# Desenvolvimento de um sistema de roteiro para apoio ao processo de Level Design de um RPG educativo

Arivan S. Bastos Leonardo P. Moura Lynn R. G. Alves

Universidade do Estado da Bahia, Departamento de Educação, Salvador – BA – Brasil.


**Figura 1**: Logomarca do jogo Tríade.

**Figura 2**: Imagens da primeira fase do jogo Tríade.

## Resumo

Este documento apresenta o desenvolvimento de um sistema de apoio à implantação do roteiro de uma fase num jogo do tipo RPG. O objetivo deste sistema é facilitar a definição dos acontecimentos durante o gameplay do jogo Tríade, em desenvolvimento pela equipe Arte Interativa.

**Keywords**: Torque Game Engine, roteiro, RPG.

**Authors' contact**:
{arivan,leonardo}@virtualizejogos.com.br
*lynnalves@yahoo.com.br

## 1. Introdução

Um sistema de roteiro pode ser definido como sendo uma rotina de software cujo objetivo é monitorar as ações do jogador que provocam mudanças no ambiente. Geralmente, as ações produzem efeitos nos NPCs (personagens controlados pelo computador), no cenário (com o surgimento de novos itens ou mudança de seus estados), nos diálogos, ou mesmo na apresentação de filmes ou hipertextos de contextualização sobre a história. Esses efeitos podem variar a depender da situação na qual eles estão sendo executados. Em geral, as mudanças diretas provocadas por uma ação no ambiente produzem algumas ramificações, ou seja, efeitos indiretos, que também devem ser monitorados pelo sistema de roteiro. Por exemplo, conversar com o soldado que vigia uma porta importante (efeito direto) resulta em um surgimento do general (efeito indireto). Caso o jogador queira conversar novamente com esse soldado o general não mais deverá aparecer, pois o estado do jogo já foi alterado (efeitos diferentes para uma mesma ação).

Para um jogo de RPG, onde o usuário é inteiramente responsável pelo desenvolvimento das habilidades do seu avatar (FINNEY, 2004), esse sistema pode envolver uma complexidade considerável, visto que a historia de um game desse tipo conta com a presença de diversos personagens, enigmas e narrativas bifurcadas.

Assim, o processo de level design de um game RPG não se resume à construção do cenário (posicionamento dos elementos) e animação dos personagens. É necessário também, adicionar a lógica dos acontecimentos, definindo as conseqüências para cada ação do jogador.

O objetivo do trabalho aqui proposto é desenvolver, sobre o motor Torque Game Engine, um sistema que possibilite ao level designer definir, através de documentos XML, a lógica dos acontecimentos de uma fase em jogos no estilo RPG. Os recursos disponíveis para tal definição serão:

- **Narrativas bifurcadas**: no decorrer do game o jogador poderá trilhar por caminhos distintos, o que acarretará em diferentes experiências da história.

**Mudança de estados do ambiente**: as ações do jogador podem ocasionar mudanças no comportamento de NPCs e nos estados dos itens do cenário.

- **Sistema de eventos**: triggers no jogo podem disparar mudanças no estado do ambiente. Esses eventos podem ser disparados por temporização, localização do jogador ou mudança de estado do mundo.

- **Sistema de diálogos**: o jogador poderá dialogar com os NPCs. Os diálogos disponíveis devem ser escolhidos de acordo com o estado do jogo e dos personagens envolvidos.
- **Passagem de conteúdo**: a interação do jogador com alguns objetos presente no cenário resultará na apresentação de janelas com hipertextos, que descreverão esses objetos observados.

## 2. Metodologia

As seguintes etapas foram seguidas para a realização do trabalho:

1. Estudo do motor Torque Game Engine.
2. Definição das características do jogo.
3. Definição do roteiro do primeiro capítulo do jogo.
4. Levantamento dos requisitos para o sistema de roteiro.
5. Pesquisa e acoplamento de componentes já existentes.
6. Modelagem do sistema.
7. Implementação.
8. Testes e ajustes.

