Desenvolvendo jogos com o engine Cadabra 3D

William Mallouk

Abstract. *The audience of this short course will learn how to create games and demos in few minutes using the Cadabra 3D Game Engine, which is made in Brazil. Topics covered include engine architecture, how to develop assets, how to program a simple demo with the engine, and the basics on programming the character animation system.*

Resumo. *Os espectadores deste mini curso irão aprender como criar jogos e demos em poucos minutos usando tecnologia inteiramente desenvolvida no Brasil. Assuntos abrangem arquitetura do engine, como desenvolver assets (modelos e animações), como programar uma demo simples e conceitos básicos do sistema de animação de personagens.*

1 Objetivo

Este mini curso tem o objetivo de ensinar como criar conteúdo e programar jogos utilizando o engine Cadabra 3D. Ao termino do curso, que durará aproximadamente duas horas, os ouvintes saberão como desenvolver demos e jogos simples utilizando o engine Cadabra 3D.

2 Dados Gerais

Seguem os dados gerais sobre o mini curso:

Tratamento dado ao tema: prático.
Nível do mini curso: básico.
Duração do mini curso: aproximadamente 2 horas.
Infra-estrutura necessária: datashow (projetor).

3 Introdução

Bem vindos ao mini curso sobre desenvolvimento de jogos com o engine Cadabra 3D. Aqui iremos tratar sobre tópicos como a origem do engine Cadabra 3D, sua arquitetura e suas features. Iremos apresentar algumas demos contidas no Cadabra SDK, demonstrar o Pipeline de Arte. Teremos também uma demonstração de programação no engine, onde criaremos uma demo em cinco minutos utilizando o Cadabra. Aprenderemos a programar o sistema de animação de personagens e, finalmente, mostraremos um jogo completo feito com o engine. Para iniciar, vamos entender o termo *engine* no contexto da indústria de jogos.

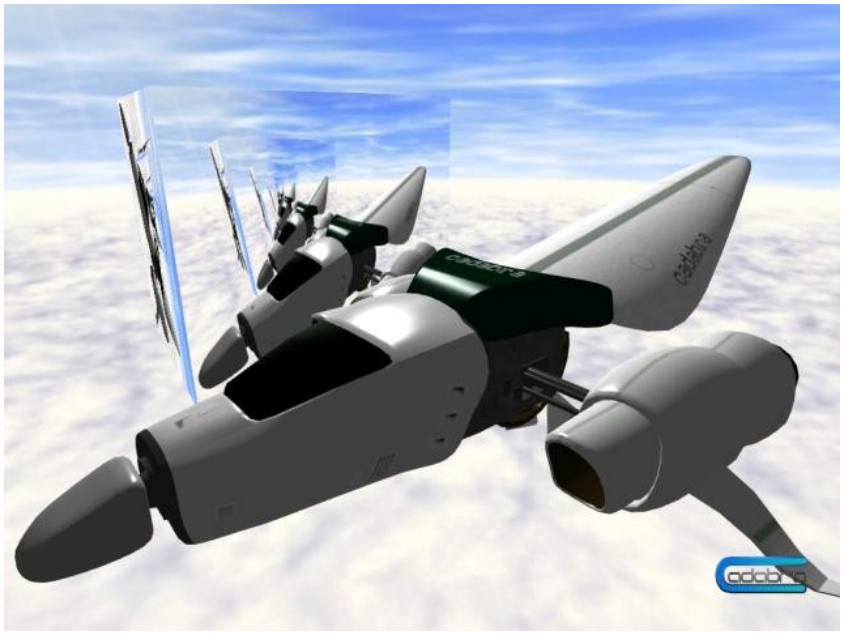
4 Engines para jogos em 3D

Por Júlio Pescuite

Primeiramente, vamos definir o termo *engine*. Muito provavelmente você já ouviu falar neste termo. Vamos tentar entendê-lo. Ir a sua loja favorita de games e ver na prateleira aquele jogo que você aguardava o lançamento há meses, é uma sensação que todos que gostam de games já passaram. Imagine que esse jogo que você tanto esperava, seja um jogo de tiro, como o Counter-Strike. Você atira, anda pelo cenário, sobe em caixas, se esconde atrás de pedras, acerta outros personagens, quando você atira, você ouve um som de disparo, e quanto mais longe alguém atira, você ouve um som mais baixo, dando a real impressão que você está distante do tiroteio. Tudo parece muito real não é mesmo? Pois então anime-se, a tecnologia vem se aprimorando, deixando os jogos mais próximos da realidade a cada dia que passa. Mas, você já se perguntou sobre o que faz aquilo tudo acontecer? Por que seu personagem não atravessa as paredes? Como o jogo sabe que a luz do sol, faz os objetos refletirem luz dependendo de sua posição no cenário? Como que

ao lançar uma granada, dependendo da força que eu a jogo, ela cai em lugares diferentes? Se cair num plano mais inclinado, ela terá um comportamento diferente se caísse numa superfície plana.

