glslAngel:
Uma Ferramenta de Depuração para a Linguagem GLSL

Bárbara Bellaver Gonçalves

Manuel M. Oliveira

Orientador

{bbgoncalves, oliveira}@inf.ufrgs.br

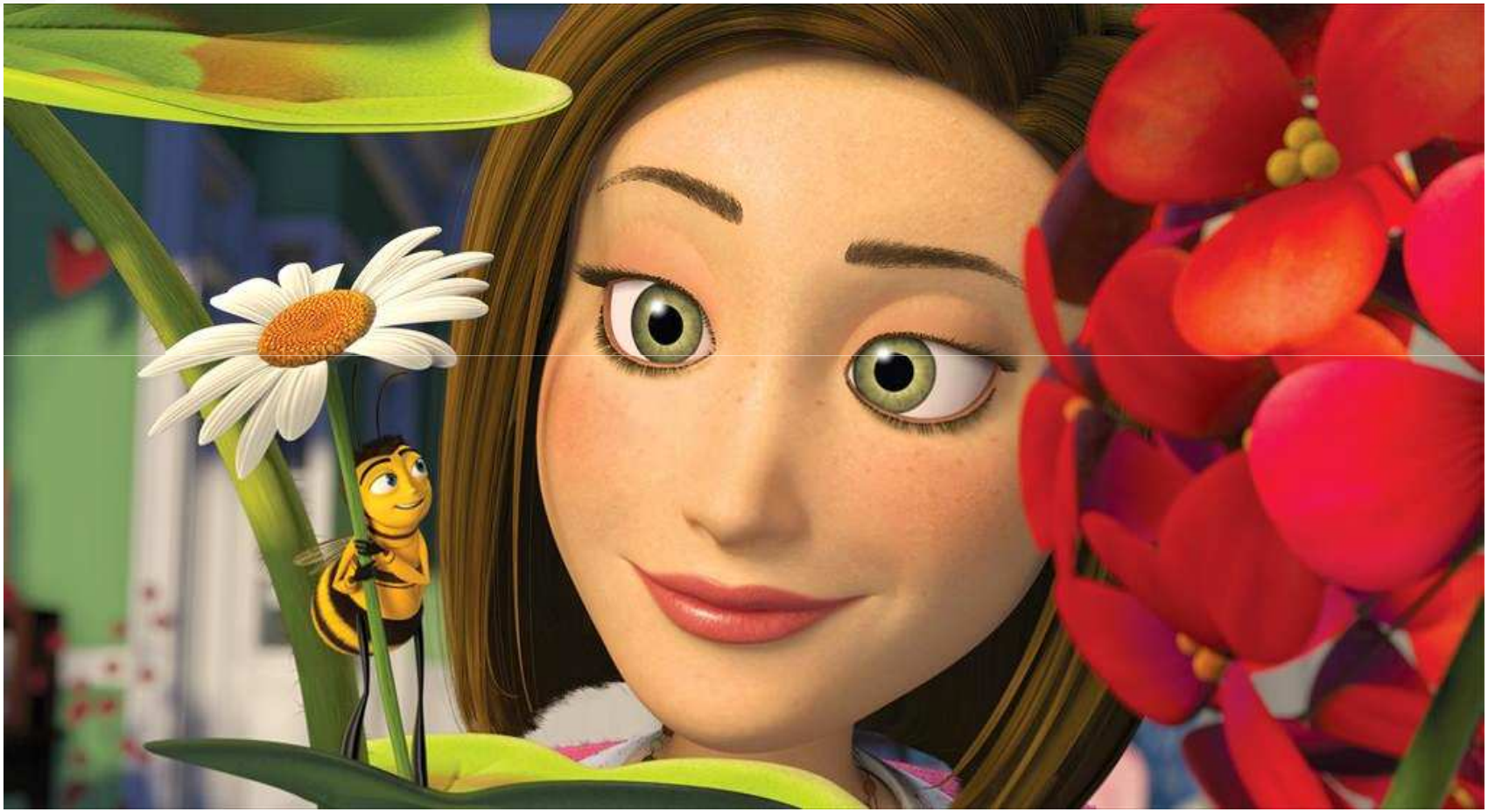
CG UFRGS

Informática UFRGS

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Microsoft

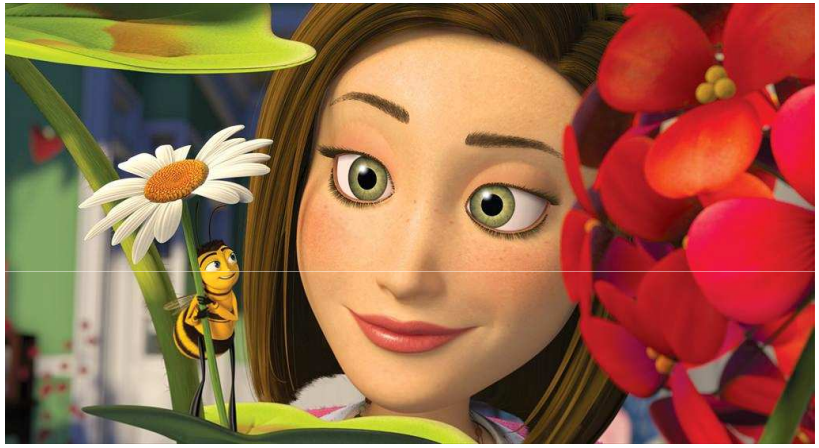
Introdução



© Bee Movie, DreamWorks

UFRGS

Introdução (cont.)