O trabalho encontra-se atualmente na etapa 7. A etapa 8 de testes poderá exigir uma remodelagem do sistema, afim de se realizar os ajustes necessários.

## 3. O sistema proposto

A principal característica do sistema de roteiro é a sua capacidade de gerenciar as diversas possibilidades de narrativa do jogo. O sistema deve permitir a existência de diversas "Quests"[1] dentro de um mesmo cenário, as quais devem ser iniciadas de acordo com as escolhas do jogador durante o game.

Para atender a tal exigência, foi definida uma modelagem XML para a definição das possíveis narrativas do jogo. Cada narrativa é definida dentro de uma tag XML <contexto>, que agrega as definições das ações e eventos para um contexto específico.

As ações são TAGs XML executadas em ordem seqüencial durante a inicialização de um contexto em particular, promovendo as alterações no estado do ambiente. Por exemplo, se durante o game o jogador escolhe desafiar um personagem, haverá uma troca de situação onde , nesse novo contexto o estado do personagem desafiado deverá ser de combate.

Os tipos de ação possíveis de serem executadas foram definidos como sendo:

- **filme:identificador** : resulta no congelamento do estado atual do jogo e na apresentação de um filme em formato .OGG (formato adotado pelo motor utilizado, o Torque Game Engine).
- **estado:objetos=valor** : resulta na mudança do estado dos objetos referenciados para o valor citado.
- **roteiro:identificador**: resulta na mudança do roteiro que está sendo executado.
- **script:identificador**: resulta na execução do script informado (script na linguagem de programação Torque Script).

Os eventos são TAGs XML que definem o que deve mudar a depender de acontecimentos durante o decorrer de um contexto. Por exemplo, se o jogador adquire uma chave, então uma determinada porta deve tornar-se acessível.

Os possíveis tipos de evento são:

- **trigger**: As Triggers de posição são objetos fornecidos pela Engine para detectar a passagem do personagem por um determinado lugar. Assim, é possível disparar uma ação assim que o jogador atingir uma posição pré-determinada dentro do jogo, possibilitando mudanças controladas no ambiente do game.
- **estado** : no decorrer do jogo, um valor pode alterar o estado de um objeto em um determinado contexto. Essa ocorrência produzirá as alterações no ambiente definidas pelo XML.
- **tempo**: Evento disparado depois de um intervalo de tempo definido, que é monitorado com o início de um determinado contexto.

A especificação do núcleo do modelo XML proposto é apresentada no Quadro 1: modelo XML do sistema de roteiro.

Além das tags citadas, foram definidas tags de inicialização, que determinam posição e comportamentos iniciais para personagens, e tags de definição, que determinam as características de parâmetros de outras tags. Por exemplo, na tag <ação filme:identificador>, "identificador" se referenciará a uma tag de definição que determina o caminho e a duração do filme a ser executado.

O modelo XML de diálogos define, para cada estado dos NPCs as possíveis perguntas do jogador e as respectivas respostas. Cada NPC do jogo possui um arquivo XML (em um diretório padrão) que define o diálogo. O formato do arquivo XML de diálogos é apresentado no Quadro 2: modelo XML do subsistema de diálogos.

[1] Quests podem ser entendidas como aventuras paralelas dentro de um RPG, entretanto usaremos a expressão "contextos" com esse mesmo significado.

```
/***
   ------------------------------------
   ESTRUTURA XML DE DEFINIÇÃO DE ROTEIRO
   ------------------------------------
$cons    = $tipoCons:identificador[=valor]
$cond    = objeto=estado

$lstCons = $cons1;$cons2;...;$consN
$lsCond  = $cond1;$cond2;...;$condN

$tipoCons= filme | estado | roteiro | script
$tipoEvt = trigger | estado | tempo

***/

<roteiro>
   <contexto id="Inicial">
   /* Ações */
      [<acao1 consequencias="[$lstCons]"/>
       <acao2 .../>
        ...
       <acaoN .../>]

      /* Eventos */
      [<evento1 tipo=$tipoEvt
        [param=valor_parametro]
        condicoes="[$lsCond]"
        consequencias="[$lstCons]"/>
       <evento2 .../>
        ...
       <eventoN .../>]

   </contexto>

   [  <contexto1 id="Id1">....</contexto>
      <contexto2 id="Id2">....</contexto>
      ...
      <contextoN id="IdN">....</contexto>
   ]
</roteiro>
```