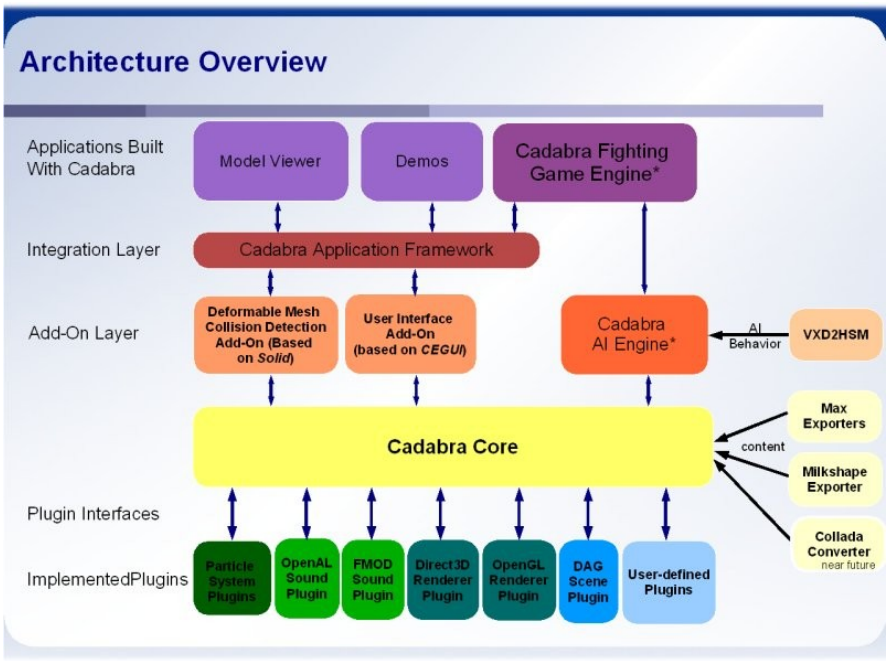
Pois é, você só acabou de descobrir que existe algo dentro do jogo que faz tudo isso, e isso se chama *engine* ou *motor* em português. Existem diversos tipos de engines. Um engine pode dar suporte gráfico ao jogo, bem como ao mesmo tempo, controlar a física e os efeitos de som. Com o avanço da tecnologia, os engines da atualidade são utilizadas para a renderização dos gráficos e outra para os cálculos da física. Um engine hoje, dependendo do seu potencial e marca, pode valer mais de U$500.000,00.



O Cadabra 3D é um *engine* com licença LGPL. Isso significa que você pode construir jogos com o Cadabra sem ter que publicar o código-fonte. A principal restrição da LGPL é que apenas modificações no código da biblioteca, ou seja, do Cadabra, devem ser publicadas. O Cadabra está em fase de desenvolvimento, porém ele já contém funcionalidades mais do que suficientes para desenvolver jogos com gráficos excelentes.

5 Arquitetura do Cadabra 3D

O Cadabra 3D tem o objetivo de ser um engine extremamente genérico, ou seja, que serve para qualquer tipo de jogo. A sua arquitetura foi desenhada de forma a permitir a extensão de funcionalidades através de *plugins* e *add-ons*, o que torna o Cadabra próprio para uso em qualquer tipo de cena, sejam cenas *indoor* ou *outdoor,* bastando que exista um plugin de gerenciamento de cena específico. Abaixo segue um diagrama contendo uma visão de alto nível da arquitetura do Cadabra 3D.

