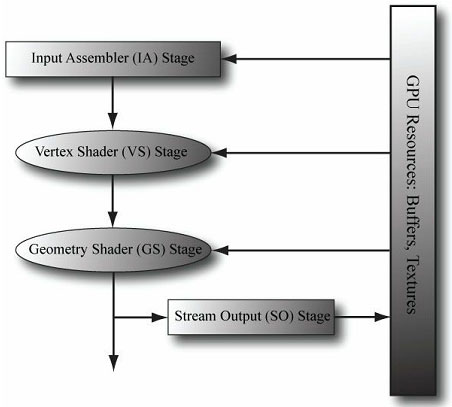
# 20.5 流输出

我们知道GPU可以向纹理写入数据。例如，GPU可以向深度/模板缓冲区和后台缓冲区写入数据。Direct3D 10的一个新特性是流输出（stream output，简称SO）阶段。它允许GPU向绑定在管线SO阶段上的顶点缓冲区*V*写入几何体数据（以一个顶点列表的形式）。尤其是从几何着色器输出的顶点都会被写入（或传送）到*V*中。随后，我们可以把*V*中的几何体渲染出来。图20.4说明了这些概念。在我们的粒子系统框架中，流输出（stream output）具有非常重要的作用。

****

**图20.4 图元被推送到管线中。由几何着色器输出的图元会被流输出到GPU内存中的顶点缓冲区内。**

## 20.5.1 创建用于流输出的几何着色器

当使用流输出时，我们必须专门创建几何着色器。下面的代码示范了如何在effect文件中完成这一工作：

GeometryShader gsStreamOut = ConstructGSWithSO(

CompileShader( gs\_4\_0, GS() ),

"POSITION.xyz; VELOCITY.xyz; SIZE.xy; AGE.x; TYPE.x");

technique11 SOTech

{

pass P0

{

SetVertexShader( CompileShader( vs\_4\_0, VS() ) );

SetGeometryShader( gsStreamOut );

SetPixelShader( CompileShader( ps\_4\_0,PS() ));

}

}

**ConstructGSWithSO**的第1个参数是编译后的几何着色器。第2个参数是一个字符串，它描述了将要被流输出的顶点的格式（即，几何着色器输出的顶点的格式）。在上面的例子中，顶点格式为：

struct Vertex

{

float3 initialPosW : POSITION;

float3 initialVelW : VELOCITY;

float2 sizeW : SIZE;

float age : AGE;

uint type : TYPE;

};

## 20.5.2 仅有流输出的technique

当在正常情况下使用流输出时，由几何着色器输出的顶点会被流输出到GPU内存中的顶点缓冲区内，并且被推送到渲染管线的下一阶段（光栅化）。如果你希望某个technique只用来传送数据，而不对数据进行渲染，那么就必须禁用像素着色器和深度缓冲区。（禁用像素着色器和深度缓冲区就相当于禁用光栅化。）下面的代码示范了如何完成一工作：

DepthStencilState DisableDepth

{

DepthEnable = FALSE;

DepthWriteMask = ZERO;

};

GeometryShader gsStreamOut = ConstructGSWithSO(

CompileShader( gs\_5\_0, StreamOutGS() ),

"POSITION.xyz; VELOCITY.xyz; SIZE.xy; AGE.x; TYPE.x" );

technique11 StreamOutTech

{

pass P0

{

SetVertexShader( CompileShader( vs\_5\_0, StreamOutVS() ) );

SetGeometryShader( gsStreamOut );

// 禁用像素着色器

SetPixelShader(NULL);

// 禁用深度缓冲

SetDepthStencilState( DisableDepth, 0 );

}

}

在我们的粒子系统中，我们将使用一个仅有流输出的technique（stream output-only technique）来创建和销毁粒子（即，更新粒子系统）。每一帧：

**1．**使用仅有流输出的technique 生成当前的粒子列表。由于我们禁用了光栅化功能，所以不会在屏幕上渲染任何粒子。

**2．**使用pass中的几何着色器，根据各种条件创建和销毁粒子，使粒子系统不断发生变化。

**3．**更新后的粒子列表会被流输出到一个顶点缓冲区内。

应用程序随后会用另一个technique来渲染更新后的粒子列表。使用两个technique的主要原因是几何着色器要完成不同的工作。在仅有流输出的technique中，几何着色器用于输入粒子、更新粒子和输出粒子。而在用于渲染的technique中，几何着色器的任务是把点扩展为面对摄像机的四边形。由于几何着色器无法输出不同类型的图元，所以我们定义了两个几何着色器。

总之，我们需要在GPU上使用两个technique来渲染粒子系统：

**1．**一个technique 用于更新粒子系统。

**2．**一个technique 用于绘制粒子系统。

在以前的Direct3D版本中，粒子的更新工作总是在CPU上完成的。

**注意**：粒子的物理属性也可以在仅有流输出的pass中更新。不过，在我们的方案中，我们有一个位置函数*p*(t)。 所以我们不需要在仅有流输出的pass中更新粒子的位置和速度。SDK中的ParticlesGS示例示范了如何在仅有流输出的pass中更新粒子的物理属性，不过，它使用了不同的物理模型。

