Alfredo José de Paula Barbosa - 16890
Michell Stuttgart Faria - 16930
Paulo Vicente Gomes dos Santos - 15993
Orientador: Prof. Dr. Enzo Seraphim

SAGA Game Library
Biblioteca para desenvolvimento de jogos eletrônicos 2D

Itajubá - MG

Alfredo José de Paula Barbosa - 16890
Michell Stuttgart Faria - 16930
Paulo Vicente Gomes dos Santos - 15993
Orientador: Prof. Dr. Enzo Seraphim

SAGA Game Library
Biblioteca para desenvolvimento de jogos eletrônicos 2D

Monografia apresentada para o Trabalho de Diploma do curso de Engenharia da Computação da Universidade Federal de Itajubá.

Universidade Federal de Itajubá - UNIFEI
Instituto de Engenharia de Sistemas e Tecnologias da Informação
Engenharia da Computação

Itajubá - MG

Sumário

Resumo

Uma biblioteca de jogos ou *game engine* pode ser vista como um conjunto 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).

Palavras-chave: *game engine*, jogos eletrônicos, ferramentas de desenvolvimento.

1 Introdução

Desde os primórdios da humanidade a competição é uma forma de diversão muito popular. O objetivo de qualquer competição é testar uma habilidade individual ou grupal e destacar quem ganha. Embora essa característica ainda seja a mesma, a competição está condicionada a uma evolução que pode ser notada, por exemplo, na corrida: no começo ela era individual e só testava a velocidade e a resistência da pessoa; hoje uma corrida automobilística testa a resistência, a destreza, a inteligência e a tecnologia da equipe. Mas essa evolução não se da só na complexidade da competição, ela se manifesta da mesma forma na sua abstração. Os jogos de estratégia que nós conhecemos hoje, por exemplo, são versões abstratas das competições físicas. A aptidão física de cada criatura imaginária é determinada por alguma característica interna do jogo, porque o que o jogo de estratégia testa é o raciocínio e a estratégia da pessoa e não a sua capacidade física propriamente dita. O xadrez internacional, simulando uma guerra da Idade Média, é um caso concreto dessa evolução. Com o avanço da microeletrônica e da computação, no entanto, o jogo de estratégia ganha uma plataforma que pode simular não só um sistema de lógica, mas toda uma realidade virtual. Qualquer jogo pode ganhar uma versão eletrônica. O jogo eletrônico, portanto, não é só uma brincadeira de criança, ele é na verdade o último estágio de um passatempo milenar. Isso se confirma pela movimentação de recursos e ganhos da indústria dos jogos eletrônicos desta geração.

1.1 Motivação

Nos últimos anos, o mercado de games do Brasil tem presenciado um crescimento significativo. [**?**] O advento dos dispositivos móveis, como *smartphones* e *tablets*, e a possibilidade de comercializar seu produto *online* em lojas virtuais e assim reduzir custos favoreceu, em parte, a redução da pirataria e fez com que o usuário preferisse a compra do produto original a investir em um produto não-original. Essa mudança de comportamento por parte do consumidor fez com que um mercado, que antes era visto como inseguro, passasse a ser considerado um mercado promissor pelas empresas desenvolvedoras de software, incluindo as desenvolvedoras de *games*.

Com o crescimento da área de desenvolvimento de jogos eletrônicos, surge também a necessidade de encontrar mão-de-obra capacitada, necessidade esta que é uma das maiores reclamações das indústrias de desenvolvimento de *games* do Brasil. Por se tratar de uma área de desenvolvimento recente, é difícil encontrar profissionais capacitados na área. Uma das soluções mais simples para esta carência de mão-de-obra é incentivar estudantes, sejam eles de nível técnico ou universitário, a aprender sobre as ferramentas e técnicas mais utilizadas no desenvolvimento de um jogo eletrônico. Assim, torna-se de suma importância a implementação de ferramentas que facilitem o primeiro contato do estudante com essa complexa área de desenvolvimento, motivo este que nos motivou a criaçãoo da *SAGA Game Library*.

1.2 Objetivos

A *SAGA Game Library* foi desenvolvida tendo em foco o meio acadêmico. Com o aumento do mercado de desenvolvimento de jogos eletrônicos no país, surge a necessidade de investir na capacitação de profissionais para atender a essa demanda. Não apenas profissionais do setor precisam estar em constante atualização, mas os

agora estudantes e futuros profissionais também precisam de capacitação. É para esse último que é direcionada esta biblioteca de desenvolvimento. Seu objetivo primário é possibilitar ao usuário, seja ele um estudante ou entusiasta, o primeiro contato com o mundo do desenvolvimento de jogos.

