



NVIDIA®

**DX10 – новые возможности
и вопросы производительности**

Юрий Уральский, NVIDIA

DX10



- **Много новой функциональности**
 - Не только геометрические шейдеры
- **Возможность перестроить ваш графический движок**
- **Мотивация к созданию новых эффектов**

Outline



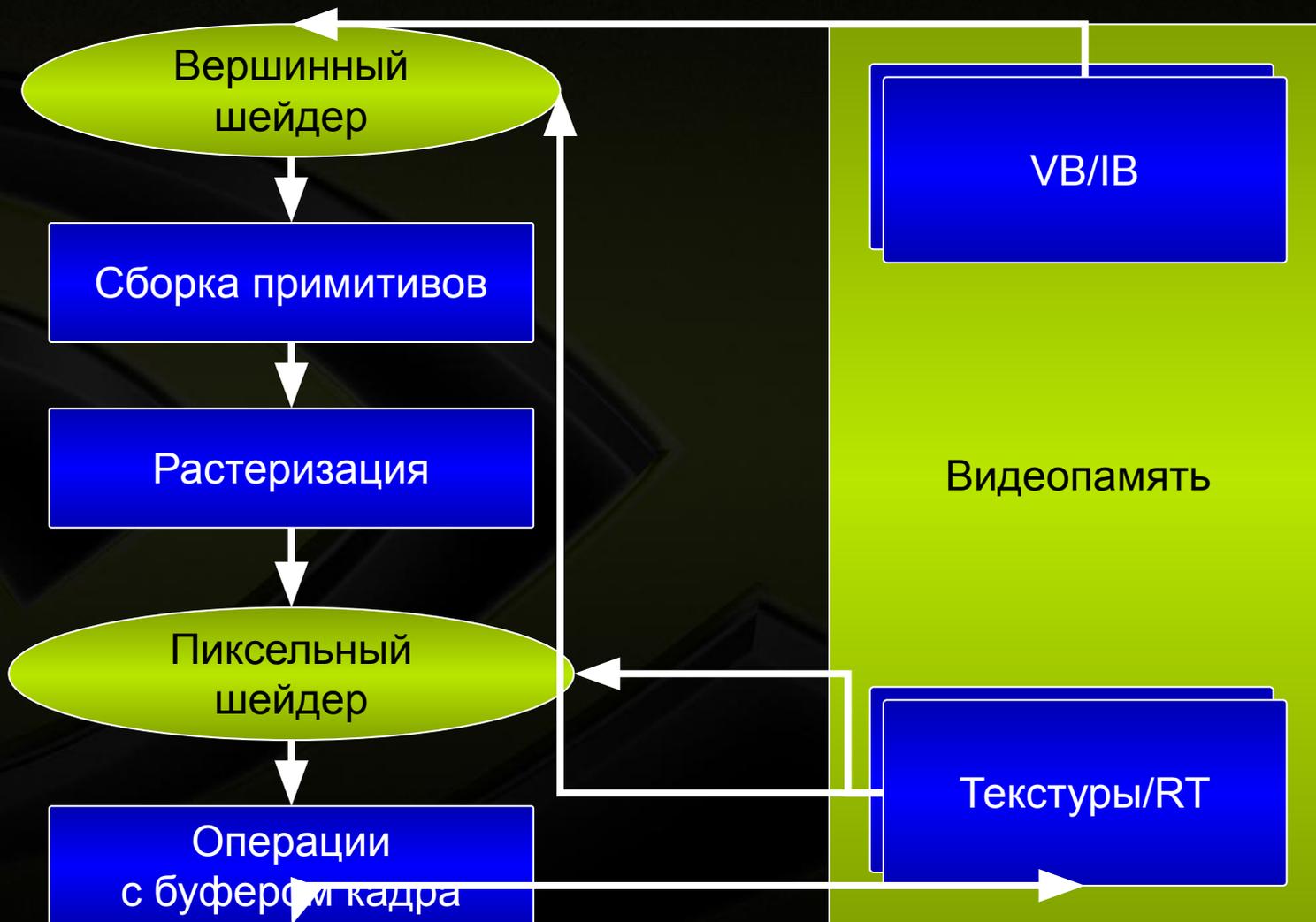
- **D3D10 API**
- **Ресурсы и представления**
- **Инстансинг**
- **Геометрические шейдеры и stream out**
- **Массивы текстур и другие возможности**

10 > 9 !