## 20.5.3 创建用于流输出的顶点缓冲区

我们在创建顶点缓冲区时必须加上一个**D3D11\_BIND\_STREAM\_OUTPUT**绑定标志值，只有这样才能把该顶点缓冲区绑定到SO阶段，让GPU向它写入数据。通常，作为流输出目标的顶点缓冲区随后就会作为管线的输入资源（即，它将要绑定到IA阶段，把内容渲染出来）。所以，我们还要加上一个**D3D11\_BIND\_VERTEX\_BUFFER**绑定标志值。下面的代码片段示范了如何创建一个用于流输出的顶点缓冲区：

D3D11\_BUFFER\_DESC vbd;

vbd.Usage = D3D11\_USAGE\_DEFAULT;

vbd.ByteWidth = sizeof(Vertex) \* MAX\_VERTICES;

vbd.BindFlags = D3D11\_BIND\_VERTEX\_BUFFER | D3D11\_BIND\_STREAM\_OUTPUT;

vbd.CPUAccessFlags = 0;

vbd.MiscFlags = 0;

HR(md3dDevice->CreateBuffer(&vbd, 0, &mStreamOutVB));

注意，缓冲区内存不必初始化，因为GPU会向它写入顶点数据。还要注意，缓冲区的大小有限，不要向缓冲区传送过多的顶点，小心溢出。

## 20.5.4 绑定到SO阶段

带有**D3D11\_BIND\_STREAM\_OUTPUT**绑定标志值的顶点缓冲区可以被绑定到管线的SO阶段，我们使用如下方法完成这一工作：

void ID3D11Device::SOSetTargets(

UINT NumBuffers,

ID3D11Buffer \*const \*ppSOTargets,

const UINT \*pOffsets);

1. **NumBuffers**：绑定到SO阶段的顶点缓冲区的数量。最大值为4。
2. **ppSOTargets**：绑定到SO阶段的顶点缓冲区数组。
3. **pOffsets**：一个偏移值数组，每个偏移值对应一个顶点缓冲区，指定SO阶段从何处开始写入顶点数据。

**注意**：这里有4个用于流输出的槽。当绑定到SO阶段的缓冲区少于4个时，空闲的槽应被设为空值。例如，当你只绑定到槽0（第1个槽）时，槽1、2、3应被设为空值。

## 20.5.5 解除与SO阶段的绑定

当顶点传送到顶点缓冲区之后， 我们可能希望绘制由些顶点定义的图元。但是，顶点缓冲区无法同时绑定到SO阶段和IA阶段。要解除顶点缓冲区与SO阶段的绑定，我们只需要将另一个缓冲区（它可以为空）绑定到SO阶段即可。下面的代码通过在槽0上绑定一个空缓冲区来解除对当前顶点缓冲区的绑定：

ID3D11Buffer\* bufferArray[1] = {0};

md3dDevice->SOSetTargets(1, bufferArray, &offset);

## 20.5.6 自动绘制

流输出到顶点缓冲区的几何体是可变的。那我们该如何确定所要绘制的顶点数量呢？幸好，Direct3D内部记录了顶点数量，我们可以使用**ID3D11Device::DrawAuto**方法绘制由SO阶段写入顶点缓冲区的几何体：

void ID3D11DeviceContext::DrawAuto();

**注意：**

1. 在调用**DrawAuto**方法之前，我们必须先将顶点缓冲区（它先前是一个流输出目标）绑定到IA阶段的输入槽0上。只有当一个带有**D3D11\_BIND\_STREAM\_OUTPUT**绑定标志值的顶点缓冲区绑定到IA阶段的输入槽0上时，才能调用**DrawAuto**方法。
2. 在调用**DrawAuto**方法之前，我们必须指定顶点在“流输出顶点缓冲区”中的顶点输入布局。
3. **DrawAuto**方法不使用索引，因为几何着色器只能输出由顶点列表定义的完整图元。

## 20.5.7 顶点缓冲区互换

如前所述，顶点缓冲区无法同时绑定到SO阶段和IA阶段。所以，我们要采用一种互换（ping-pong，直译为：乒乓球）机制来解决一问题。当使用流输出进行绘制时，我们创建两个顶点缓冲区。一个充当输入缓冲区，另一个充当输出缓冲区。当渲染到下一帧时，我们交换这两个缓冲区。使先前的输出缓冲区变成当前的输入缓冲区，使先前的输入缓冲区变成当前的输出缓冲区。下面的表格展示了顶点缓冲区*V*0和*V*1的3次互换过程。

|  |  |  |
| --- | --- | --- |
|  | 绑定到IA阶段的输入顶点缓冲区 | 绑定到SO阶段的输出顶点缓冲区 |
| 第*i*帧 | *V*0 | *V*1 |
| 第*i*+1帧 | *V*1 | *V*0 |
| 第*i*+2帧 | *V*0 | *V*1 |