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. Simplicidade, eficiência e aprendizado são as palavras-chave da *SAGA Game Library*.

2 Metodologia

Uma vez que a proposta inicial do projeto foi decidida, a pra etapa do trabalho consistiu na defini de como a biblioteca seria desenvolvida. Com isso em mente, comeos pelos pontos mais bcos: a linguagem de programa e a API a ser utilizada para acesso ao hardware.

2.1 Linguagem de Programa

Atda de 90 cada jogo tinha a sua *engine*, feita para possibilitar a maior eficiia no uso da mem e da unidade de processamento possl, de acordo com as exigias de cada jogo. Um jogo que sva formas geomicas, por exemplo, nrecisava tratar imagens na sua *engine*. O nl da microeletrnica e da computa jssibilita o uso de *engines* gencas, mas o desenvolvimento de uma ainda demanda uma programa muito pra da mina.

or isso que a escolha da linguagem de programa precisa ser feita com cuidado. Considerando o conhecimento da equipe e o propo do projeto, que era uma *engine* didca, para influenciar o designe e o desenvolvimento de jogos de acordo com o nosso alcance, as linguagens de programa selecionadas para a anse foram Actionscript, C++, C# e Java, de acordo com a simplicidade, o poder e a compatibilidade de cada uma.

2.1.1 Estudo Comparativo

O Actionscript a linguagem orientada a objetos desenvolvida pela Macromedia. O que no ino era uma ferramenta para controlar animas se tornou uma linguagem de script tomplexa que podia ser usada no desenvolvimento de um jogo. Embora essa linguagem ainda seja muito usada no desenvolvimento de jogos de web, o que provocou a decisontra a ela foi a expectativa de que o HTML 5 viesse a incorporar o Javascript e, dessa forma, modificar ou inutilizar o Actionscript.

O C# (C Sharp) a linguagem multi-paradigma da Microsoft feita para o desenvolvimento de sistemas pros para a plataforma .NET. C# e Java compartilham a mesma simplicidade na leitura e na codifica, assim como a mesma forma de interpreta e compila, mas um programa em C# estis pro da mina do que um programa em Java. A escolha parecia feita quando ntendemos que a liga do C# com a Microsoft poderia custar a compatibilidade do nosso jogo.

Java a linguagem orientada a objetos desenvolvida pela Sun, hoje possu pela Oracle. A sua fama de espaa e pesada n coerente com a realidade: hoje a linguagem conta com a Compila na Hora ou Just in Time Compilation (JIT Compilation ou s), para que a sua execu neja mais interpretada. Mas a sua principal caracterica ompatibilidade: a Mina Virtual do Java ou Java Virtual Machine (JVM) a plataforma virtual que pode ser feita compatl para qualquer plataforma fca.

Por mais evolu que seja a JVM, no entanto, o Java ndmite o acesso quina necesso para o desenvolvimento de uma Game Engine, senom o uso do C++, por meio da Interface Nativa do Java ou Java Native Interface (JNI). Em outras palavras, para usar o Java, nesse caso, nros que usar o C++. Esta, por sua vez, n a mais simples na codifica, mas nem limita alguma tanto em termos de compatibilidade quanto em termos de acesso quina.

2.1.2 A Escolha: C++

C++ (C Mais Mais ou C Plus Plus) [Perucia 2007] a linguagem de programa multi-paradigma, com suporte para a programa imperativa e a programa orientada a objetos, de uso geral, desenvolvida por Bjarne Stroustrup, para formar uma camada de orienta a objetos sobre a linguagem de programa C. O C++ possibilita a programa de baixo nl assim como a programa de alto nl e por isso nsiderado uma linguagem de programa de nl mo em termos de proximidade da mina.

Na verse 2011, com uma a e construtiva influia da Biblioteca Boost, o C++ ganha uma se de caractericas, entre as quais podemos citar: verifica de tipo dinca, estruturas de convers controle de alto nl, reflexadronizada, uma biblioteca de computa paralela padronizada, coleta de lixo automca, etc. Com essas mudan o C++ ganha em simplicidade, sem, contudo, perder a proximidade da mina de outrora.