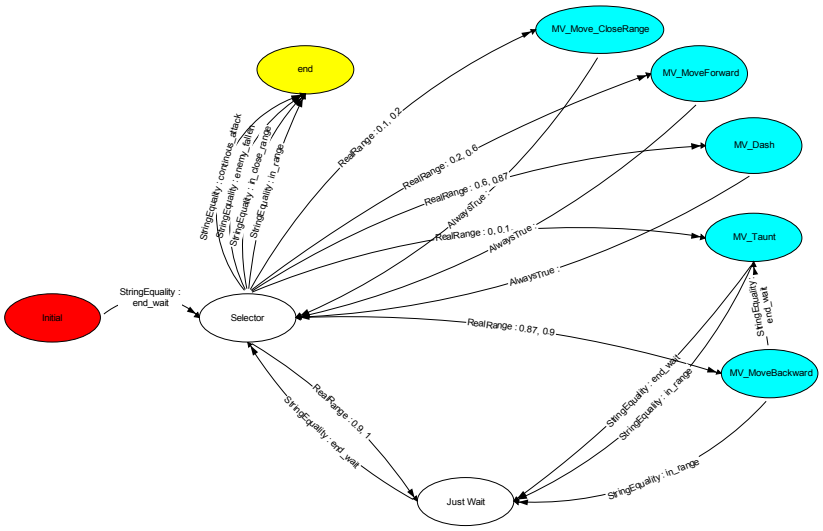


Vamos iniciar a nossa discussão falando sobre o *Cadabra Core*. O core, ou núcleo, é responsável pela coordenação das atividades de todos os módulos existentes em uma aplicação construída com o Cadabra. Ele contém a maior parte dos algorítimos de síntese e processamento de imagens, rotinas de captura e processamento de eventos gerados pelo usuário, entrada e saída de dados, e muitas outras funcionalidades.

A camada mais baixa contém os *plugins*, Os plugins são a principal maneira de estender as funcionalidades do Cadabra Core. Além disso, os plugins complementam o core, de modo que é necessário um conjunto mínimo de plugins para que o sistema funcione corretamente. Atualmente o Cadabra contém *slots* para plugins das seguintes categorias: renderização de som em 3D, renderização de gráficos em 2D e 3D, gerenciamento de cenas e sistemas de partículas.

Logo acima do core, existe a camada de *add-ons*. Add-ons são módulos que também complementam as funcionalidades do core. Os add-ons diferem dos plugins no que as suas funcionalidades podem ser acessadas diretamente pela aplicação do usuário, enquanto as funcionalidades dos plugins podem ser acessadas apenas indiretamente através do core.

Temos três add-ons disponíveis para o Cadabra atualmente. O primeiro é o sistema de detecção de colisão para malhas de polígonos deformáveis. O segundo é o sistema de interface gráfica. O terceiro e último é o Cadabra AI Engine, que permite que o comportamento e inteligência artificial de agentes de jogos sejam definidos de forma inteiramente visual, através de diagramas criados com o Microsoft Visio. O Cadabra AI Engine utiliza uma técnica baseada em máquinas de estados hierárquicas. Abaixo, segue um exemplo de um diagrama feito utilizando o Cadabra AI engine.



Utilizando da tecnologia disponível na Cadabra AI Engine é possível que a inteligência artificial de personagens seja programada de maneira totalmente visual.

As camadas superiores contém aplicações feitas com o Cadabra. São elas as demos disponíveis na SDK, o Cadabra Fighting Game Engine, o Model Viewer (visualizador de modelos) e aplicações criadas pelos usuários do Cadabra.

6 Features do Cadabra 3D

Vamos listar abaixo as principais funcionalidades disponíveis no Cadabra 3D. É possível que estas funcionalidades sejam utilizadas por aplicações construídas utilizando o Cadabra sem que seja necessário o entendimento dos algorítimos envolvidos.

Funcionalidades gerais

- Renderização de alta performance para jogos de última geração
- Arquitetura orientada a plugins. Todas as funcionalidades podem ser estendidas através de plugins criados por usuários.
- Simples de usar. A API Orientada a Objetos foi desenhada para que o desenvolvimento de jogos e aplicações de síntese de gráficos em tempo real seja extremamente fácil
- Renderização com OpenGL ou DirectX
- API bem documentada
- Suporta a grande maioria dos formatos de texturas disponíveis na atualidade (jpg, tga, tif, etc)

Geometria

- Pré-processamento para efeitos de iluminação avançados, como bump/normal/parallax mapping
- Permite ativar e desativar subconjuntos de modelos eficientemente
- Reutilização e customização total de modelos. Um exercito completo pode ser representado a partir de apenas uma cópia matriz do modelo
- Alpha blending utilizando Radix Sort para melhor performance

Engine de Gráficos

- Técnicas de renderização e composição de quadros totalmente customizáveis por usuários. Permite que cada *render pass* seja definido por usuários.
- *Render-to-texture*, modificação do *framebuffer* e fácil definição de efeitos de pós-processamento (ou processamento de imagens)
- Renderização eficiente de imagens em 2D
- Suporta um conjunto de tipos renderizáveis que abrange desde pontos até modelos completos.
- Luzes dinâmicas
- Sombras dinâmicas utilizando shadow maps
- Suporte a vários canais de textura, cada um sendo com coordenadas por vértice ou por face
- Material framework e scripts utilizando a linguagem LUA
- *Cube Mapping* e *Dynamic Cube Mapping*

