



Trabalho Arq. de Computadores II

Wolfenstein

GRUPO:

- Cauã Cola Zandonadi
- Davi Ribeiro Carvalho
- Guilherme Sousa Fagundes
- Giseli Rosa Lourenço
- Matheus Miranda Griffio

Programação e Ambiente de desenvolvimento

- O DESENVOLVIMENTO FOI FEITO COM BORLAND C++ 3.1, MAS A LINGUAGEM USADA FOI C.
- O AMBIENTE DA BORLAND INTEGRAVA FERRAMENTAS DE COMPILAÇÃO, EXECUÇÃO E DEPURAÇÃO, DENTRO DO PRÓPRIO AMBIENTE INTEGRADO.
- POR PADRÃO RODAVA NO MODO VGA 3, OFERECENDO UMA TELA COM 80 CARACTERES DE LARGURA E 25 DE ALTURA.

```
File Edit Search Run Compile Debug Project Options Window Help
WL_MAIN.C 2=11
=====
=
= main
=
=====
*/
char *nosprtxt[] = {"nospr",nil};

void main (void)
{
    int i;
    CheckForEpisodes();
    Patch386 ();
    InitGame ();
    DemoLoop();
    Quit("Demo loop exited???");
}

1588:19
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
```

Divisão das Responsabilidades

John Carmack

Cuidou do código de tempo de execução

John Romero

Programou muitas das ferramentas (editor de mapas TED5, compactador de recursos IGRAB, compactador de som MUSE)

Jason Blochowiak

Escreveu subsistemas importantes do jogo (gerenciador de entrada, gerenciador de páginas, gerenciador de som, gerenciador de usuários).

Solução Técnica

PARA CONTORNAR AS LIMITAÇÕES DE VISUALIZAÇÃO DAS TELAS CTR DURANTE O PROCESSO DE DEPURAÇÃO CONSISTIU NA UTILIZAÇÃO DE DOIS MONITORES SIMULTANEAMENTE.

- A **PLACA VGA** OPERAVA EM MODO GRÁFICO (MODO 13H), UTILIZANDO O ENDEREÇO 0XA0000 PARA O FRAMEBUFFER, ISTO É, PARA EXECUTAR O JOGO NORMALMENTE.
- A **PLACA MDA** OPERAVA EM MODO TEXTO MONOCROMÁTICO, ACESSANDO O ENDEREÇO 0XB8000 PARA COLETA DE DADOS.

ESSA SEPARAÇÃO POSSIBILITAVA A **EXECUÇÃO DO JOGO** EM MODO GRÁFICO SEM INTERFERÊNCIAS, **AO MESMO TEMPO** EM QUE FERRAMENTAS COMO O TURBO DEBUGGER 386 EXIBIAM INFORMAÇÕES DE **DEPURAÇÃO** NO MONITOR SECUNDÁRIO.

Solução Técnica

```
File Edit View Run Breakpoints Data Options Window Help READY...
[ ]=CPU 80486
main:
cs:17FA>55      push  bp
cs:17FB 8BEC     mov   bp,sp
#WL_MAIN#1606:
cs:17FD 9A1C3D614F call  far _CheckForEpisodes
#WL_MAIN#1608:
cs:1802 0E       push  cs
cs:1803 E80BEB     call  _Patch386
#WL_MAIN#1610:
cs:1806 0E       push  cs
cs:1807 E863FA     call  #WL_MAIN#1145
#WL_MAIN#1612:
cs:180A 0E       push  cs
cs:180B E8DBFD     call  _DemoLoop
#WL_MAIN#1614:
48AB:0000 CD 20 FF 9F 00 EA FF FF = f Ω
48AB:0008 AD DE E0 01 CC 15 AA 01 i |x|S-
48AB:0010 CC 15 89 02 27 10 99 01 ||Së'>00
48AB:0018 01 01 01 00 02 FF FF FF 000 0
48AB:0020 FF FF FF FF FF FF FF FF

ax FF00 c=0
bx 4768 z=1
cx 0000 s=0
dx 4768 o=0
si 474A p=1
di 4768 a=0
bp FFEC i=1
sp FFE0 d=0
ds 824B
es 824B
ss 824B
cs 4D44
ip 17FA

ss:FFE2 48BB
ss:FFE0>0170
ss:FFDE 3246
ss:FFDC 4D44
ss:FFDA 17FB

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

Tela MDA com o depurador Borland 3.1

Solução Técnica



Tela VGA executando jogo

Solução Técnica

Outra maneira de melhorar o espaço na tela era usar o modo de texto de "alta resolução" 50x80.

Exemplo: O arquivo WL_MAIN.C aberto em ambos os modos demonstra a compensação entre legibilidade e visibilidade.