2.2 Allegro

Nas nossas pesquisas para escolher uma biblioteca com a qual trabalhar, duas se destacaram: a Allegro e a SDL. A SDL (Simple Direct Media Layer - Camada de Ma Direta Simples) a biblioteca multima simples de usar, multiplataforma, de co aberto, e amplamente usada para fazer jogos. Ela poderia tambtender as nossas necessidades, mas a Allegro se destacou por ter um co mais limpo e intuitivo, e rotinas especcas para o desenvolvimento de jogos, como renderiza acelerada por hardware e suporte nativos a diversos formatos de imagens e arquivos de io. Por esta razla foi escolhida.

Allegro [http://alleg.sourceforge.net/docs.html] a biblioteca grca multiplataforma, de co fonte aberto e feita na sua maioria em C, mas utilizando internamente tambssembly e C++. Seu nome acrnimo recursivo que representa "*Allegro Low Level Game Routines*" ("Rotinas de jogo de baixo nl Allegro"). Funciona em diversos compiladores e possui rotinas para a manipula de funs multima de um computa-

dor, ale oferecer um ambiente ideal para o desenvolvimento de jogos, tornando-se uma das mais populares ferramentas para esse fim atualmente. Originalmente desenvolvida por Shawn Hargreaves, ela se tornou um projeto colaborativo, com colaboradores de todo o mundo.

Ela possui funs para jogos 2D e 3D, mas n indicada para o ltimo caso. Apesar de ner suficiente para o completo desenvolvimento de um jogo, existem pequenas bibliotecas adicionais (add-ons), feitas para serem acopladas legro, permitindo assim a sua extens Atravesses add-ons ssl, por exemplo, obter suporte a arquivos MP3, GIF, imagens JPG e vos AVI.

Internamente, a biblioteca vida em blocos. Isso il para que o usuo nenha que colocar uma por de funs que nsa na hora de distribuir seu jogo, incluindo somente as partes que for utilizar, diminuindo consideravelmente o tamanho do mesmo.

Atualmente a biblioteca se encontra na sua quinta vers Allegro 5 foi completamente reescrita de suas verses anteriores. Foi feito um esforara tornar a API mais consistente e segura, o que trouxe melhorias funcionais e uma grande mudana sua arquitetura, sendo agora orientada a eventos. Entretanto n compatl com suas verses antigas.

A Allegro 5.0 suporta as seguintes plataformas:

- Windows (MSVC, MinGW);
- Unix/Linux;
- MacOS X;
- iPhone;
- Android (Suporte provido pela Allegro 5.1, que ainda se encontra instl).

2.2.1 Principais Recursos e Funs

A seguir, encontramos um conjunto dos principais recursos da Allegro 5, comedo com as funs mais gerais da biblioteca.

- **al_init()**: Inicializa a biblioteca Allegro, dando valores a algumas variis globais e reservando mem. Deve ser a primeira fun a ser chamada.
- **al_exit()**: Encerra a Allegro. Isto inclui retornar ao modo texto e remover qualquer rotina que tenha sido instalada. Ncessidade de chamar essa fun explicitamente, pois, normalmente, isto ito quando o programa termina.

As rotinas de video:

- **ALLEGRO_DISPLAY**: Tipo que representa a janela principal. A biblioteca permite que se trabalhe com mltiplas janelas.
- **al_create_display(width, height)**: Cria uma instia da janela, retornando um ponteiro para ALLEGRO_DISPLAY. Os partros indicam as dimenses em pixels.
- **al_flip_display()**: Fun para atualizar a tela.
- **al_destroy_display(var)**: Finaliza a instia *var* do tipo ALLEGRO_DISPLAY $\times$.

As rotinas para manipula de arquivos de imagem:

- **ALLEGRO_BITMAP**: Tipo que representa o arquivo de imagem carregado pela Allegro.
- **al_init_image_addon()**: Inicializa o add-on da Allegro 5 para utiliza de imagens.

- **al_load_bitmap("example.jpg")**: Carrega a imagem indicando no partro o nome e tipo. Ela deve estar previamente salva na pasta do programa. Recebe o caminho relativo ou absoluto da imagem a ser carregada, retornando um ponteiro para o tipo ALLEGRO_BITMAP.

- **al_draw_bitmap(bitmap, x, y, mirror)**: Fun para desenhar a imagem na tela. Os partros s bitmap a ser desenhado, as posis x e y e as flags de espelhamento (0, ALLEGRO_FLIP_HORIZONTAL, ALLEGRO_FLIP_VERTICAL).

As rotinas de io:

- **ALLEGRO_SAMPLE**: Tipo que representa arquivos pequenos, geralmente efeitos sonoros.