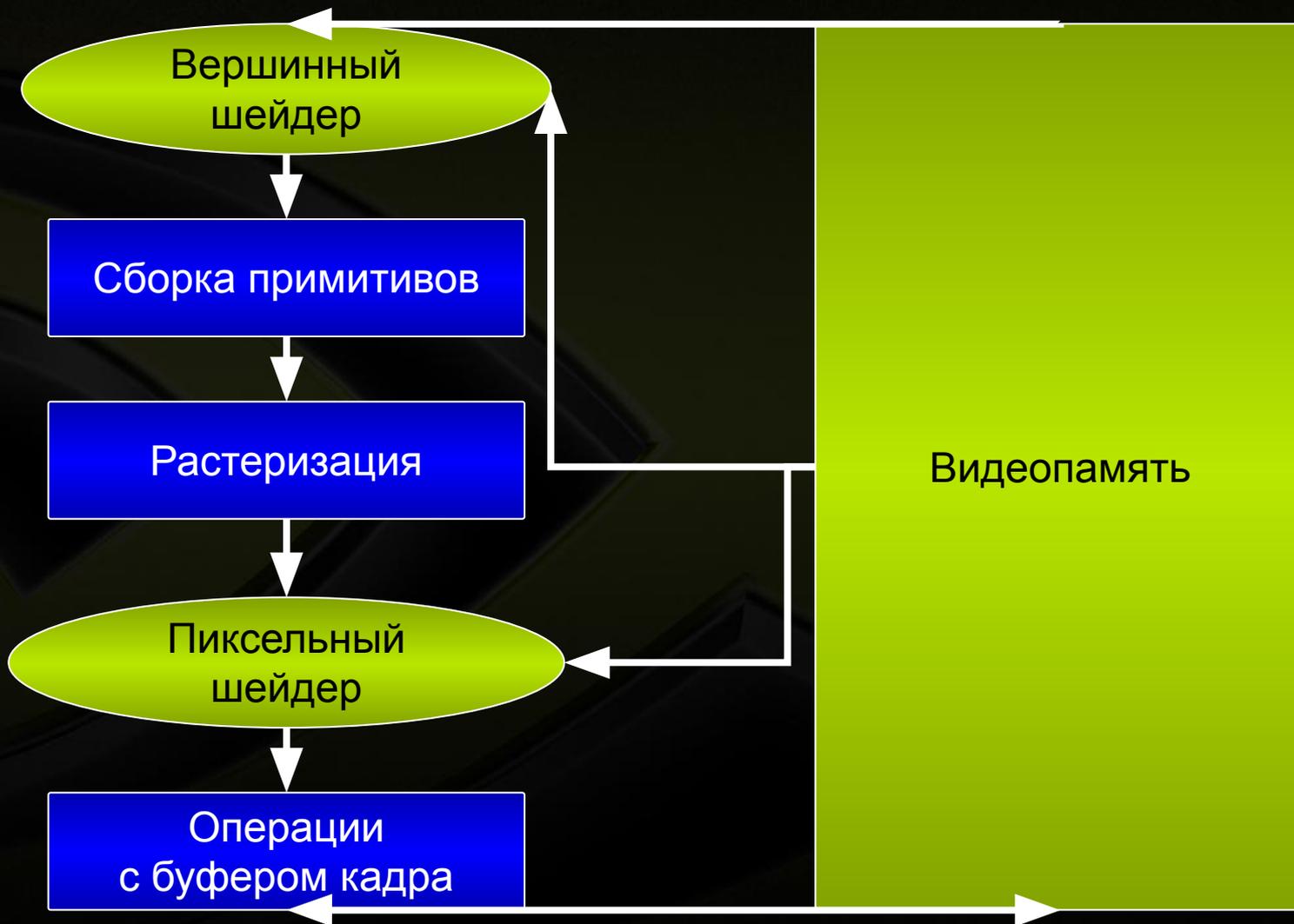


- **Новый, компактный API**
 - Меньше **HRESULT**, больше **void**
 - Стейт-объекты
- **Новая модель драйвера**
 - Позволяет значительно сократить CPU overhead
- **Виртуализация ресурсов**
 - Больше нет lost devices!
- **Новый уровень программируемости**
 - Унифицированные шейдеры (SM 4.0)

