像素着色器的设计

# 内存布局

## Shader流水线



## 内存布局类型

### 水平布局（Horizontal Memory Layout）

输入数据按成员顺序排列，相同的成员放置在一起。例如对于像素而言，有Block

|  |  |
| --- | --- |
| P00 | P01 |
| P10 | P11 |

每个像素有三个成员A,B,C，那么内存布局是依据下图所示进行排列。

|  |  |  |
| --- | --- | --- |
| P00 | A | Memory |
| P01 |
| P10 |
| P11 |
| P00 | B |
| P01 |
| P10 |
| P11 |
| P00 | C |
| P01 |
| P10 |
| P11 |

### 垂直布局（Vertical Memory Layout）

输入变量将相同变量的数据排列在一起。上例中的垂直内存布局如下例：

|  |  |  |
| --- | --- | --- |
| P00 | A | Memory |
| B |
| C |
| P01 | A |
| B |
| C |
| P10 | A |
| B |
| C |
| P11 | A |
| B |
| C |

### 流式布局（Stream Memory Layout）

输入数据为数据流的指针，需要进行二次寻址。每一条数据流均是一段连续的数据，流内数据是垂直布局，例如对于一组顶点数据，我们可以有：

|  |  |  |
| --- | --- | --- |
| A Vertex | A\* | Memory |
| B\* |
| C\* |

对应的Vertex Streams的布局为：

AB流：

|  |  |  |
| --- | --- | --- |
| V0 | A | Memory |
| B |
| V1 | A |
| B |
| … | … |

C流：

|  |  |  |
| --- | --- | --- |
| V0 | C | Memory |
| V1 | C |
| … | … |

## Shader流水线数据内存布局

|  |  |  |
| --- | --- | --- |
| Data | Layout | Alignment |
| Vertex Stream Data | Stream | N/A3 |
| VS Constant Buffer | Vertical1 | 变量16字节对齐  成员由小到大排列  C Structure兼容  1,2,4,8字节的同类型数据均被打包为16字节，首个成员为16字节对齐，余位补0值。如两个float2被打包为一个16字节的float4。  16字节以下的非2幂的数据均补齐至16字节。  对于matrix，将每行均补齐或打包至16字节。4 |
| Processed Vertex | Vertical | 同上 |
| Pre-Processing Pixel Block Data | Horizontal | 变量16字节对齐。  每个Block均有16个像素水平布局而成。  16字节以下，非二幂大小的数据均补齐为2幂的大小。  对于matrix，将行打包或补齐至16字节。5 |
| PS Constant Buffer | Vertical1 | 同VS Constant Buffer |
| Frame Buffer | Stream2 | N/A3 |

注1 Constant Buffer只有一份，所以为Vertical布局

注2 Frame Buffer一定流式布局的，而且基本上每个流都只有一个成员。

注3 流式数据均从外部获取，所以无法确定Alignment。一般情况下，2幂的数据对齐与其大小相同。非2幂的数据默认是非对齐的。

注4 （补充实例）

注5 （补充实例）

# 指令布局

水平布局（Horizontal Instruction Layout）

垂直布局（Vertical Memory Layout）

分支处理（Branch）