© Bee Movie, DreamWorks

© Wall-E, Pixar

© Ratatouille, Pixar

Introdução (cont.)

© Crysis, Crytek

© Grid, Codemasters

© Far Cry 2, Ubisoft

UFRGS

Como Obter Estes Efeitos?

- Shaders
  - Programas desenvolvidos para GPUs.

Sem shader

© Crysis, Crytek

Com shader

© Crysis, Crytek

Evolução das Placas Gráficas

© Quake 1, Activision

© Fallout 3, Bethsoft

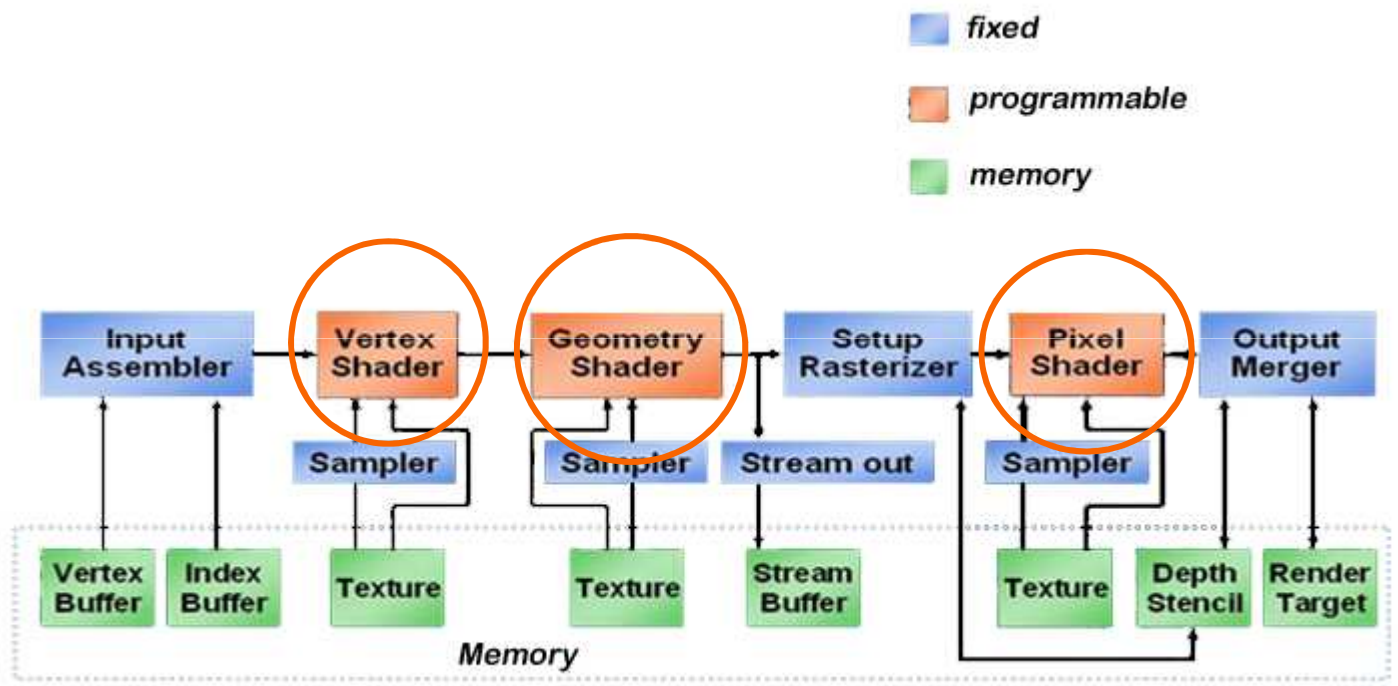
Voodoo 1, 1997

- 2 engines de renderização 3D
- 1 processador multimidia 2D
- 6MB de EDO RAM

GeForce GTX 280, 2008

- 240 Núcleos de processamento
- 1GB de memória

O Pipeline Gráfico



© MSDN, http://msdn.microsoft.com

Programação de Shaders

- Evolução similar à programação de CPUs
  - Assembly → Alto Nível
- Exemplos de linguagens de alto nível
  - Cg, HLSL e GLSL
- Dificuldades atuais:
  - Programação paralela
    - Cultura de programação seqüencial é a mais difundida
  - Ferramentas de desenvolvimento deficitárias
    - Editores e depuradores separados, por exemplo