Sistema de Animação de Personagens

- User-specifiable bones-per-vertex para weighted blending
- Anexar objetos bones (simula agarrar objetos, etc.)
- Exporta dados amigáveis a algorítmos de detecção de colisão
- *Lightweight tokens* permite que os estados de objetos sejam salvos e restaurados com facilidade e eficiência
- Automatic splitting of vertices based on normal and material boundaries;
- Controle explícito do timing das animações
  - Reproduz animações para frente ou para trás, em quaisquer velocidades
  - Facilmente avança o tempo para todos os modelos em uma cena ou somente para aqueles objetos visíveis
  - Ajusta tempos por modelo ou por animação
- Reutilização de dados de animações em múltiplos modelos
- Algorítmo de *skinning* implementado em GPU e CPU
- Interpola arbitrariamente muitas animações simultaneamente

Sistema de Partículas

- Totalmente customizáveis para efeitos especiais como sangue, explosões, fumaça, etc.
- Permite o uso de quaisquer tipo de renderizáveis como partícula (meshes, pontos, billboards, etc)
- Configuráveis através de LUA Scripts
- Extensíveis através de plugins

Outras funcionalidades

- Captura de entrada (mouse/teclado/joystick) de diversas maneiras (assíncronas ou diretas)
- Billboards, Skyboxes
- Light maps e Fog (Névoa)
- Framework de logging framework permite que eventos sejam registrados
- Configuração do engine é feita utilizando LUA Scripts
- Sistema de interface com usuário poderoso, utilizando CEGUI

7 Abra Cadabra!

Esta seção é um pequeno tutorial, que ao invés de chamarmos de “Hello World”, como se faz

quando se está aprendendo a programar em uma nova linguagem de programação, chamamos de *Abra Cadabra*. Este é sem dúvida o lugar onde se deve começar a desenvolver programas usando o Cadabra.

7.1 Primeiro Passo

Faça o download da SDK do Cadabra que está disponível em www.cadabra3d.org se você ainda não o fez. Instale-o.

7.2 Segundo Passo

Dentro do diretório de instalação do Cadabra SDK, existe um diretório chamado "tutorials/tutorial 1". Abra o projeto do visual studio que existe dentro deste diretório. No momento suportamos Visual Studio 6 e 7, e no futuro pretendemos suportar outros compiladores também.

Existem dois arquivos no projeto que foi aberto. São eles o Tutorial 1.cpp e o Tutorial 1.h. As linhas de código destes dois arquivos somam menos que 100. Examine o código atentamente. É relativamente fácil de entender muito do que está acontecendo apenas com a leitura do mesmo.

7.3 Terceiro Passo

Compile e rode o programa. O que você vê? Nada. Isso mesmo, uma tela preta. Isto aconteceu por que a cena está vazia. Pressione *esc* para sair e vamos modificar o código para que algo aconteça. Remova o comentário da seguinte linha:

```
//scene->setSceneBackground(new SkyBox(textureManager, "clouds"));
```

Esta linha instrui o objeto da classe Scene (scene) a utilizar um novo SkyBox que vai ser criado do conjunto de texturas "clouds" como objeto de fundo. A propósito, o conjunto de texturas "clouds" é formado pelos seguintes arquivos encontrados no diretório "tutorial 1/bin/resources": clouds_b.jpg, clouds_d.jpg, clouds_f.jpg, clouds_l.jpg, clouds_r.jpg e clouds_u.jpg.

Agora que você removeu o comentário desta linha, compile e execute o programa. Você deverá ver algo como o que segue:

Mecha um pouco com o mouse para se acostumar com a movimentação da camera utilizando o framework de exemplo que vem com o Cadabra. Pressione esc para continuarmos a criar a nossa demo.

7.4 Último passo

Remova os comentários de todas as linhas que estão comentadas no tutorial 1. As linhas comentadas seguem abaixo.

```
Node* biplaneNode = new Node("Biplane Node");
```

Esta linha cria um novo nó (Node) que pode ser visto como uma estrutura de dados que carrega objetos visíveis pela cena (Scene). No geral, um nó pode carregar quaisquer tipo de objeto renderizável (Renderable) ou outros nós.

```
Model* planeModel = loader->loadModelFile("resources\\biplane.cdm");
biplaneNode->attach(planeModel);
```

Estas linhas carregam um novo modelo (Model) do arquivo biplane.cdm e o anexa ao nó criado acima. Após isso, o nó é anexado à cena, o tornando visível pela camera, como é ilustrado abaixo.

```
scene->attach(biplaneNode);
```

As seguintes linhas ajustam a orientação e a posição do avião.

```
biplaneNode->rotateX(Math::degToRad(90));
biplaneNode->setPosition(0,0,500);
```

