Game Engine, do Quake Arena III, entre outros. Como um exercício complementar, pesquise as linguagens-base das engines de *games* do mercado.

Para mais detalhes sobre as características do C++, recorra ao livro Introdução ao Desenvolvimento de *games*, de Steve Rabin.

Java

A linguagem Java, atualmente da Oracle, evoluiu do C++ e se tornou uma linguagem de alto nível, independente de plataformas, pois utiliza uma máquina virtual sobre a qual executa. A máquina virtual Java, a JVM (Java Virtual Machine), foi implementada para as diversas plataformas existentes, tornando as aplicações desenvolvidas em Java portáveis. Ela é compilada em bytecode e interpretada na JVM. Mas a linguagem Java não é vastamente utilizada no desenvolvimento de jogos e, mais recentemente, segundo Rabin (2012, p. 191), esta linguagem está pronta para entrar para o mundo dos *games*.

Por que Java? Ela evoluiu da C++, apresentou uma versão mais limpa desta, evitou, por exemplo, os ponteiros do C. Também criou um recurso de manipulação da memória explícita. Java introduziu o mecanismo de serialização de grande utilidade para os jogos. A linguagem possui bibliotecas de entrada e saída, gráficas, de rede baixo nível (*sockets*) e de alto nível (protocolos HTTP e FTP). As APIs OpenGL (gráfico) e Open AL (som) agora possuem bibliotecas de integração com a Java. Assim ocorre a consolidação desta linguagem para o desenvolvimento de jogos.

A linguagem ainda carece de desempenho. Muitos jogos desenvolvidos com ela tinham como foco a web, porém não demandavam um desempenho crítico. Quando se trata de plataformas específicas comerciais, a linguagem teve que passar por algumas adaptações, como a compilação Just In Time, que faz a compilação do código para o formato binário nativo, ou as máquinas HotSpot, que observam os pontos críticos do código e os otimizam em tempo de execução. Outro aperfeiçoamento incluído buscando melhorar a colocação da linguagem no mercado de *games* foi a inclusão da Java Native Interface (JNI), que permite escrever parte do código com C++.

O Java ainda não está disponível para as plataformas consoles. Seu mercado limita-se aos navegadores web e aos portáteis. A vantagem que oferece é não se preocupar com o *hardware*, que fica a cargo da JVM. Ela também é adequada para jogos para PC, no entanto, são lançados poucos títulos desenvolvidos com ela. Rabin (2012) destaca que o Java domina o desenvolvimento para portáteis, está bastante presente em jogos para download, e que logo pode surpreender o mercado de console, assumindo uma posição relevante em termos de desenvolvimento.

Pesquise! Busque na internet informações de jogos desenvolvidos com a linguagem Java, independentemente de plataforma.

Linguagens de *script*

As linguagens de *script* são utilizadas normalmente no desenvolvimento de jogos para codificar eventos, controlar uma animação ou toda lógica e comportamento do jogo. São linguagens de alto nível.

Por que linguagens de *script*? Utilizamos as linguagens de *script* em função da facilidade de desenvolvimento, do tempo para ver o resultado em função de alguma modificação, pela possibilidade de acrescentar recursos em formato de código e por apresentarem recursos úteis para o desenvolvimento de *games*, como a serialização, por exemplo.

Como desvantagens, estas linguagens apresentam um desempenho inferior ao das linguagens de alto nível. Este desempenho inferior ocorre porque estas linguagens são interpretadas em tempo de execução, diferentemente das linguagens compiladas, em que o código é transcrito para binários nativos (códigos de máquina) antes da execução. Algumas linguagens de alto nível apresentam o gerenciamento automático de memória, o que nem sempre é eficiente.

O suporte e as ferramentas disponíveis melhoraram com o aumento da utilização de linguagens de *script* nos motores de jogo mais conhecidos do mercado, as ferramentas de depuração (debugger) de código estão mais desenvolvidas para estas ferramentas. No entanto, para o desenvolvimento “caseiro”, as ferramentas de depuração se tornam mais escassas. Um exemplo é o uso da IDE Eclipse, que fornece este tipo de recurso para a linguagem de *script* que será utilizada, como a linguagem Lua.

Quais são as linguagens de *script* populares? Diversos organismos ligados ao desenvolvimento de sistemas fazem comparações entre as linguagens de programação mais usadas. Temos a Code Eval, a comunidade Biobe, o Github, repositório on-line de código, a Stackoverflow e Redmonk como fontes de comparação da utilização das linguagens de programação. Voltando para o desenvolvimento de *games*, temos duas linguagens de *script* bastante populares: Python e Lua.

O Python é uma linguagem de *script* interpretada e orientada a objetos. Ela é utilizada junto com a Blender Game Engine, por exemplo. É um linguagem que possui uma comunidade de desenvolvimento bastante forte, em função de estar no mercado há um bom tempo.

A Lua é uma linguagem de *script* leve que pode ser usada com orientação a objetos e possui uma boa eficiência em termos de custo de memória. Mas é uma linguagem limitada, pois utiliza apenas um tipo numérico (duplo), o que deixa lenta as operações. Ela é ideal para projetos pequenos e se adequa bem aos *hardware*s limitados. Porém, sua utilização é mais indicada junto com uma linguagem mais potente.

Para saber mais sobre a linguagem de *script* Lua, acesse o site: .

Como escolher uma linguagem de *script*? Para escolher uma linguagem de *script*, você deve avaliar a sua necessidade em função das características que precisa. Outros fatores que influenciarão na sua escolha são o desempenho, as facilidades de depuração, a comunidade de práticas e suporte e as plataformas foco de desenvolvimento.

Por fim, existem ainda outras possiblidades de linguagens de *script*, inclusive linguagens customizadas, como a UnrealScript, ou Blueprint, do motor de jogo Unreal, o motor Quake com a Quake C, o jogo NeverWinter Nights com a NWNScript, entre outros casos. A Unity 3D, por exemplo, trabalha com as linguagens de *script* C#, JavaScript e Boo (variação do Python). São diversas as possibilidades de trabalho, tal qual são

diversas as fontes disponíveis para embasar a escolha da linguagem que você utilizará em seu projeto. Que tal você complementar o seu estudo verificando a bibliografia e buscando por mais sites na internet? Não se esqueça de que conhecer bem as ferramentas disponíveis é essencial para o bom desempenho profissional.

Referências bibliográficas

BOOST. Disponível em: <http://www.boost.org/>. Acesso em: 28 out. 2015.

ISO/IEC 14482:2014: Information technology -- Programming languages -- C++. Disponível em: <http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=64029>. Acesso em: 1 jun. 2016.

RABIN, Steve. **Introdução ao Desenvolvimento de *games*:** Programação – técnica, linguagem e arquitetura. Vol2. São Paulo: Cengage Learning, 2012.

TIOBE Index for October 2015. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acesso em: 28 out. 2015.