**Quadro 1 -** Modelo XML do sistema de roteiro.

```
/***
   ------------------------------------
   ESTRUTURA XML DE DIÁLOGOS
   ------------------------------------
$cons    = $tipoCons:identificador[=valor]

$lstCons = $cons1;$cons2;...;$consN
$tipoCons= filme | estado | roteiro | script
           dialogo | fimDialogo
***/

<dialogo estado="Id1"
         [consequencias="$lstCons"]>
   <texto></texto>

   /* Possíveis perguntas e respostas
      para escolha do jogador. */
   [<escolhas>
      <opção texto=""
       consequencias="$lstCons">
      [<opção texto=""
       consequencia="$lstCons">
      ...
      <opção texto=""
       conseqüência="$lstCons">]
   </escolhas>]
</dialogo>

[<dialogo estado="Estado1">....</dialogo>
 <dialogo estado="Estado2">....</dialogo>
 ...
 <dialogo estado="Estado2">....</dialogo>]
```

**Quadro 2 -** Modelo XML do subsistema de diálogos.

A tag XML <dialogo> define uma tela de dialogo apresentada no jogo. Essa é responsável por apresentar a fala dos personagens NPCs e as possíveis perguntas/respostas relativas às escolha realizadas pelo jogador.

De forma semelhante ao XML de roteiro da fase, cada escolha do jogador pode disparar um conjunto de conseqüências. Nos diálogos, existem dois tipos de identificadores especiais. O primeiro, denominado de "dialogo", será a identificação do inicio de uma nova conversa. Já o segundo, "fimDialogo", será o responsável pela finalização de um dialogo entre o personagem e o NPC.

## 4. Implementação

Conforme citado, a implementação do sistema foi desenvolvida visando a sua utilização pelo motor Torque Game Engine, da empresa Garage Games (http://www.garagegames.com).

Esse motor apresenta-se projetado sobre uma arquitetura cliente-servidor, que é utilizada tanto para o desenvolvimento de jogos multiplayer como para jogos single player. (FINNEY, 2004)

O lado cliente é utilizado para realizar o controle entre a interface e o jogador, e tem como algumas das responsabilidades, a captação dos dados de entrada e a geração dos dados de saída, objetivando garantir a navegabilidade e interação no ambiente. O lado servidor por sua vez, tem como objetivo a tomada decisão através do processamento dos dados vindos dos jogo (FINNEY, 2004). Logo, de forma simplificada, pode-se dizer que essa arquitetura define um ambiente onde toda a lógica do jogo permanece definida no lado servidor, enquanto que o cliente possui apenas a sistemática de apresentação dessa lógica.

O motor Torque implementa a lógica de ambos os lados através de um script próprio, denominado TorqueScript. Trata-se de uma linguagem orientada à objetos muito flexível, que possui características e capacidades de uma linguagem de programação moderna, e uma sintaxe familiar ao da linguagem C/C++ (FINNEY, 2005). Scripts são frequentemente utilizados para a criação de jogos, pois, segundo (FINNEY, 2005), eles permitem um desenvolvimento *ad hoc* de códigos para realizar tarefas distintas como gerenciar e controlar o motor, ou mesmo formalizar as regras do jogo.

Tendo em vista essa arquitetura, o controle de eventos e ações é realizado no lado servidor do game. O cliente não deve ter acesso a essas informações, esse apenas precisa envia ao servidor as suas ações, e recebe como resposta as conseqüências destas.

O sistema de roteiro foi desenvolvido tendo como base o fluxograma desenvolvido pela equipe de roteiro. Como pode ser visto na figura 3, este mostra graficamente as bifurcações presentes em uma determinada fase, auxiliando na visualização global dos eventos da fase.