E esta linha impede que o avião atravesse o far clipping plane. Para mais detalhes, consulte a documentação da classe Camera.

```
scene->getCurrentCamera()->setFarClippingPlane(3000);
```

E finalmente, as seguintes linhas estão na classe TutorialFrameListener.

```
planeNode->rotateY(evt.getTimeSinceLastFrame() * 0.5f);
planeNode->moveRelative(-Vector3::UNIT_X * 6.85f);
```

Estas linhas fazem exatamente duas coisas: rodam o avião em torno do seu eixo Y e o faz se movimentar para a frente. Compile e rode a demo! Você deve ver o avião voando.

Agora vá siga em frente. Tente melhorar esta demo ou escrever os seus proprios jogos utilizando o Cadabra. Não se esqueça de consultar a documentação do Cadabra API Documentation e os fórum disponíveis em www.cadabra3d.org.

8 Introdução à programação do sistema de animação de personagens

Você gostaria de animar os personagens de seu jogo feito com o Cadabra? Se a resposta for sim, esta é a seção ideal a ser lida. O Cadabra possui um sistema de animação de personagens bastante sofisticado; dentre suas principais características, ele permite que vários canais de animação sejam reproduzidos simultaneamente e que canais de animação sejam agrupados para formar transições suaves entre diferentes estados de animações. Nesta seção, vamos focar nos pontos básicos do uso do sistema de animação de personagens do Cadabra e adquirir o conhecimento que irá permitir que funcionalidades mais avançadas sejam exploradas no futuro. Se você ainda não leu a seção 7, este é o momento certo para que ela seja lida, pois este tutorial é totalmente fundamentado nos conhecimentos obtidos anteriormente.

De agora em diante vamos focar nos seguinte aspectos da programação do sistema de animação de personagens:

1. Carregar um modelo .cdm animável (skinned)
2. Carregar um ou mais objetos de animação (Animation – arquivos .anim)
3. Ligar uma animação a um modelo carregado
4. Dizer ao modelo, através de seu Skeleton como ele deve se posicionar

O último passo pode ser executado de uma variedade de maneiras. Vamos focar na mais simples delas neste tutorial. Poderemos utilizar este conhecimento para utilizarmos blended animations e física em tutoriais futuros.

8.1 Preparação

Vamos iniciar criando uma cópia da pasta “tutorial 1” e a renomeando para “tutorial 2”. Depois disso, vamos remover as linhas de código que não são mais necessárias. A estas alturas você já deve estar familiarizado com o Tutorial 1, então remover as linhas que fazem com que o SkyBox seja exibido não deve ser um grande desafio para você (faça-o antes de continuar). Remova também as linhas que carregam o modelo do avião e as linhas que movimentam o mesmo. Se o

código for compilado neste momento, você deve ver uma tela preta - a mesma que é vista quando você compila e roda o tutorial 1 sem modificações.

Nós vamos precisar também de um modelo animável (skinned) para o nosso programa. Por quê não pegamos este modelo emprestado das demos que acompanham o Cadabra? Copie os seguintes arquivos da pasta "bin/resources/leiha" para a pasta "tutorials/tutorial 2/bin/resources":

```
leiha.cdm
leiha_slap.anim
leiha_me.tga
```

Opcionalmente, copie o arquivo "bin/resources/dx9shaders/HardwareSkinning.fx" para a pasta "tutorials/tutorial 2/bin/resources". Este arquivo somente será utilizado se você desejar utilizar o plugin do DirectX 9.

Se você quiser, você pode visualizar o modelo leiha.cdm com o Cadabra Model Viewer. Você também pode carregar a animação através do menu *Animation* ou apenas arrastar o arquivo .anim para o Model Viewer para ver a animação em ação. Lembre-se que o código-fonte do Model Viewer e de todas as ferramentas do Cadabra estão disponíveis na SDK do Cadabra e servem como exemplo do modo de programar alguns dos subsistemas do engine.