```
File Edit Search Run Compile Debug Project Options Window Help
\WOLF3D\WOLFSRC\WL_MAIN.C 1=[↑]
/*
=====
=
= main
=
=====
*/
char *nosprtxt[] = {"nospr",nil};

void main (void)
{
    int i;

#ifdef BETA
    //
    // THIS IS FOR BETA ONLY!
    //
    struct dosdate_t d;

```

1595:3

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
File Edit Search Run Compile Debug Project Options Window Help
\WOLF3D\WOLFSRC\WL_MAIN.C 1=[↑]
/*
=====
main
=====
*/
char *nosprtxt[] = {"nospr",nil};
void main (void)
{
    int i;

#ifdef BETA
    // THIS IS FOR BETA ONLY!
    struct dosdate_t d;
    dos_getdate(&d);
    if (d.year > YEAR ||
        (d.month >= MONTH && d.day >= DAY))
    {
        printf("Sorry, BETA-TESTING is over. Thanks for you help.\n");
        exit(1);
    }
#endif

    CheckForEpisodes();
    Patch386 ();
    InitGame ();
    DemoLoop();
    Quit("Demo loop exited???");
}

```

9:43

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Assets Gráficos

- ADRIAN CARMACK PRODUZIU TODOS OS ASSETS GRÁFICOS.
- TODO O TRABALHO FOI FEITO NO DELUXE PAINT, EDITOR GRÁFICO EM BITMAP DESENVOLVIDO PELA ELECTRONIC ARTS.
- NÃO UTILIZARAM FERRAMENTAS DE SCANNEAMENTO PRESENTES NA EPOCA, TODOS OS ASSETS FORAM FEITOS MANUALMENTE, PIXEL POR PIXEL.
- ADRIAN DESENHAVA NA RESOLUÇÃO NATIVA DO JOGO, 320×200 PIXELS, DIRETAMENTE COM MOUSE.