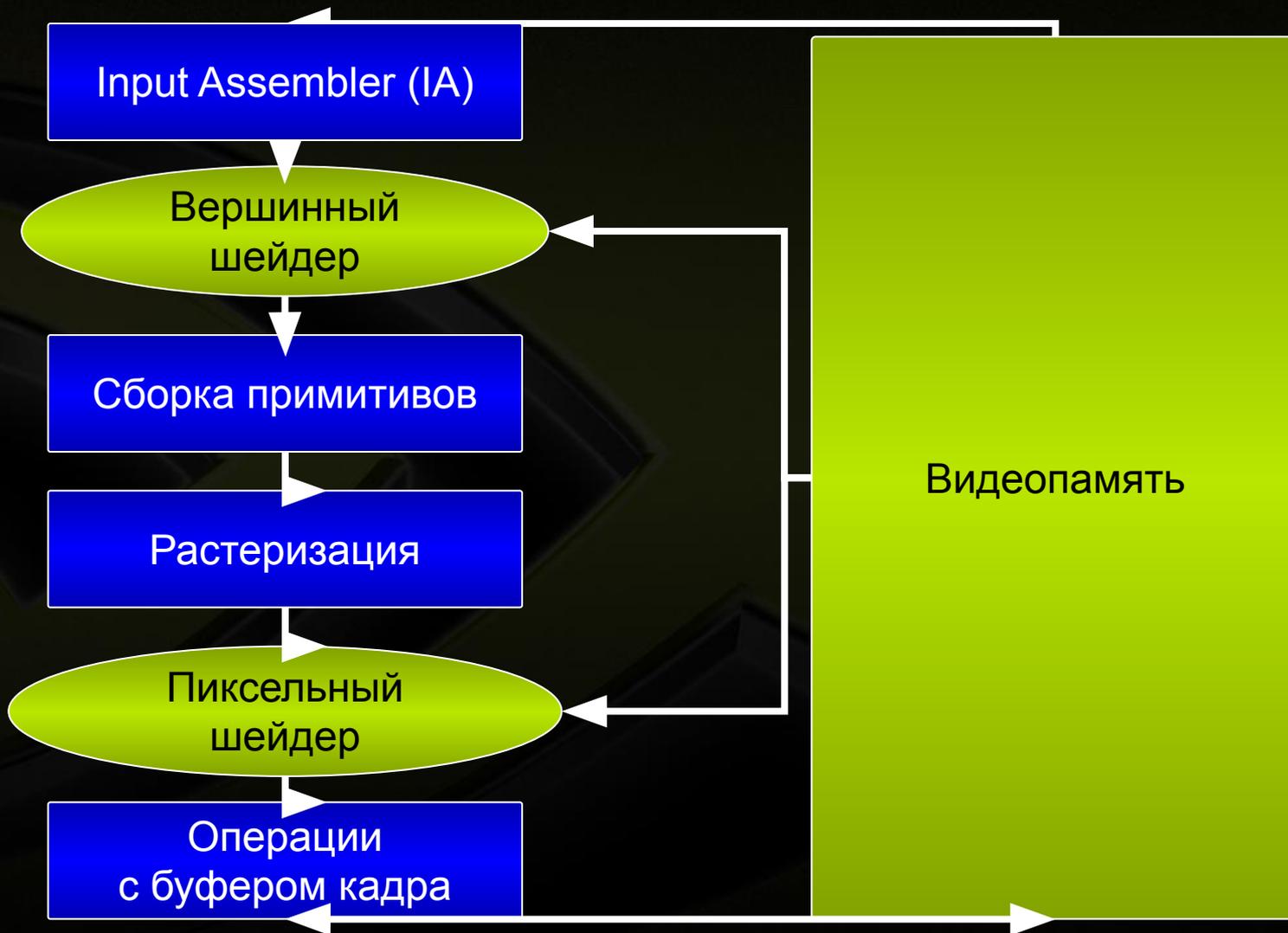
D3D9



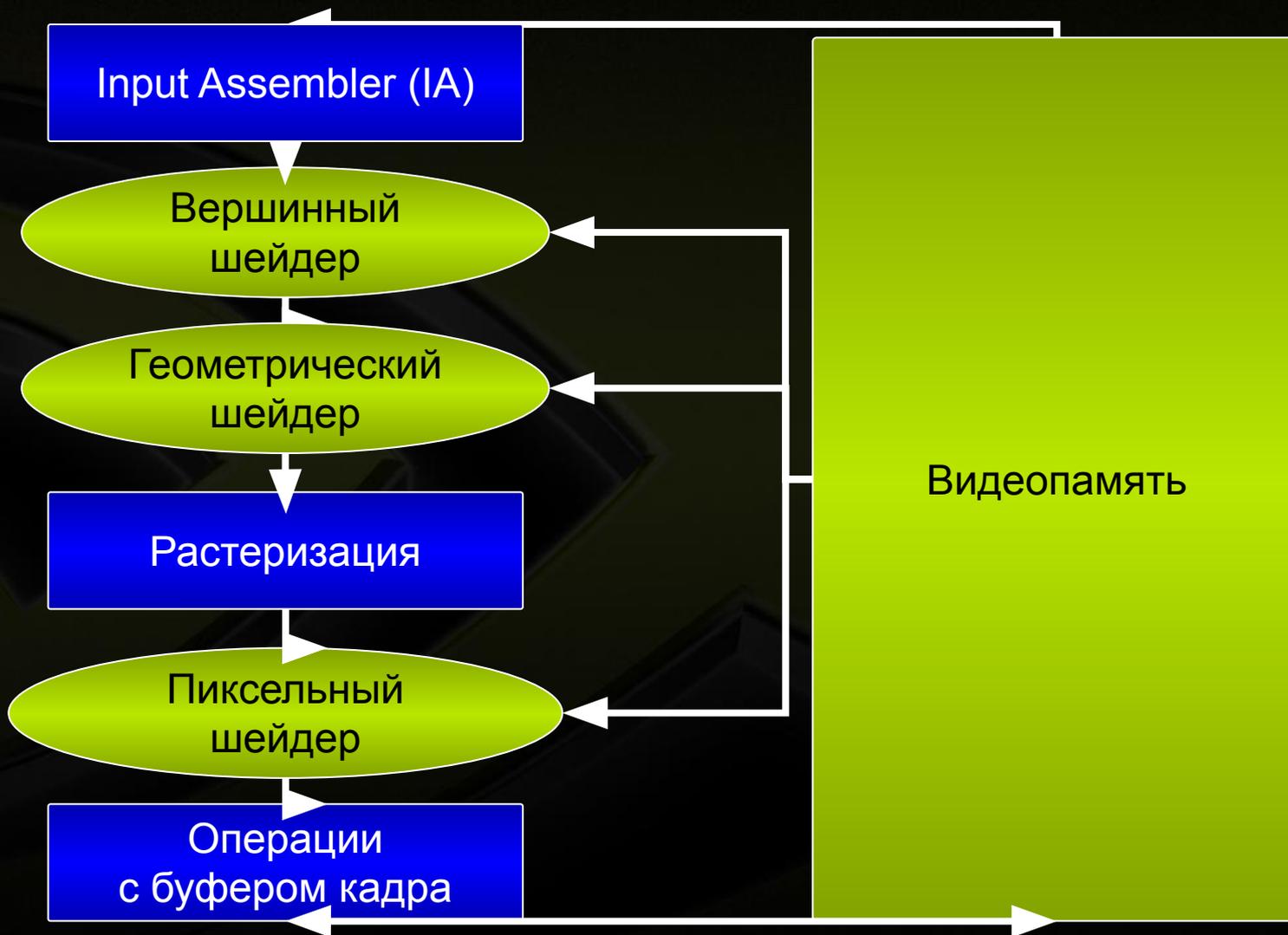
D3D10



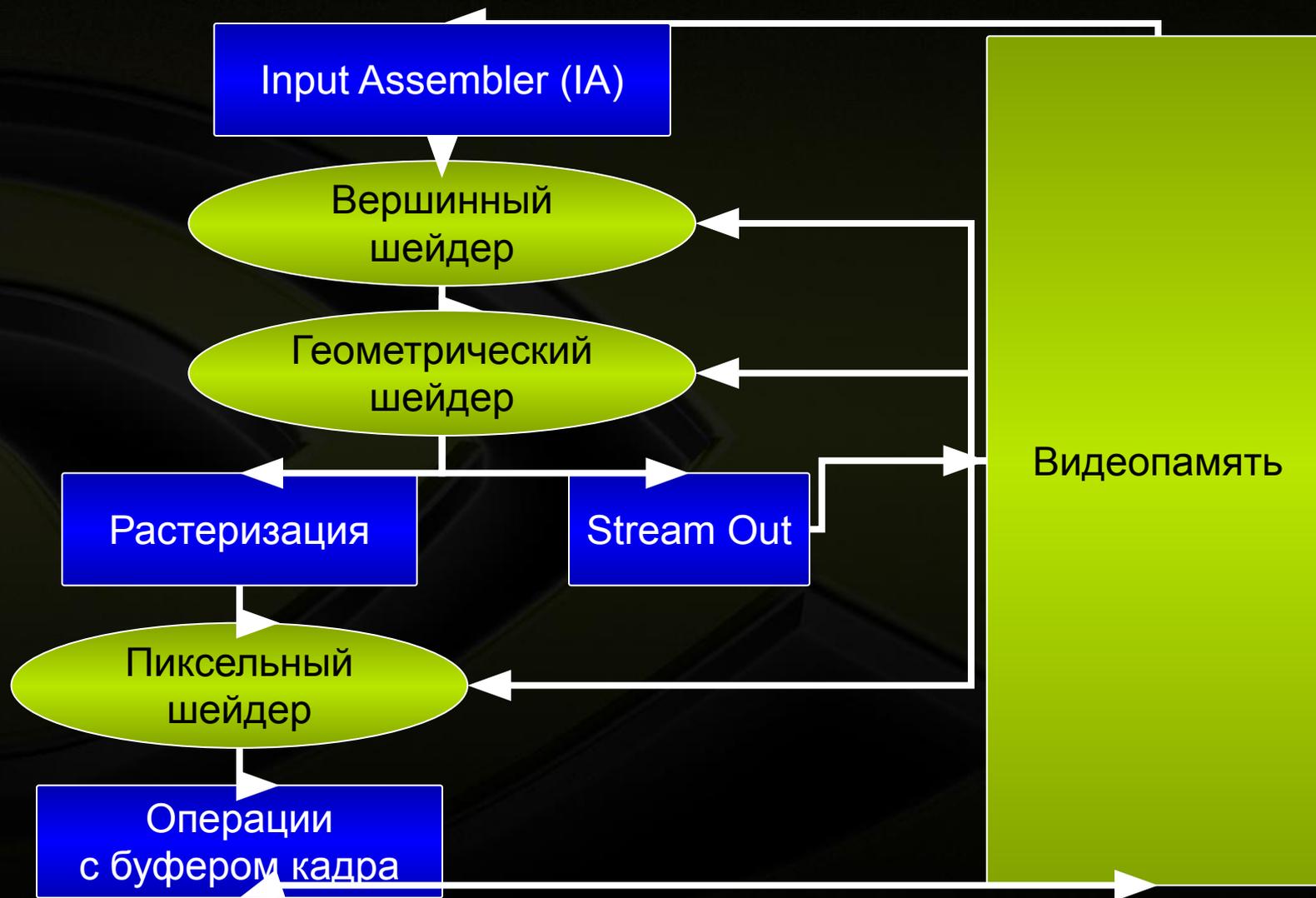
D3D10



D3D10



D3D10



D3D10 API

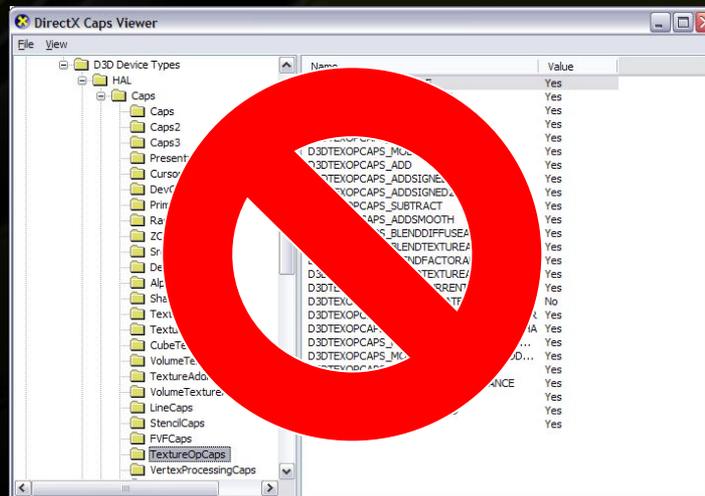


- `ID3D10Device::IA...` □ **I**nter **A**ssembler