- **ALLEGRO_AUDIO_STREAM**: Tipo para representar arquivos grandes, de forma que o arquivo n carregado de uma vez para a mem. Geralmente representa os arquivos que irompor uma trilha sonora.

- **al_install_audio()** e **al_init_acodec_addon()**: A primeira inicializa as funs relativas ao io. A segunda inicializa os codecs necessos para carregar os diversos formatos de arquivo suportados. Fornece suporte a alguns formatos, como Ogg, Flac e Wave.

- **al_set_audio_stream_playing(musica, true)**: Fun que recebe o arquivo de io jrregado no primeiro partro e um tipo booleano no segundo (true para faza tocar ou false, em caso contro).

- **al_destroy_audio_stream(musica)** e **al_destroy_sample(sample)**: Funs de desaloca dos arquivo de io carregados pela Allegro.

A Allegro ainda possui muitos recursos que noram citados por fugirem do escopo deste caplo. Posteriormente, nemos explorar mais a fundo os recursos que a Allegro oferece.

2.3 O software Tiled

A grande maioria dos jogos eletrnicos 2D apresenta, alos *sprites* dos personagens e itens, uma imagem representando o ceno do jogo. Dependendo da natureza do jogo, esses cenos podem possuir grandes dimenses, o que torna custoso para o software do jogo carregar e armazenar essa imagem em mem (RAM e em disco). Mas, se analisarmos a imagem que representa o ceno, vamos verificar que o mesmo rmado por pequenas partes que se repetem com muita frequia. Assim, aproveitando dessa caracterica e com o objetivo de diminuir o consumo de mem e o desempenho ao carregar imagens de resolu elevada, foi desenvolvida uma tica conhecida como *TileMap*. A tica consiste no uso de uma imagem, chamada *tileset*, contendo pequenos peda de imagens, conhecidos como *tiles*, que ss imagens que se repetem em grande quantidade na imagem do ceno de jogo. Estes *tiles* ssados para criar uma imagem composta denominada *tiled layer*.

O ceno final do jogo pode ser constitu de um nico *tiled layer* ou ser resultante da combina de dois ou mais *tiled layers*. Atrava tica de *Tilemap*, torna-se possl construir inmeros cenos, com variadas dimenses, usando como base o mesmo *tilesets*, aumentando a economia de mem e neduzindo o desempenho no carregamento de imagens.

O software Tiled [http://www.mapeditor.org/ 2013] a ferramenta gratuita desenvolvida em C++ para a cria de layouts e mapas usando *tilesets* baseado na tica de *Tilemap*. De simples manuseio e grande versatilidade, o Tiled faz a edi de vas camadas de *tiles* e salva tudo em um formato padronizado de extens‘.tmx”. Uma das principais vantagens do formato TMX a organiza, detalhamento e praticidade, sendo que seu contedo pode ser lido atravo uso de um *parser* para arquivos XML.

O Tiled editor de nis que suporta mapas com projes ortogonais e isomicas e ainda permite que objetos personalizados sejam salvos como imagens na resolu que desejar. Tem suporte tamb comandos externos, *plugins* e formatos usados por

outros editores. ompatl com diversas *engines* de cria de jogos e fornece meios de comprimir seus dados de modo a diminuir o tamanho em disco do arquivo TMX. ossl, ainda, redimensionar e alterar o mapa posteriormente, criar mltiplos mapas em uma nica sess ainda salvar ou restaurar atve vezes. Com ele pode-se especificar o tamanho de cada *tile* em um *tileset*, ou criar um mapa sem tamanho estrito sobre as imagens [Brunner 2012]. Mesmo que o desenvolvedor nueira que seu jogo seja baseado em *tiles*, o software nda uma excelente escolha como um editor de nis. Pode-se uso tambas entidades invisis, tais como as de colis o aparecimento de objetos dentro do mapa. Por sua simplicidade ela pode ser usada por programadores iniciantes ou experientes.

O processo de cria de um mapa com o Tiled ito basicamente usando os passos abaixo:

1. Escolher o tamanho do mapa e tamanho do *tile* base;
2. Adicionar *tilesets* vindos de imagens;
3. Adicionar quaisquer objetos que representem algo abstrato;
4. Salvar o mapa no format TMX;
5. Importar o arquivo TMX e interpreto para o jogo.

