



*N*VIDIA™

## **Vertex Programs for Fixed Function**

Erik Lindholm  
[erikl@nvidia.com](mailto:erikl@nvidia.com)

# Fixed Function Emulation

---

- Provide vertex shader instructions that can be used to emulate the standard fixed function transform and lighting pipeline.
- **NOTE:** this is a work in progress. A final implementation will follow.



*n*VIDIA™

# Fixed Function Emulation

---

- **Master register mapping**
- **Master template**
- **Master constant map**
- **Extract code of interest**
- **Compress (no re-arrangement required)**
- **Optional performance tuning**
- **Load**



*n*VIDIA™

# Registers

---

- **Most common data stored at top of register file**
- **Feel free to reorganize as you see fit**
- **You have 12 registers**



*n*VIDIA™

# Master register mapping

---

- R0                      scratch
- R1                      scratch
- R2                      scratch
- R3 (Rd)                scratch/light vector
- R4 (Rr,Rf,Rl)        scratch/reflection vector/eye vector/light ve
- R5 (Rs,Rv)            scratch/sphere vector/eye vector
- R6 (Rx)                scratch/specular color
- R7 (Rc)                scratch/diffuse color
- R8 (RH)                scratch/half angle vector
- R9 (Rh)                eye position homogeneous
- R10 (Re)              eye position non-homogeneous
- R11 (Rn)              eye normal



NVIDIA™

# Program

---

- **Code is organized into blocks**
- **1. Transform/skinning**
- **2. Fog**
- **3. Point parameters**
- **4. Lighting**
- **5. Texture**
- **Replicate blocks for more lights or textures**



*n*VIDIA™

# Normal Transform Eye Space

```
/* eye space normal */
```

```
DP3 Rn.x,v[NRML],c[NORMAL0_MATRIX_X];
```

```
DP3 Rn.y,v[NRML],c[NORMAL0_MATRIX_Y];
```

```
DP3 Rn.z,v[NRML],c[NORMAL0_MATRIX_Z];
```

```
/* eye space normal 2 matrix skinning */
```

```
DP3 R2.x,v[NRML],c[NORMAL1_MATRIX_X];
```

```
DP3 R2.y,v[NRML],c[NORMAL1_MATRIX_Y];
```

```
DP3 R2.z,v[NRML],c[NORMAL1_MATRIX_Z];
```

```
/* eye space normal 3 matrix skinning */
```

```
DP3 R4.x,v[NRML],c[NORMAL2_MATRIX_X];
```

```
DP3 R4.y,v[NRML],c[NORMAL2_MATRIX_Y];
```

```
DP3 R4.z,v[NRML],c[NORMAL2_MATRIX_Z];
```

```
/* eye space normal 4 matrix skinning */
```

```
DP3 R6.x,v[NRML],c[NORMAL3_MATRIX_X];
```

```
DP3 R6.y,v[NRML],c[NORMAL3_MATRIX_Y];
```

```
DP3 R6.z,v[NRML],c[NORMAL3_MATRIX_Z];
```



NVIDIA

# Position Transform Eye Space

```
/* eye space position homogeneous */
DP4 Rh.x,v[OPOS],c[MODELVIEW0_MATRIX_X];
DP4 Rh.y,v[OPOS],c[MODELVIEW0_MATRIX_Y];
DP4 Rh.z,v[OPOS],c[MODELVIEW0_MATRIX_Z];
DP4 Rh.w,v[OPOS],c[MODELVIEW0_MATRIX_W];

/* eye space position homogeneous 2 matrix skinning */
DP4 R3.x,v[OPOS],c[MODELVIEW1_MATRIX_X];
DP4 R3.y,v[OPOS],c[MODELVIEW1_MATRIX_Y];
DP4 R3.z,v[OPOS],c[MODELVIEW1_MATRIX_Z];
DP4 R3.w,v[OPOS],c[MODELVIEW1_MATRIX_W];

/* eye space position homogeneous 3 matrix skinning */
DP4 R5.x,v[OPOS],c[MODELVIEW2_MATRIX_X];
DP4 R5.y,v[OPOS],c[MODELVIEW2_MATRIX_Y];
DP4 R5.z,v[OPOS],c[MODELVIEW2_MATRIX_Z];
DP4 R5.w,v[OPOS],c[MODELVIEW2_MATRIX_W];

/* eye space position homogeneous 4 matrix skinning */
DP4 R7.x,v[OPOS],c[MODELVIEW3_MATRIX_X];
DP4 R7.y,v[OPOS],c[MODELVIEW3_MATRIX_Y];
DP4 R7.z,v[OPOS],c[MODELVIEW3_MATRIX_Z];
DP4 R7.w,v[OPOS],c[MODELVIEW3_MATRIX_W];
```