- `ID3D10Device::VS...` □ **V**ertex **S**hader
- `ID3D10Device::GS...` □ **G**eometry **S**hader
- `ID3D10Device::SO...` □ **S**tream **O**utput
- `ID3D10Device::RS...` □ **R**asterizer **S**tage
- `ID3D10Device::PS...` □ **P**ixel **S**hader
- `ID3D10Device::OM...` □ **O**utput **M**erger

Забудьте про CapsBits!



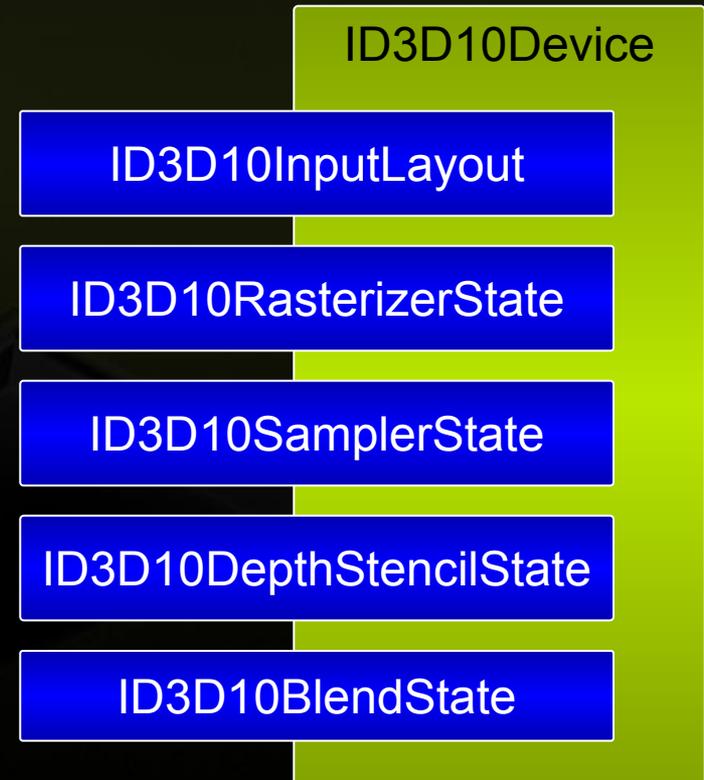
- Вся функциональность всегда присутствует
- Большинство форматов всегда поддерживается
 - есть исключения □ блендинг RGB32F
- Поддержка форматов проверяется вызовом `ID3D10Device::CheckFormatSupport`