O Tiled talmente gratuito. Esse detalhe aliado a facilidade de uso e versatilidade, tornaram-no extremamente popular em meio munidade de desenvolvedores de jogos, onde nudantes e entusiastas, mas tambmpresas e profissionais da a, passaram a adoto como editor de nis padr Seguindo esse pensamento, o nosso *framework* a ser desenvolvido tambossui suporte nativo aos nis construs usando esse software. Um dos recursos mais interessantes do Tiled ossibilita de exportar os dados contidos no arquivo .tmx de forma compactada e codifica. A principal vantagem do uso dos algoritmos compacta e codifica edu do tamanho em disco do arquivo .tmx resultante e aumento da velocidade de carregamento do ceno, uma vez que a bibli-

oteca carrega os dados codificados e/ou compactados e os decodifica/descompacta a nl de software ao inve realizar a leitura dos mesmo em disco.

A compacta de dados fornecida pelo Tiled alizada pelas bibliotecas ZLIB e GZIP, enquanto a codifica alizada pelo algoritmo Base64. Sornecidas as ops de exportar os dados no arquivo .tmx na forma codificada e compactada ou apenas na forma codificada. Tambferecidas as ops de exportar os dados em formato puro (sem codifica) XML ou no formato CVS.

A seguir podemos verificar a redu do tamanho em disco de um arquivo .tmx em rela ao seu tamanho em disco sem codifica/compacta.

- Arquivo XML puro(sem codifica/compacta): 31,3 KB;
- Com codifica Base64: 9,6 KB;
- Com codifica Base64 + GZIP: 2,2 KB;
- Com codifica Base64 + ZLIB: 2,1 KB;
- No formato CVS: 4,9 KB.

A *SAGA Game Library* prove suporte total e otimizado as 5 ops de exporta acima. Ela, assim com o Tiled, tambaz uso das bibliotecas GZIP e ZLIB para descompacta e tambealiza a decodifica dos dados em Base64 contidos no arquivo .tmx.

2.4 A biblioteca TinyXML

ma biblioteca escrita em C++ que analisa uma sequia de entrada no formato XML e cria uma estrutura independente de plataforma ou linguagem. Em outras palavras, ela realiza o *parser* de uma arquivo .xml e armazena a informa em objetos C++ que podem ser manipulados livremente.

A TinyXML [Rodrigues 2010] pode ser facilmente integrada em outros programas, bastando apenas adicionar seus arquivos ao projeto. Com ela ssl realizar

o acesso aos dados direta ou iterativamente, altera da estrutura atrave inser e remo de elementos, remo de espa duplicados e a grava para ficheiros em formato XML. TinyXML a estrutura extremamente compacta e robusta, elaborada para um rdo e fl aprendizado. Pode ser usada para fins de co aberto ou comerciais. Ela mpatl com UTF-8, de modo a permitir que arquivos XML sejam manipulados em qualquer linguagem.

No *framework* que sersenvolvido, a TinyXML sersponsl por realizar o *parser* do arquivo .tmx gerado pelo software Tiled.

Figura 2.1: *Exemplo de tileset.*

Figura 2.2: *Exemplo de um ceno constru com tileset.*

Figura 2.3: *Interface do software Tiled.*

3 Desenvolvimento

O desenvolvimento da *SAGA Game Library* ou SGL, assim como todo *software*, passou por diversas etapas para que no final torna-se possível obter um produto condizente com a proposta do trabalho.

O desenvolvimento de um *software*, de maneira geral, sempre é composto das seguintes etapas:

- Especificação dos requisitos do *software*: Descrição do objetivo e do se espera do *software*.
- Projeto do sistema: decisão dos conceitos relacionado ao que deve ser implementado, incluindo a escolha da linguagem de programação adequada, sistema operacional alvo, bibliotecas e ferramentas auxiliares.
- Implementação: O próprio desenvolvimento do *software*. Consiste na transformação de todo conteúdo formulado na fase de projeto em código.
- Teste e depuração: Fase que consiste no teste do *software* já implementado e procura por erros e correção destes.
- Documentação: A fase final do desenvolvimento consiste em documentar o *software* criado, incluindo manuais de uso da biblioteca.

A SGL também seguiu de maneira consiste as etapas acima e a seguir serão descritas as particularidades de cada uma delas.

Referências Bibliográficas

BRUNNER, N. Introduction to tiled map editor: A great, platform-agnostic tool for making level maps. 2012.

Allegro - a game programming library docs.

Editor de mapas: Tiled. 2013.

PERUCIA, A. S. *Desenvolvimento de Jogos Eletronicos - Teoria e Pratica*. [S.l.]: Novatec Editora., 2007.

RODRIGUES, R. Notas sobre tinyxml. 2010.