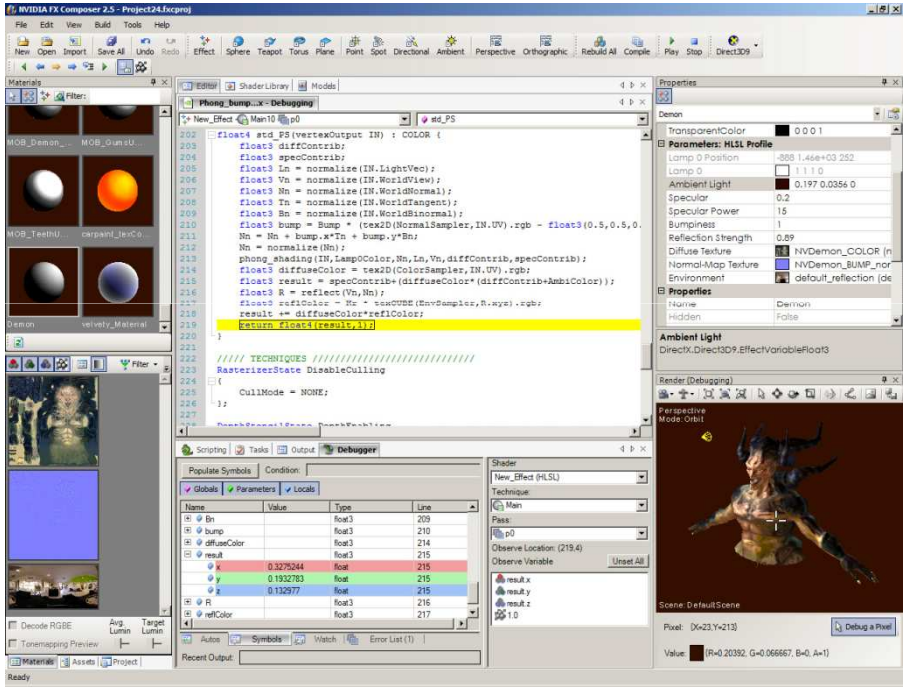
Exemplo de Shader

```
PSSceneIn VSSkymain(VSSceneIn input)
{
    PSSceneIn output;


    // Transform the vert to view-space

    float4 v4Position = mul(float4(input.pos, 1),
                            g_mWorldViewProj);
    output.pos = v4Position;


    // Transfer the rest

    output.tex = input.tex;
    output.color = float4(1,1,1,1);

    return output;
}
```

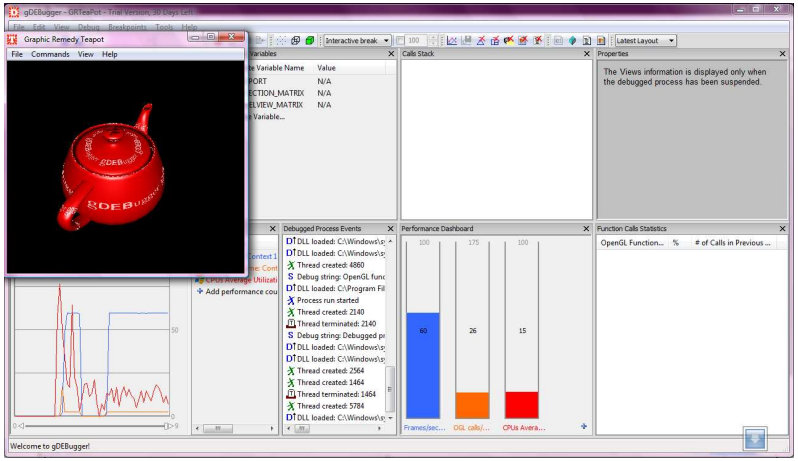
GLSL

- Linguagem padrão
- Garante compatibilidade
- Fácil adaptar para aplicações existentes
- Código otimizado para hardware utilizado

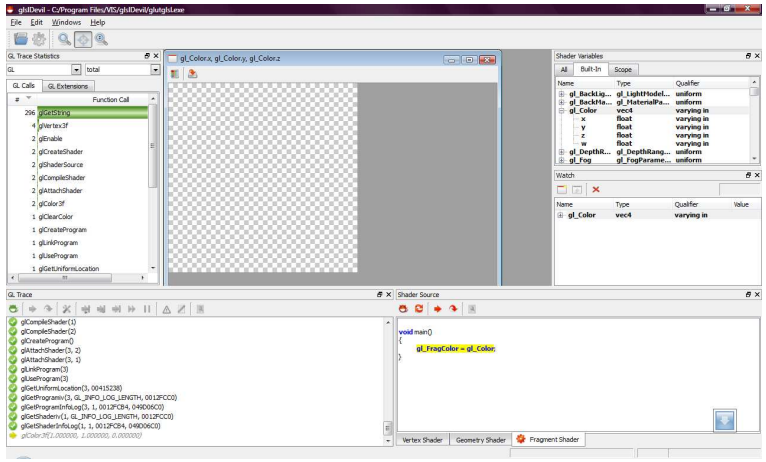
Ferramentas Existentes



[FxComposer, 2008]