8.2 Inicialização

Agora que temos todos os *assets* necessários em seus devidos lugares, vamos começar a programar. Primeiramente, vamos incluir alguns cabeçalhos que iremos precisar:

```
#include "cadabra/Animation.h"
#include "cadabra/AnimationFileLoader.h"
#include "cadabra/Skeleton.h"
```

Em seguida, vamos fazer algumas modificações no código de inicialização. O nosso método *initialize()* deve estar como segue:

```
void initialize() {
    scene = new Scene(Application::renderer, "BasicSceneGraphPlugin");
    Node* leihaNode = new Node("Pretty Girl");
    Model* leihaModel = loader->loadModelFile("resources/leiha.cdm");
    leihaNode->attach(leihaModel);
    leihaNode->rotateX(Math::degToRad(90));
    leihaNode->setPosition(0,-5,10);
    scene->attach(leihaNode);
    AnimationFileLoader animationLoader;
    Animation* slapAnimation =
            animationLoader.loadAnimation("leiha_slap.anim");
    leihaModel->getSkeleton()->attachAnimation(slapAnimation);
    Application::renderer->addFrameListener(
         new TutorialFrameListener(leihaModel->getSkeleton(),
```

```
                    scene->getCurrentCamera(), userInput, renderer)
    );
}
```

As primeiras seis linhas depois de 'scene = new Scene...' fazem o que aprendemos a fazer na seção anterior deste tutorial, com o avião (se você não se lembra, use o tutorial 1 como referência).

As próximas três linhas executam ações novas. Elas carregam o arquivo .anim que contém a animação slap e então a conectam ao Skeleton do modelo. Fácil, não? Com isso, nosso personagem está pronto para ser animado e é exatamente isso que faremos na sequência.

8.3 Animando o personagem

Vamos modificar a classe TutorialFrameListener para que ela fique desta maneira:

```
class TutorialFrameListener : public ExampleFrameListener {
public:
    Skeleton* skeleton;
    CAReal animationTime;
    TutorialFrameListener(Skeleton* s, Camera* c, UserInput* i,
                          Renderer* r) :
    ExampleFrameListener(c, i, r) {
        skeleton = s;
        animationTime = 0;
    }
    virtual bool onFrameStart(FrameEvent& evt) {
        animationTime += evt.getTimeSinceLastFrame();
        if (animationTime > skeleton->getTotalAnimationTime()) {
            animationTime = 0.0f;
        }
        skeleton->updatePose(animationTime);
        return ExampleFrameListener::onFrameStart(evt);
    }
};
```

O quatro primeiras linhas do método onFrameStart são exatamente as linhas que animam o nosso personagem. Elas incrementam a variável animationTime com o tempo que se passou desde o último quadro. Se o tempo total da animação exceder o tempo das animações anexadas ao esqueleto, nós reiniciamos a animação do tempo t=0. Segue um screenshot do que nós devemos ver ao executar o programa.

Pronto! Agora você conhece os conceitos básicos do uso do sistema de animação de personagens do Cadabra.

8.4 Próximos passos

Existem muitas técnicas envolvidas na programação do [sub] sistema de animação de personagens do Cadabra. Nós tocamos apenas na ponta do iceberg. Recomendamos que a documentação das seguintes classes sejam lidas para um aprofundamento maior no assunto:

* AnimationRange
* AnimationShuffler
* Skeleton::Token
* SkeletalAnimationChannel
* SkeletalAnimationController
* SkeletalAnimationEvent

Novos tutoriais contendo aspectos mais avançados do sistema de animação de personagens deverão ser escritos no futuro. Por enquanto, segue uma lista de recursos que podem ser utilizados para se obter mais conhecimentos sobre este subsistema do Cadabra:

* Model Viewer
* Skeletal Animation Demo
* Hardware Skinning Demo
* Animation Controller Demo

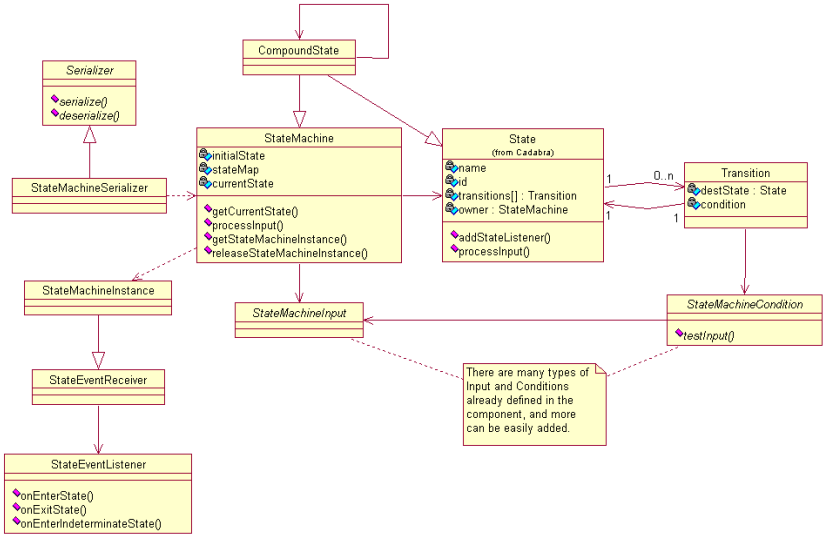
Todos estes recursos estão disponíveis no Cadabra SDK e eles tem o código-fonte incluso na distribuição, todos prontos para serem compilados.