Настройка конвейера D3D10



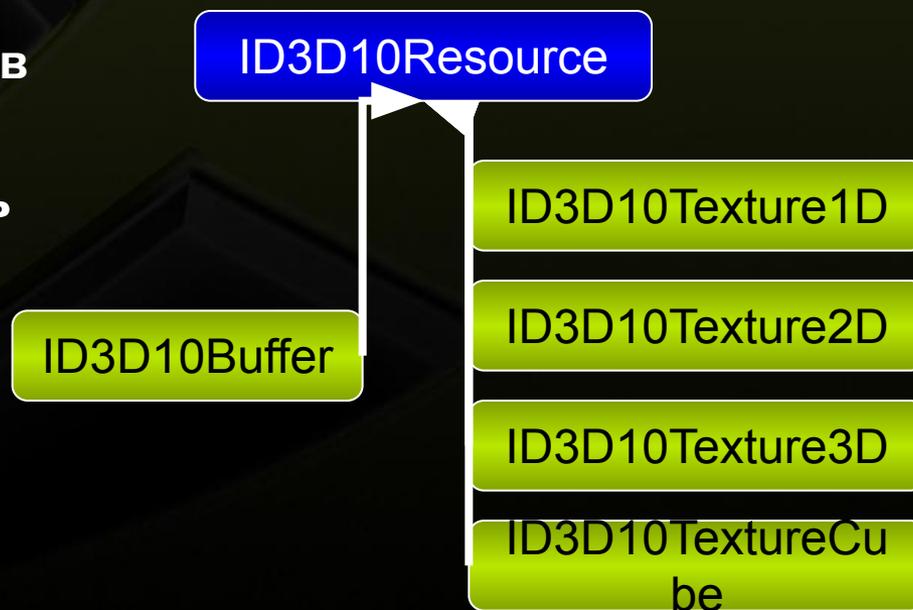
- Настройки конвейера сгруппированы в стейт-объекты
 - Больше нет `SetRenderState (...)`
- Драйвер может эффективно кэшировать состояния конвейера
- 4096 стейт-объектов каждого типа



Ресурсы D3D10



- Буферы
 - ~Последовательный доступ
 - Элементы могут быть разного типа/размера
- Текстуры
 - Произвольный доступ
 - Состоят из суб-ресурсов
- Тип элементов указывать необязательно
- Виртуализированы OS



Типы доступа к ресурсам



- **D3D10_USAGE_DEFAULT**
 - Обновляется редко, маппинг невозможен
- **D3D10_USAGE_DYNAMIC**
 - Обновляется часто, CPU имеет прямой доступ
- **D3D10_USAGE_IMMUTABLE**
 - Никогда не обновляется
- **D3D10_USAGE_STAGING**
 - Используется для получения данных из GPU

Представления (view)



- Позволяют по-разному интерпретировать данные в ресурсах
 - Рендеринг в VB/IB
 - Рендеринг в константы шейдера
 - Использование stencil-буфера как текстуры
- Позволяют получить доступ к суб-ресурсам
 - Рендеринг в отдельные mip-уровни

ID3D10ShaderResourceView

ID3D10RenderTargetView

ID3D10DepthStencilView



Методы Draw() в D3D10



- Draw (...)
- DrawInstanced (...)
- DrawIndexed (...)
- DrawIndexedInstanced (...)
- DrawAuto (...)
 - **Используется вместе с StreamOut**

Инстансинг в D3D10



- **Полноценная поддержка в API**
 - Рендеринг без инстансинга – просто частный случай
- **Другие возможности D3D10 делают инстансинг еще более полезным**
 - Геометрические шейдеры
 - Массивы текстур
 - Stream Out
 - Instance ID

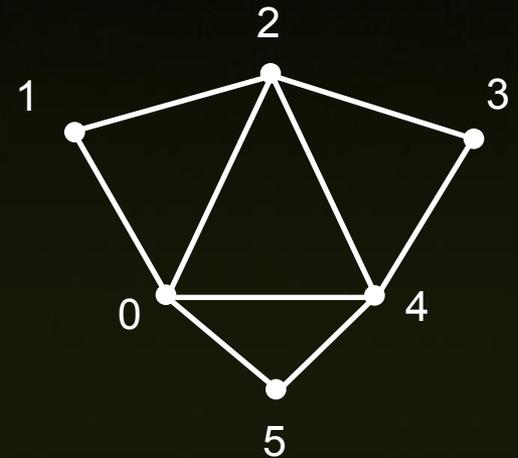
Вершины	
0	$(x_0 \ y_0 \ z_0) \ (n_{x0} \ n_{y0} \ n_{z0})$
1	$(x_1 \ y_1 \ z_1) \ (n_{x1} \ n_{y1} \ n_{z1})$
⋮	...
	$(x_{99} \ y_{99} \ z_{99}) \ (n_{x99} \ n_{y99} \ n_{z99})$

Атрибуты объектов	
0	worldMatrix ₀
1	worldMatrix ₁
⋮	...
	worldMatrix ₄₉

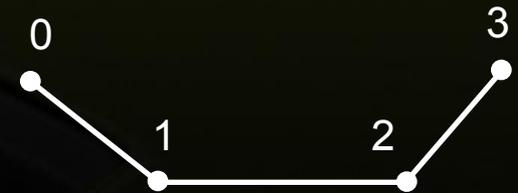
Геометрический шейдер (GS)