[gDEBugger, 2008]



[glslDevil, 2007]

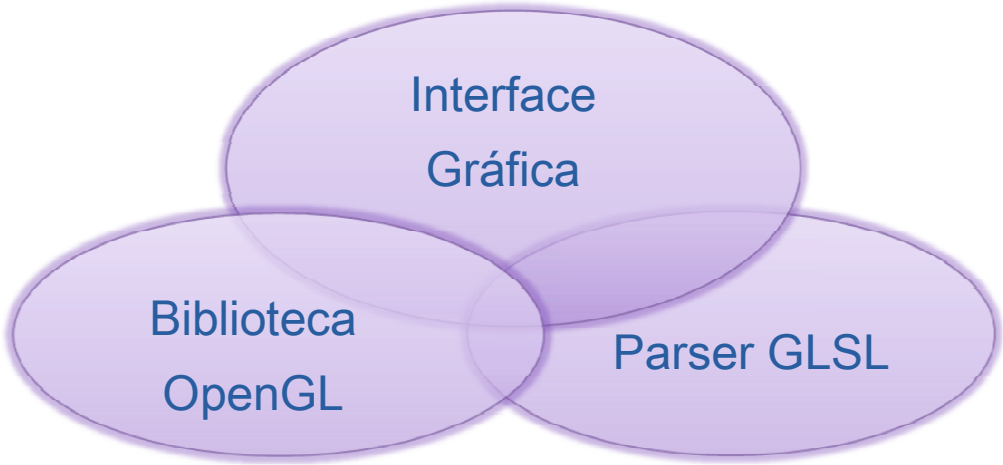
Objetivos

- Desenvolver um depurador de shaders GLSL
  - GLSL é um padrão aberto
  - Poucas ferramentas existentes
  - Multiplataforma (Linux, Windows, etc.)

- Principal dificuldade:
  - Dados não estão facilmente disponíveis

A Ferramenta

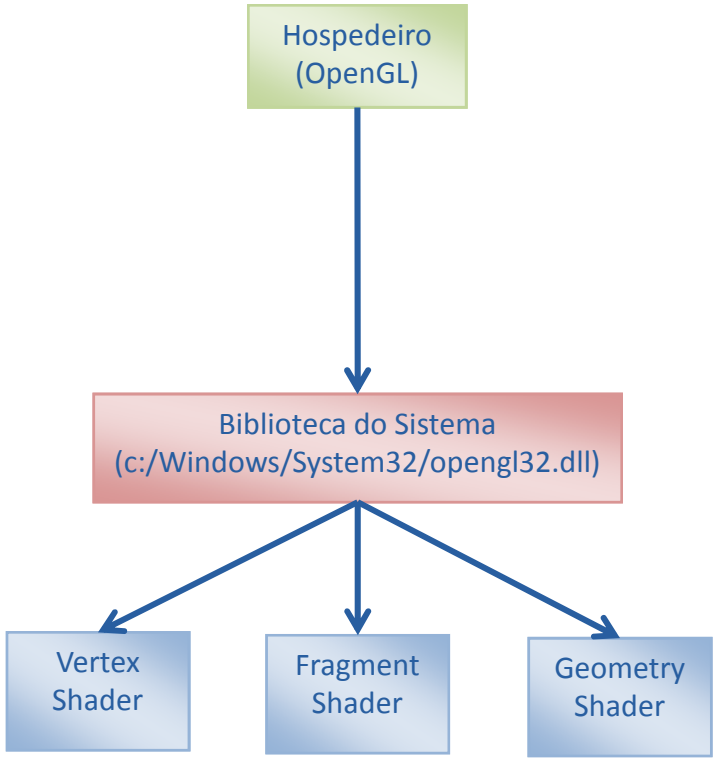
- Inspirada na ferramenta *glslDevil* [Strengert et. Al, 2007]
- Projeto de código aberto
  - www.codeplex.com/hl2glsl
- Permite depurar, editar e compilar o shader