9 Cadabra HSM (AI Engine)

Cadabra HSM é um motor de máquinas de estados um que pode ser facilmente integrado a jogos e outras aplicações. Com o Cadabra HSM você pode projetar máquinas de estados hierárquicas e orientadas a objetos (Object-Oriented Hierarchical State Machines) para vários usos, incluindo

Inteligência Artificial de NPCs (Non-Player Characters). Os diagramas de máquinas de estados podem ser construidos em ferramentas de auto nível como o Microsoft Visio e integrado facilmente no processo de build utilizando ferramentas de conversão, como o vdx2hsm, e o framework de serialização.

O Cadabra HSM é totalmente orientado a objetos. Abaixo segue um diagrama de classes de alto nível descrevendo os principais componentes e seus relacionamentos.



O Cadabra HSM é uma ferramenta muito fácil de se utilizar para adicionar inteligência artificial à jogos e não requer nenhum tipo de programação, seja em C++ ou scripts. Desta forma, não-programadores podem definir o comportamento de personagens no jogo.

Mais detalhes sobre o Cadabra HSM podem ser obtidos na seguinte URL: http://www.cadabra3d.org/api-docs/cadabra_hsm.html

10 Exporters para o 3D Studio Max

O pacote de exporters do 3D Studio Max para o Cadabra é um conjunto de ferramentas feitas para serem utilizadas com as versões 7 e 8 do 3D Studio Max. Estes utilitários são:

* Geometry Exporter
* Animation Exporter
* Cadabra Material
* Quick Export Macro

Nesta seção deste artigo você irá encontrar instruções sobre como utilizar estas ferramentas.

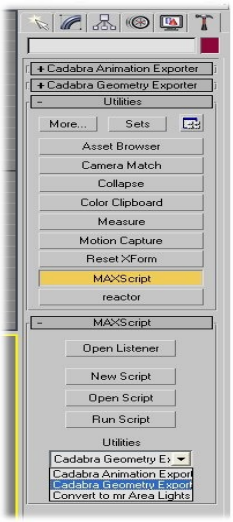
10.1 Instalação

Os exporters do 3D Studio Max para o Cadabra estão inclusos no SDK do Cadabra. Após ter feito o download e instalado o SDK é possível escolher a opção “Install Max Exporters” a partir do menu iniciar do Windows. Veja a imagem abaixo:

Os exporters foram testados nas versões 7 e 8 do 3D Studio Max. Por favor, entre em contato com a Wet Productions se você descobrir que estes utilitários também funcionam em versões anteriores do 3D Studio Max. Depois de ter instalado os exporters, você deve reiniciar o seu 3D Studio Max.

10.2 Exportando Dados

Da aba “tools”, selecione “utilities” e depois “Geometry Exporter” para exportar modelos (arquivos .cdm) ou selecione “Animation Exporter” para exportar animações (arquivos .anim).



10.2.1 Geometry Exporter

O Geometry Exporter exporta arquivos .cdm contendo modelos (malhas de polígonos) no formato do Cadabra. O Geometry Exporter tem várias opções que afetam o modo como a malha de polígonos (mesh) é exportada. Segue uma cópia da tela mencionada seguida pela descrição das opções.

Geometry To Export. Nesta caixa de diálogo devem ser listados (selecionados) os objetos (meshes) que você deseja exportar. Você pode selecionar quaisquer número de objetos: eles serão unificados em um único arquivo .cdm e carregados como um único modelo (Model) no Cadabra.

Texture Channels. Especifique os canais de textura que você deseja exportar nesta caixa. Esta funcionalidade é útil para multitexturing, light maps gerados no Max, etc. Pelo menos um canal de textura deve ser exportado.

Collision Meshes. Especifique nesta caixa os modelos de colisão que serão exportados como malha de polígonos simplificada. Estes meshes serão invisíveis como padrão e não gerarão qualquer overhead de processamento se não utilizados. Estas malhas de polígonos também podem ser úteis para outras funcionalidades, como para lançar sombras com o algorítmo *shadow volume,* já que os meshes de colisão possuem geometria simplificada.

Shadow Volume. Aqui você deve especificar qual geometria você deseja utilizar para lançar sombras com o algorítmo *shadow volume*. Lembre-se que também é possível utilizar *shadow maps* no Cadabra.