- Программируемость на этапе “сборки примитивов”
- Можно создавать новые примитивы!
- Имеет доступ к соседним примитивам
- Может использовать семантику `SV_PrimitiveID`



Triangles with adjacency



Lines with adjacency

Геометрический шейдер (GS)

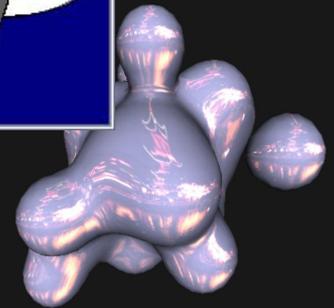
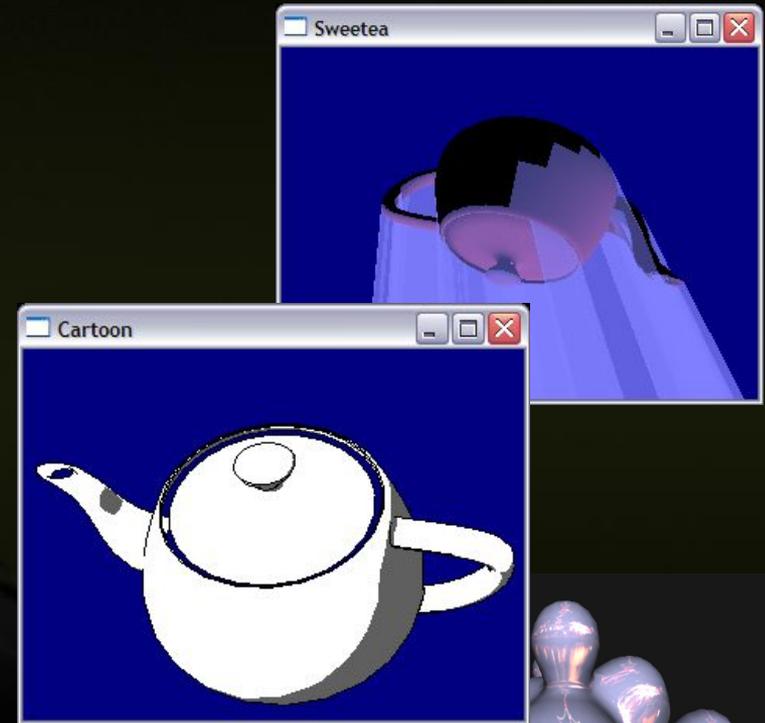


```
[MaxVertexCount(10)]  
void GS_Simple(lineadj VSOut In[4], inout  
    TriangleStream<GSOut> Stream)  
{  
    GSOut v;  
    ...  
    Stream.Append(v);  
    ...  
    Stream.Append(v);  
    ...  
    Stream.RestartStrip();  
    ...  
    Stream.Append(v);  
}
```

Примеры использования GS



- Генерирование shadow volume “на лету”
 - Определяем ребра силуэта, генерируем грани
- Motion Blur
 - Считаем локальные скорости, генерируем дополнительную геометрию
- Изо-поверхности на GPU
 - GDC’06, “Practical Metaballs and Isosurfaces”



Геометрический шейдер (GS)



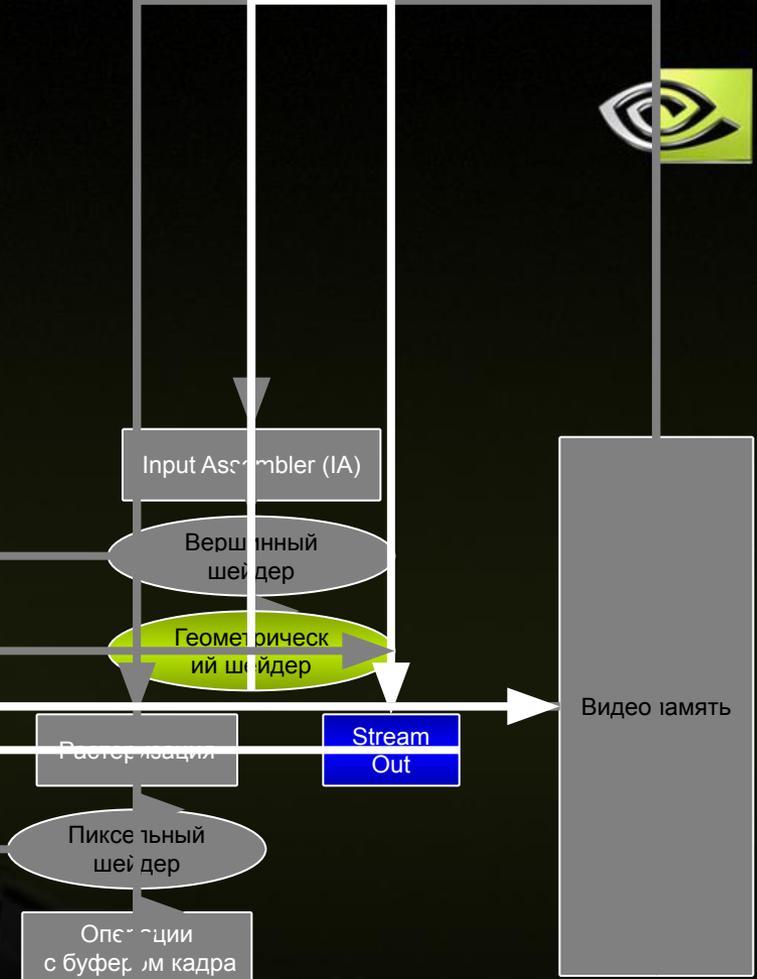
- Количество выходных атрибутов ограничено
 - Максимум - 1024 скалярных атрибута в D3D10
- Не используйте для тесселяции!
- Старайтесь не использовать большие значения **MaxVertexCount**

