

Instituto de Matemática e Estatística - USP  
MAC499 - Trabalho de Formatura Supervisionado

# PROJETO NUKENIN

UM JOGO DE AÇÃO PARA O GAMEBOY ADVANCE

10 de novembro de 2005

Edgar Kenji Yamamoto (edgar@linux.ime.usp.br)

Roberto Seiti Yamashiro (seiti@linux.ime.usp.br)

Supervisor: Siang Wun Song



LOADING...



IME - USP

<http://GBAgame.sourceforge.net>

# INTRODUÇÃO

- ◇ Objetivo do projeto: produzir um jogo de plataforma
- ◇ Linguagem **C** com o conjunto de ferramentas do **devkitARM**
- ◇ Áreas da Computação relacionadas:
  - Organização de Computadores,
  - Estrutura de Dados,
  - Engenharia de Software.



# O JOGO

Por que Nukenin?

# 抜忍

- ◇ Jogo de plataforma 2D tendo como personagem principal um ninja
- ◇ Nuke = renegado
- ◇ Nin = ninja
- ◇ Ambientado no Japão feudal (ou quase isso)



# JOGOS DE PLATAFORMA

Muitos de nós já jogaram algum jogo do estilo:

- ◇ Vista lateral
- ◇ Movimentação restrita à um plano
- ◇ Exemplos: Pitfall, Super Mario Bros, Sonic, Mega Man etc.



# HARDWARE: GAMEBOY ADVANCE

O aparelho portátil GameBoy Advance pode ser considerado um pequeno computador, contendo:

- ◇ CPU ARM 32Bits, capaz de utilizar instruções em 16Bits
- ◇ Memória RAM especializada:
  - 256 KB de External Working RAM
  - 32 KB de Internal Working RAM
  - 96 KB de Video RAM
  - mais alguns bytes para paleta de cores, I/O etc.



# DESENVOLVIMENTO

Configuramos um ambiente de desenvolvimento no Eclipse, facilitando bastante a edição do código e realização de testes:

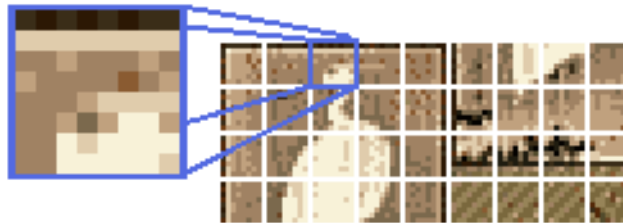
- ◇ linguagem **C**, compilador **GCC**
- ◇ devkitARM (bibliotecas e compiladores)
- ◇ Eclipse com o plugin CDT (necessário para editar código **C/C++**)
- ◇ SourceForge (repositório de nosso código fonte)



# MECANISMO GRÁFICO

O sistema é primariamente 2D, baseado em tiles e sprites:

◇ **Tile** é um pequeno bloco, no caso medindo  $8 \times 8$  pixels



◇ **Sprite** é uma pequena imagem manipulável em termos de posição, rotação etc.



# MECANISMO GRÁFICO

Conseguimos animar os personagens modificando os sprites que o representam, substituindo, na memória, cada tile que constitui o sprite.



Para rotacionarmos um sprite, devemos criar uma matriz de transformação linear do tipo:

$$P = \begin{bmatrix} \cos(\alpha) & -\text{sen}(\alpha) \\ \text{sen}(\alpha) & \cos(\alpha) \end{bmatrix} = \begin{bmatrix} p_a & p_b \\ p_c & p_d \end{bmatrix}$$

E colocarmos os  $p_a$ ,  $p_b$ ,  $p_c$  e  $p_d$  em áreas específicas da memória, relacionando-os à algum sprite.



# BAIXANDO O NÍVEL

Escovar bits e escrever macros, necessidade do projeto.

```
#define RGB(r,g,b)    (r+(g<<5)+(b<<10))
```

A macro acima representa uma “função” que, dados três números de 0 a 31 (5 bits), monta um número de 15 bits que representa uma cor no sistema RGB, utilizado na paleta de cores do GBA.

```
VIDEOMODE = (VMODE_0 | OBJ_ENABLE | OBJ_MAP_1D | H_BLANK_OAM);  
Objeto *objeto = (Objeto*) 0x02000000;
```

O trecho de código acima configura o modo de vídeo através de macros e depois “aloca” memória para uma estrutura do tipo **Objeto**. 0x02000000 é o endereço em que começa a memória **EWRAM**.