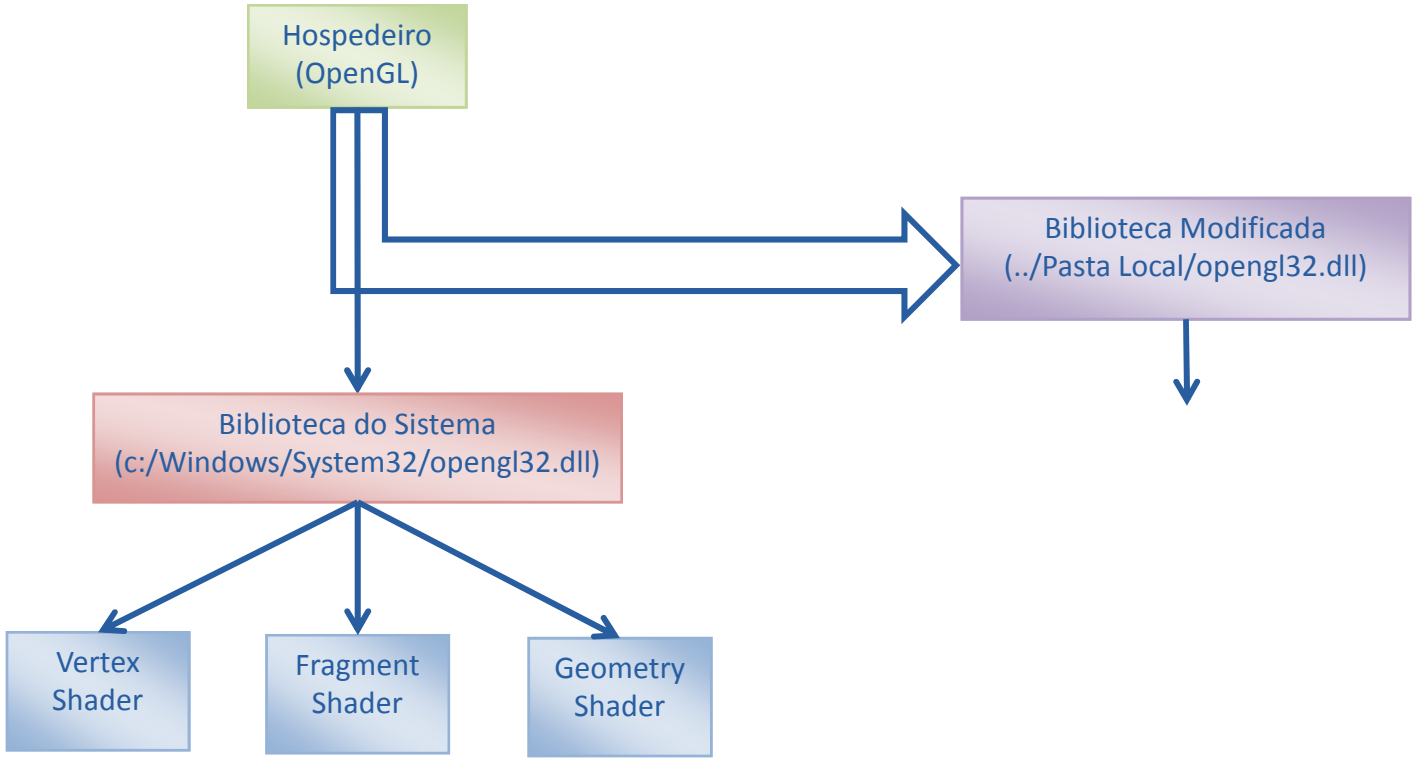
A Ferramenta (cont.)