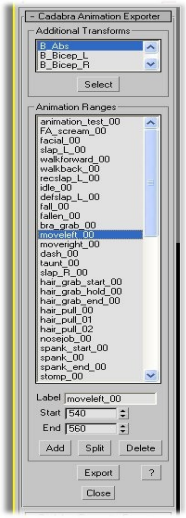
Export Skin. Se você criou um personagem animado utilizando o *Skin Modifier* do Max e você quer animá-lo com o Cadabra você deve escolher a opção *Export Skin*. Com esta opção ativada algumas informações adicionais serão exportadas em conjunto com o modelo, o que permitirá que objetos do tipo Animation sejam anexados aos *Skeletons* dos modelos em tempo de execução.

Generate Tangents. Escolha esta opção se você está utilizando um shader que precisa de Tangentes e Binormais para efetuar o cálculo da iluminação. Bump mapping, parallax mapping, e outras técnicas de iluminação por pixel utilizam estas informações.

Preserve Max Normals. Se você modificou as normais do seu modelo manualmente dentro do 3D Studio Max e escolheu a opção *Generate Tangents,* você deve considerar selecionar esta opção. A opção *Preserve Max Normals* é irrelevante se você não escolheu a opção *Generate Tangents*.

10.2.2 Animation Exporter

Com o Animation Exporter é possível exportar arquivos .anim que contém animações para modelos exportados com a opção *Export Skin* ativada. Abaixo segue uma ilustração da tela do Animation Exporter.

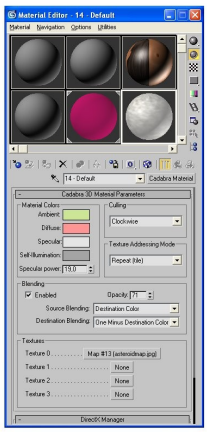


Additional Transforms. Em alguns casos objetos que não são os *Bones* dos personagens geram keyframes na animação. Estes objetos geralmente são *helpers e IK Solvers*. Se a sua animação não está perfeita dentro do Cadabra considere adicionar os helpers e IK Solvers dentro da caixa *Additional Transforms*.

Animation Ranges. Esta caixa permite que você divida sua animação em vários intervalos. Por exemplo, você pode querer determinar o tempo correto para as animações “Andar” e “Pular”. For example, a character may have a "walk" and a "jump" animation. This is pretty straightforward. Each animation range that you define will be available in runtime as an AnimationRange object. See the Skeletal Animation demo source code that is in the SDK for more details.

10.2.3 Cadabra Materials

Mesmo não sendo de uso obrigatório, é recomendado o uso de materiais do Cadabra nos modelos que serão exportados. Os materiais do Cadabra se tornam disponíveis no 3D Studio Max assim que o exporter do Cadabra é instalado.



Com estes material você obtém controle de várias opções como modos de transparência, blending, culling e cores do pipeline fixo. Um material do Cadabra (Cadabra Material) pode ser utilizado como um sub material de um shader do Directx 9, como veremos em seguida.

10.2.4 Direct 9 X Shaders

Se você está utilizando o plugin do DX9 com o Cadabra (Renderer) é possivel configurar e exportar materiais utilizando DX9 Shaders (HLSL) em conjunto com modelos do Cadabra de dentro do Max. Se você tentar renderizar um objeto exportado com um shader do DX9 utilizando o plugin do OpenGL, o mesmo irá utilizar o pipeline fixo, pois shaders HLSL não são suportados no OpenGL. Este é chamado o procedimento de *fall-back* do plugin do OpenGL. Neste caso, recomendamos o uso de *Cadabra Materials*.

10.3 Solucionando Problemas

MaxScript Memory. Se você obter a mensagem de erro 'out of memory' em algum momento enquanto exportando modelos ou animações, recomendamos que você aumente o valor da variável 'MaxScript Memory'. Esta opção está disponível no menu 'Customize->Preferences'. Utilizar 60 megas de heap geralmente elimina todas as mensagens de erro deste tipo. Se você receber esta mensagem novamente, tente aumentar o valor da variável novamente.

11 Licensa

Abaixo segue o texto da licensa open source que é utilizada pelo engine Cadabra 3D.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

* a) The modified work must itself be a software library.

* b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

* c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

* d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the

entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

* a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

* b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

* c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

* d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

* e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library

facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

* a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

* b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Referências

Mallouk, W. e Amerson, M. “Cadabra 3D Engine”
http://www.cadabra3d.org/, Junho de 2006

Pescuite, J. Tutorial Coopergames.
http://br.groups.yahoo.com/group/coopergames/

Free Software Foundation, Lesser General Public Library License
http://www.gnu.org/licenses/lgpl.html

Penn Baillie De Byl
Programming Believable Characters in Computer Games, pp. 234-243.