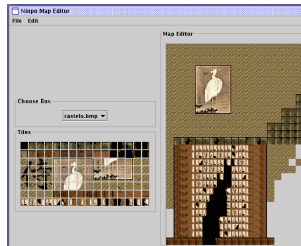
# SUB PRODUTOS

Tivemos de criar algo além do jogo principal:

**Asteróide** um pseudo-jogo para nos familiarizar com o GBA



**Ninpo** ferramenta de criação de tiles e edição de mapas



# TESTES

Para testarmos e depurarmos o jogo utilizamos:

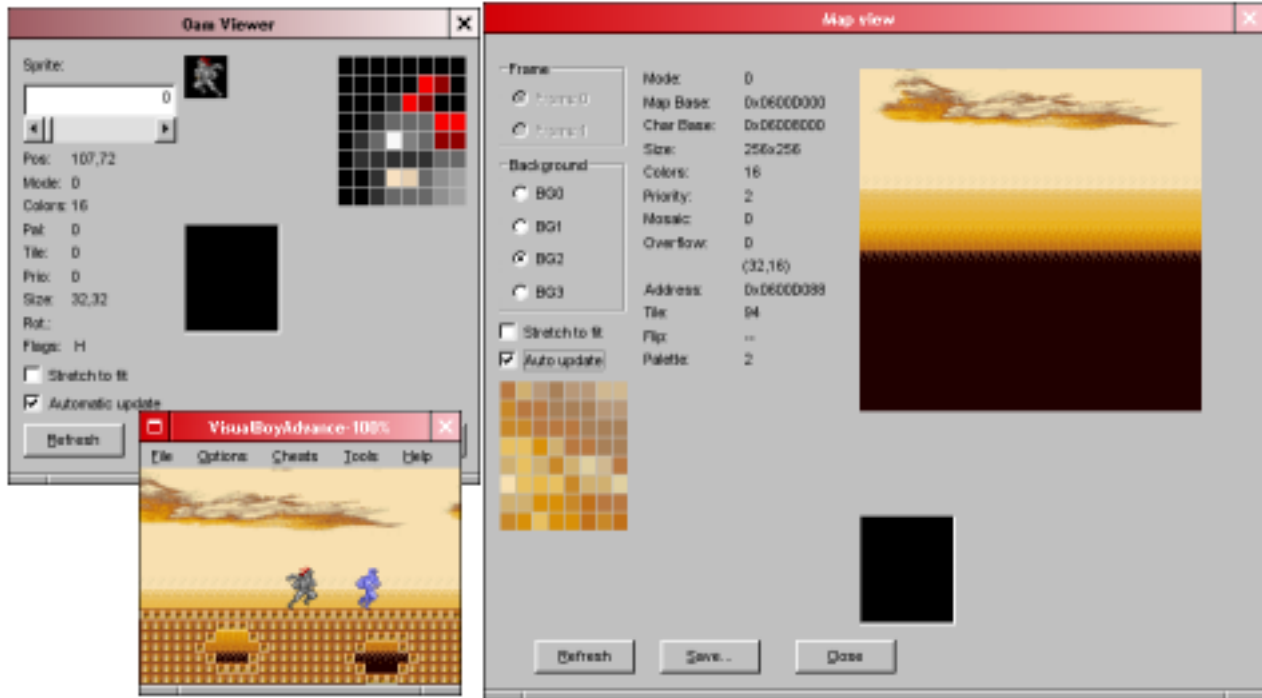
- ◇ o emulador Visual Boy Advance
- ◇ cartuchos com memória flash, permitindo gravação de dados



O Visual Boy Advance fornece recursos muito úteis para a depuração de nosso jogo.



# TESTES



# O BCC

Muitas das disciplinas foram úteis neste projeto:

- ◇ Estrutura de Dados
- ◇ Organização de Computadores
- ◇ Laboratório de Programação I e II
- ◇ Programação Orientada a Objetos
- ◇ MAC110, MAC122, MAC300 ...



# PERGUNTAS E SUGESTÕES

Dúvidas?



# REFERÊNCIAS

1. HAPP, Tom.: CowBite Virtual Hardware Specifications.
2. KERNIGHAN, B., Ritchie, D.: C: A linguagem de programação padrão ANSI. São Paulo, 1989. Ed. Campus.
3. VIJN, Jasper: TONC GBA Programming