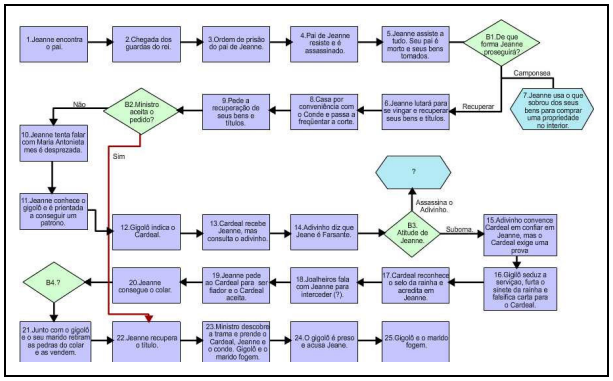

**Figura 3** – Fluxograma da primeira fase

Isoladamente, o sistema de roteiro apenas define a lógica da cena. Entretanto, são necessários outros subsistemas para transformar essa lógica na experiência do jogador. Na implementação adotada neste trabalho, o sistema de roteiro está acoplado ao sistema de inventário, de inteligência artificial, de diálogos, de combate e de triggers:

- **Sistema de inventário**: notifica a entrada e saída de itens no inventário do jogador.
- **Sistema de IA**: informa os estados dos NPCs, que podem ser alterados pelo sistema de roteiro, mudando os seus comportamentos.
- **Sistema de diálogo**: apresenta ao jogador o diálogo selecionado pelo roteiro, de acordo com o estado do personagem.
- **Sistema de combate**: notifica ao sistema de roteiro os eventos decorrentes das batalhas, como por exemplo a morte dos personagens.
- **Sistema de triggers**: notifica ao sistema de roteiro eventos de tempo e de colisão do personagem com triggers no cenário.

Dessa forma, o sistema de roteiro é notificado sobre as mudanças do estado do ambiente, e sua tarefa se resume apenas a executar as ações e monitorar os eventos do contexto atual, executando as conseqüências de um evento que seja disparado.

De uma forma mais técnica, o sistema de roteiro se caracteriza como uma Thread, que faz um parsing do XML de roteiro, monitorando o contexto atual, e trocando o contexto quando necessário. A troca de contexto implica na recriação da estrutura de dados que armazena suas informações, e na execução das ações de inicialização do novo contexto.

## 5. Perspectivas futuras

Novas exigências surgirão com o decorrer do desenvolvimento do jogo. Dentre as já vislumbradas destaca-se a possibilidade de controle dos seguintes eventos pelo sistema de roteiro:

- Mudança de fases
- Mudanças climáticas
- Controle de música ambiente
- Controle de posicionamento de câmera

## 6. Conclusões

A definição do roteiro de um jogo RPG é uma tarefa complexa e exige o desenvolvimento e interação entre diversos subsistemas no motor do jogo.

O Torque Game Engine se mostrou flexível, e ofereceu os recursos necessários para a implantação dos requisitos exigidos para a criação do RPG Tríade. Grande parte dos sistemas auxiliares foram encontrados na própria comunidade do motor, e a adaptação e o acoplamento ao game foram realizados sem dificuldades.

O documento XML criado possibilitou a definição de toda a lógica da primeira fase do game. Testes e ajustes do sistema serão realizados para avaliar os impactos de performance e revelar as falhas do sistema proposto.

## 7. Agradecimentos

Agradecemos a toda a equipe do grupo Arte Interativa por estar determinado a fazer deste um grupo de referência de desenvolvimento de jogos na Bahia, ao nosso financiador: a FINEP, e aos nossos colaboradores: FAPESB e UNEB, por acreditarem no projeto e oferecer o apoio necessário para o desenvolvimento do mesmo.

## Referências

GARAGEGAMES, 2007. *Torque Game Engine*, Disponível em http://www.garagegames.com/products/torque/tge/, acessado em12/08/2007.

FINNEY, K. C., 2004. *3D Game Programming All in One.* Thomson Course Technology, Boston.

FINNEY, K. C., 2005. *Advanced 3d Game Programming All In One.* Thomson Course Technology, Boston.