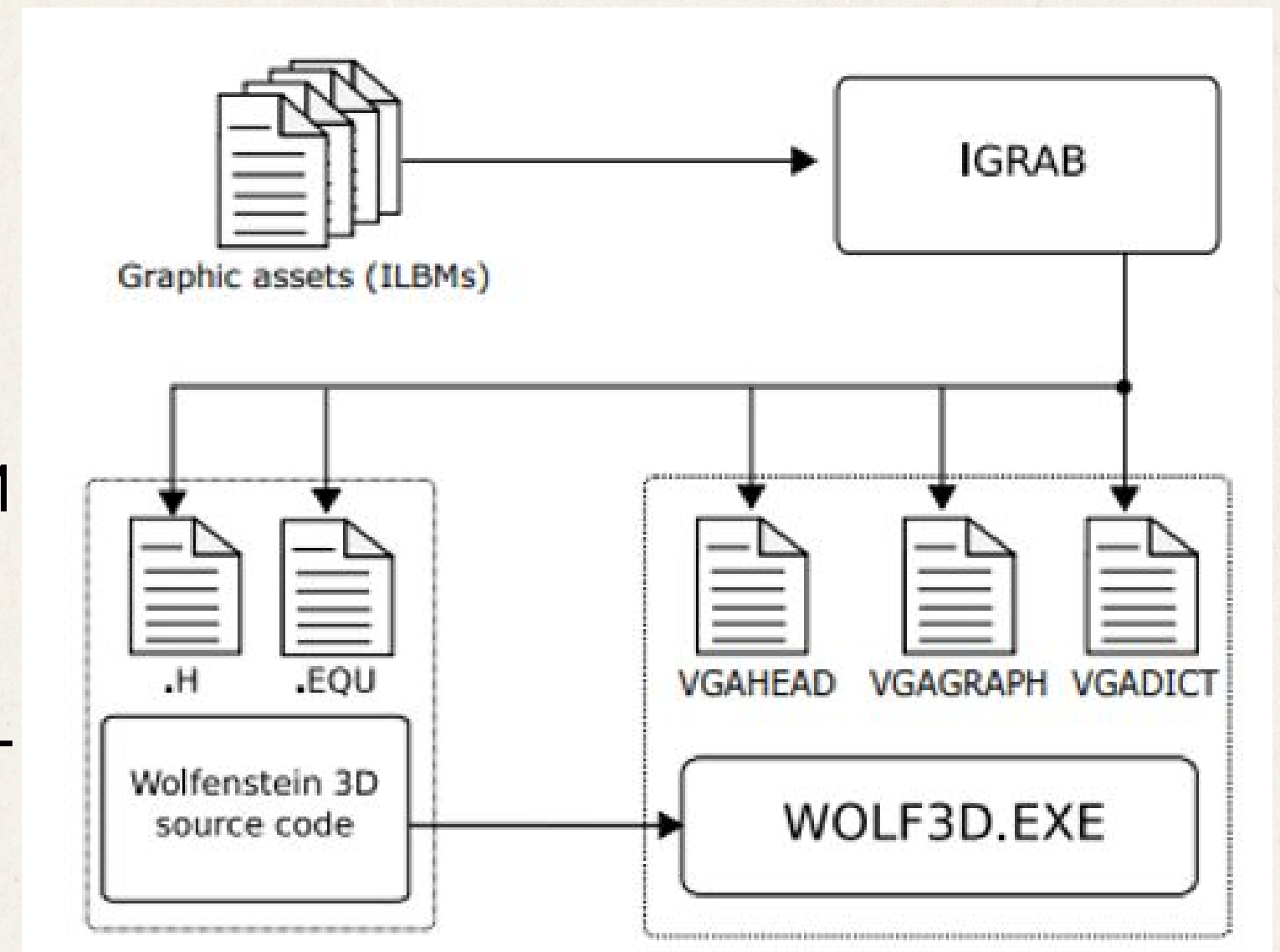
Assets Gráficos

- O VGA NO MODO 13H NÃO OPERAVA NO PADRÃO 24BITS-RGB, EM VEZ DISSO, ERA UTILIZADO UMA PALETA COM 256 CORES, ONDE CADA PIXEL POSSUIA UM INDICE QUE APONTAVA PARA A PALETA.
- ADRIAN CRIOU A PALETA COM AS 256 CORES E AS UTILIZOU EM TODOS OS ASSETS DO JOGO
- CADA PIXEL CONTIA UM INDICE QUE INDICAVA PARA A PALETA.



Workflow de Assets

- DEPOIS DE GERADOS, OS ASSETS GRÁFICOS, JÁ NO FORMATO ILBM (PADRÃO DO DELUXE PAINT), ERAM EMPACOTADOS EM UM ARQUIVO.
- ERA ENTÃO GERADO UM ARQUIVO HEADER EM C, CONTENDO OS IDS DE CADA ASSET.
- A ENGINE ENTÃO, REFERÊNCIAVA CADA ASSET USANDO DIRETAMENTE ESSES IDS.
- ESTE PROCESSO ERA FEITO UTILIZANDO A FERRAMENTA IGRAB.



Workflow de Assets

- NO CÓDIGO DA ENGINE, A UTILIZAÇÃO DOS ASSETS É FEITA USANDO UM ENUM.
- ESSE ENUM, É UM OFFSET DO ARQUIVO HEADER, QUE POR SUA VEZ, É UM OFFSET DO ARQUIVO DE ASSETS.
- COM ESSAS CAMADAS INDIRETAS, ERA POSSÍVEL REGENERAR E REORDENAR ASSETS A VONTADE, SEM MEXER NO CÓDIGO FONTE.

```
////////////////////////////////////  
//  
// Graphics .H file for .WL1  
// IGRAB-ed on Sun May 03 01:19:32 1992  
//  
////////////////////////////////////  
  
typedef enum {  
    // Lump Start  
    H_BJPIC-3,   
    H_CASTLEPIC, // 4  
    H_KEYBOARDPIC, // 5  
    H_JOYPIC, // 6  
    H_HEALPIC, // 7  
    H_TREASUREPIC, // 8  
    H_GUNPIC, // 9  
    H_KEYPIC, // 10  
    H_BLAZEPIC, // 11  
    H_WEAPON1234PIC, // 12  
    H_WOLFLOGOPIC, // 13  
    ...  
    PAUSEDPIC, // 140  
    GETPSYCHEDPIC, // 141
```

Encerramento