Demo 1

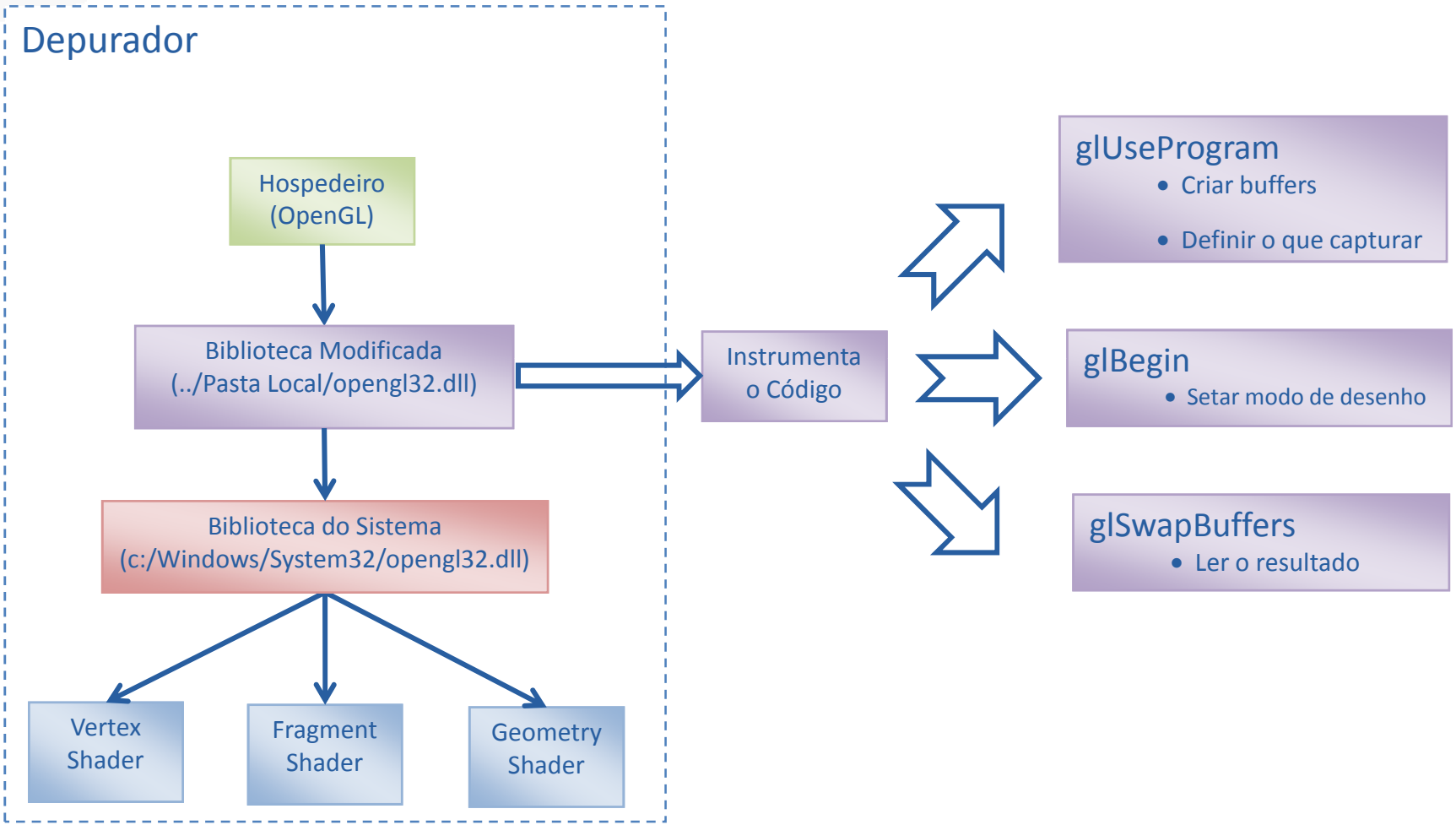
Fluxograma de Execução



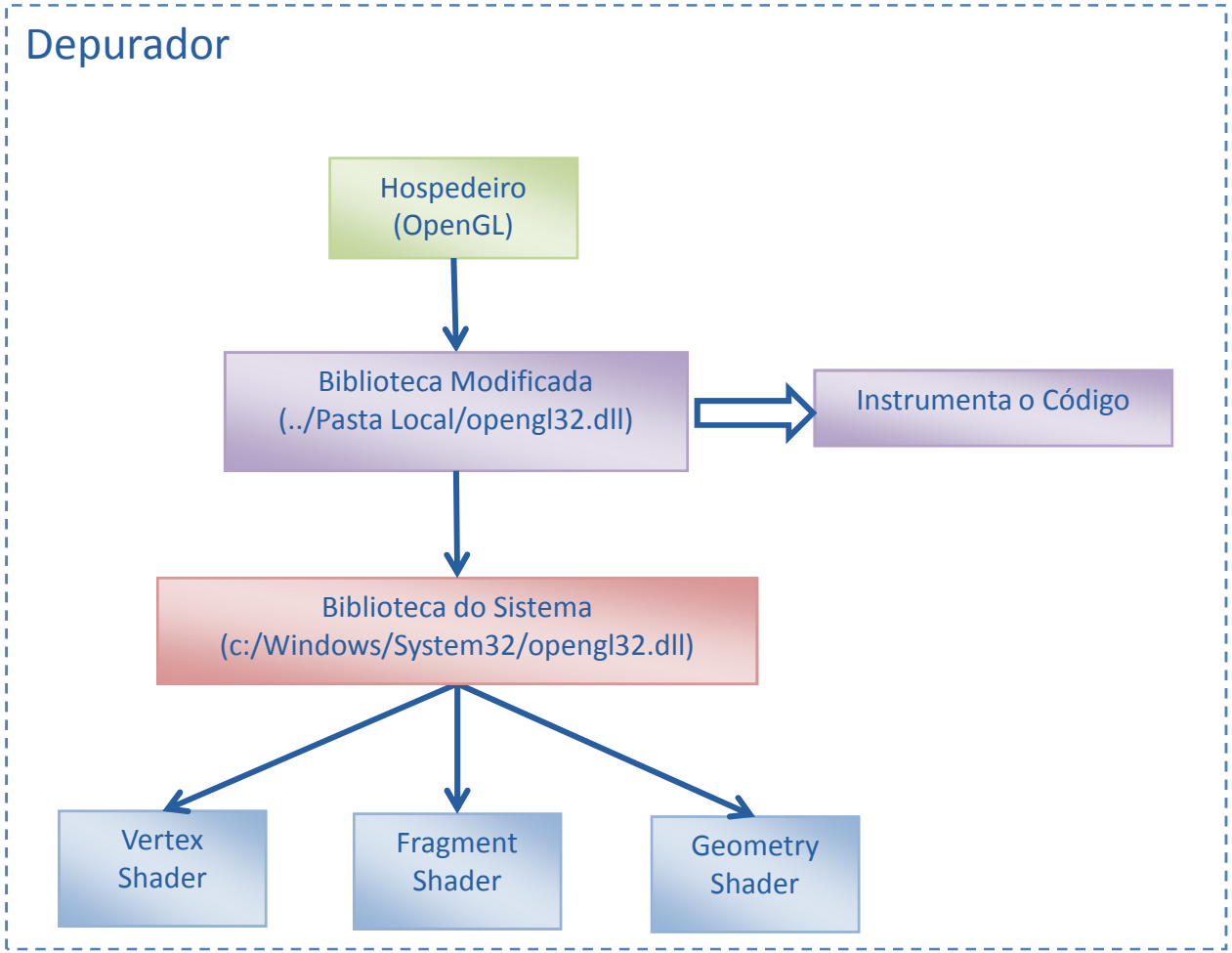
Intercepta Chamadas OpenGL



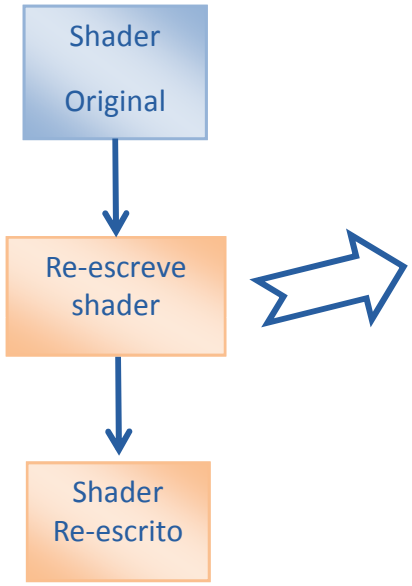