Stream Out

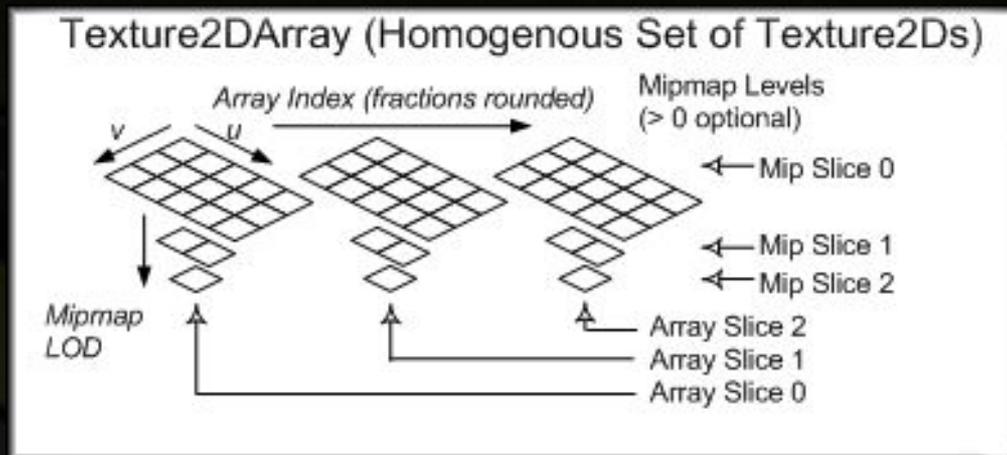
- Возможность записи результатов работы GS в вершинный буфер
 - Или VS если GS отсутствует

- Всегда выводятся *списки примитивов*

- Дает возможность повторно использовать результаты работы GS/VS



Массивы текстур



- Могут быть динамически индексированы из шейдеров
- Текстуры для инстанцируемых объектов
 - **SV_InstanceID** можно использовать в качестве индекса

Рендеринг в массив текстур



- Массивы текстур могут использоваться как RT через `ID3D10RenderTargetView`
- В GS переменная с семантикой `SV_RenderTargetArrayIndex` определяет индекс render-текстуры в массиве
 - Можно использовать для рендеринга в submap в одном прохоже
- Не то же самое, что MRT!
 - Каждый примитив из GS проходит растеризацию и PS

Прощай, Fixed Function!



- Нет тумана...
 - ...Эмулируем в VS/PS
- Нет point-спрайтов...
 - ...Используйте GS для создания геометрии
- Нет плоскостей отсечения...
 - ...Используйте семантику `SV_ClipDistance[n]` в VS/GS
- Нет альфа теста!
 - ...Используйте инструкцию `clip()` в пиксельном шейдере
- См. пример FixedFuncEMU в DX10 SDK

Предикативный рендеринг



- **Условное выполнение вызовов Draw()**
 - *Без остановки конвейера*
- **... по результату occlusion query**
 - *Эффективный occlusion culling*
- **... при переполнении буфера stream out**

Предикативный рендеринг



```
m_pPredicateQuery->Begin();  
//...  
// Здесь рисуем простую геометрию  
//...  
m_pPredicateQuery->End(NULL);  
  
pD3D10Device->SetPredication(m_pPredicateQuery, FALSE);  
//...  
// Здесь рисуем сложную геометрию  
//...  
  
// отключаем предикативный рендеринг  
pD3D10Device->SetPredication(NULL, FALSE);
```

Заключение



- **DX10 – большой шаг вперед в функциональности и гибкости**
- **Начинайте изучать DX10 уже сегодня!**
 - **Документация/примеры в февральском DX9 SDK**

Наши следующие лекции



16:00 – 16:50 **Инструменты для разработчиков от NVIDIA**
Raul Aguaviva/Филипп Герасимов

17:00 – 18:00 **Расчет физики на GPU**
Simon Green

Есть идеи?



- Пишите - yuralsky@nvidia.com !