Instrumentação do Hospedeiro



Instrumentação do Hospedeiro (cont.)



Re-escrita do Shader



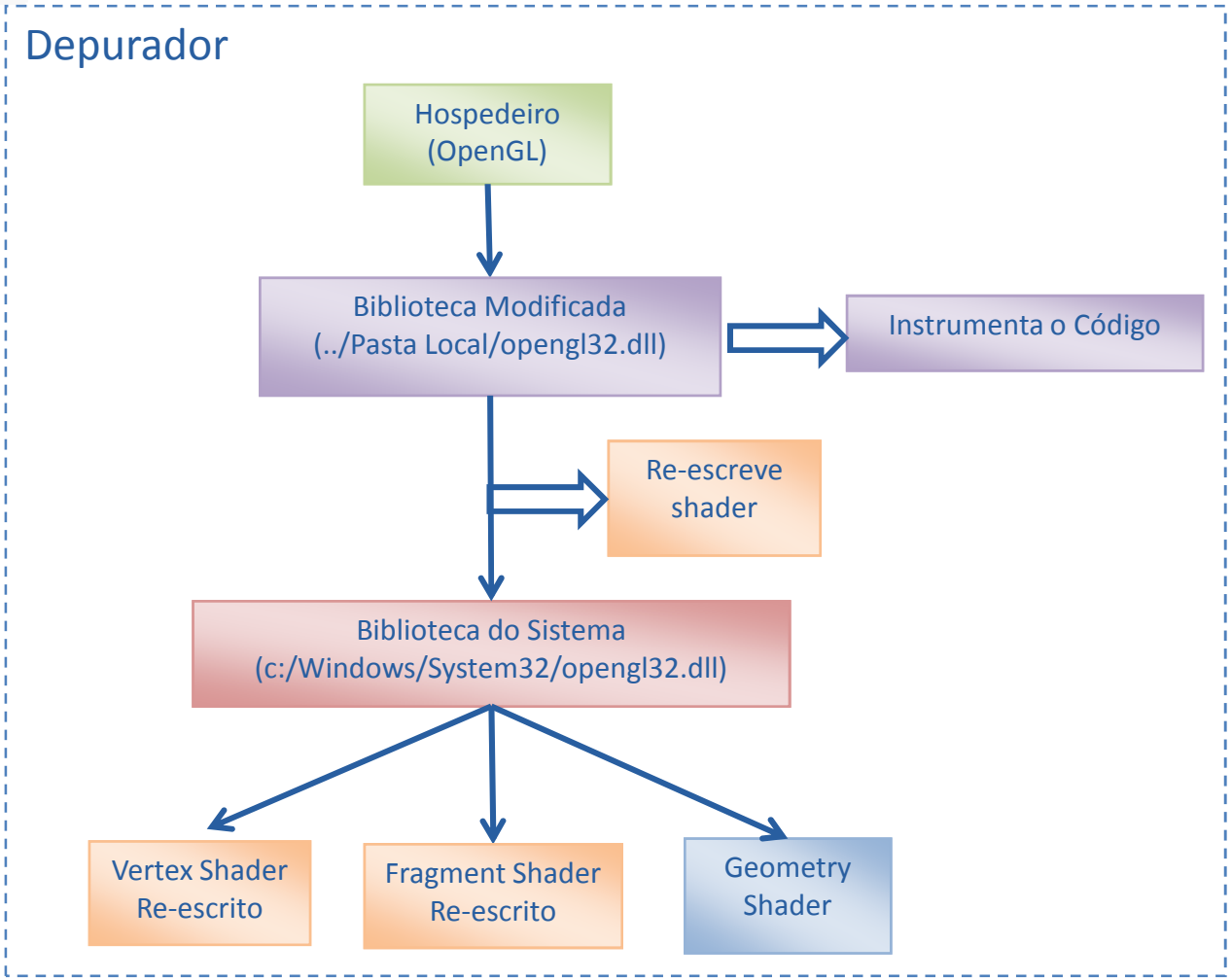

```
void main(){
   vec4 cor = vec4(0.2,0.2,0.1,0.3);
}
```

*código recebido

```
varying vec4 dbg_varUser;
void main(){
   dbg_varUser = vec4(0.2,0.2,0.1,0.3);
}
```

*código gerado

Fluxograma de Execução Final



Resultados

Demo 2

Conclusões

- Ferramenta de depuração
  - Programa OpenGL hospedeiro
  - Shader GLSL
- Desenvolvimento integrado da aplicação
- Atualmente, é dependente de hardware

Sugestões para Projetistas de HW

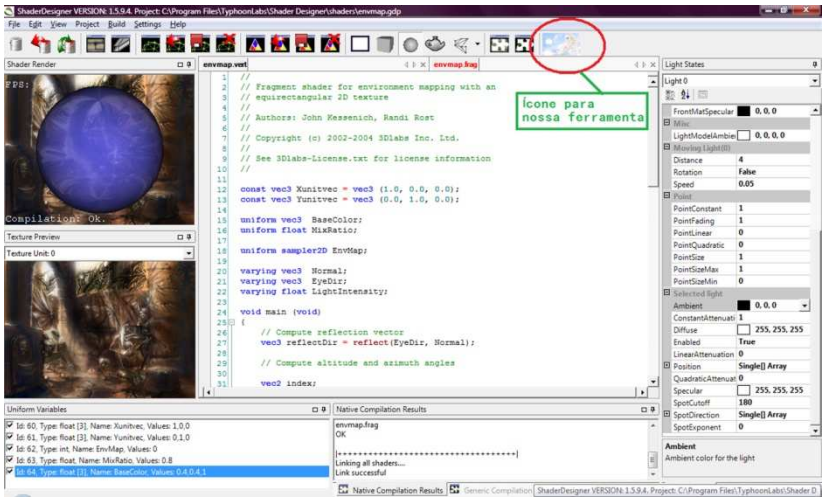
- Modificações que tornariam tudo mais fácil
  - Acesso ao que acontece na placa
  - Uso de buffers mais transparente
  - Melhoria da documentação

Trabalhos Futuros

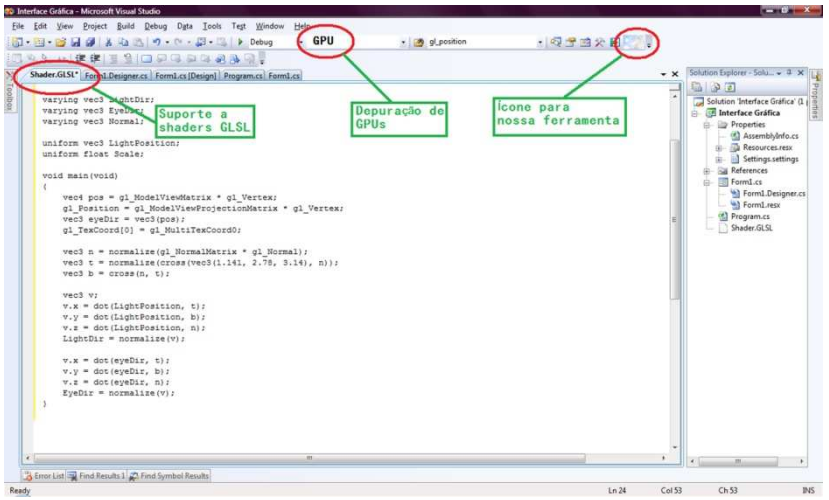
- Suporte à geometry shaders
- Novas formas de interação
- Seleção do vértice para obter informação
- Otimização do código
- Suporte a mais de um shader do mesmo tipo

Trabalhos Futuros

- Integração com ferramentas existentes
  - Shader Designer [Typhoon Labs, 2006]
  - Microsoft Visual Studio
- Ambiente único de desenvolvimento



Shader Designer



Visual Studio

Agradecimentos

- Microsoft
- Prof. Manuel Oliveira
- Grupo de Computação Gráfica
- Leandro Fernandes e Vitor Pamplona
- Minha família
- Power Trio Quindim
- Alexandre Coster, Eduardo Camaratta e Mariane Machado

Perguntas?