nVIDIA™



# Skinning: Blend Weights

---

```
MOV R0,v[WGHT];  
MOV R0.w,c[CONSTANT0].z;          /* if need new weight */  
DP4 R0.y,R0,c[CONSTANT0].xyyz;    /* new 2nd weight */  
DP4 R0.z,R0,c[CONSTANT0].xxyz;    /* or new 3rd weight */  
DP4 R0.w,R0,c[CONSTANT0].xxxz;    /* or new 4th weight */
```



NVIDIA™

# Skinning: Position/Normal Blend

```
MUL Rh,R0.x,Rh;      /* position */
MUL Rn,R0.x,Rn;      /* normal */
MAD Rh,R0.y,R3,Rh;    /* if 2+ matrix skinning */
MAD Rn,R0.y,R2,Rn;    /* if 2+ matrix skinning */
MAD Rh,R0.z,R5,Rh;    /* if 3+ matrix skinning */
MAD Rn,R0.z,R4,Rn;    /* if 3+ matrix skinning */
MAD Rh,R0.w,R7,Rh;    /* if 4 matrix skinning */
MAD Rn,R0.w,R6,Rn;    /* if 4 matrix skinning */
```



NVIDIA™

## Skinning: Position Output

---

DP4 ○ [HPOS] .x, Rh, c [PROJECTION\_MATRIX\_X] ;

DP4 ○ [HPOS] .y, Rh, c [PROJECTION\_MATRIX\_Y] ;

DP4 ○ [HPOS] .z, Rh, c [PROJECTION\_MATRIX\_Z] ;

DP4 ○ [HPOS] .w, Rh, c [PROJECTION\_MATRIX\_W] ;



NVIDIA™

## Position Output (No Skinning)

---

```
DP4 o[HPOS].x,v[OPOS],c[COMPOSITE_MATRIX_X];  
DP4 o[HPOS].y,v[OPOS],c[COMPOSITE_MATRIX_Y];  
DP4 o[HPOS].z,v[OPOS],c[COMPOSITE_MATRIX_Z];  
DP4 o[HPOS].w,v[OPOS],c[COMPOSITE_MATRIX_W];
```



NVIDIA™

# Normalize Eye Normal

---

DP3 Rn.w, Rn, Rn;

RSQ Rn.w, Rn.w;

MUL Rn, Rn, Rn.w;



NVIDIA™

# Non-homogeneous Eye Position

---

RCP  $R0.w, Rh.w;$

MUL  $Re, Rh, R0.w;$



nVIDIA™

## Vertex->Eye Vectors

---

```
ADD  R0, -Re, c[EYE_POSITION];  
DP3  R0.w, R0, R0;  
RSQ  R1.w, R0.w;  
MUL  Rv, R0, R1.w;           /* direction */  
DST  Rf, R0.w, R1.w;         /* distance */
```



NVIDIA™

# Fog Output

---

```
/* radial fog */
```

```
MOV o[FOGC].x,Rf.y;
```

```
/* z linear fog */
```

```
MOV o[FOGC].x,-Re.z;
```



NVIDIA™



# Point Parameters

---

```
DP3  R0.w,Rf,c[POINT_PARAMETER_ATTENUATION];  
RSQ  R0.w,R0.w;  
MUL  R0.w,R0.w,c[POINT_PARAMETER].x;  
MIN  R0.w,R0.w,c[POINT_PARAMETER].y;  
MAX  o[PSIZ].x,R0.w,c[POINT_PARAMETER].z;
```



NVIDIA™

# Lighting Initialization

---

```
/* diffuse only */  
MOV Rc,c[GLOBAL_ILLUMINATION];
```

```
/* diffuse and specular */  
MOV Rc,c[GLOBAL_ILLUMINATION];  
MOV Rx,c[CONSTANT0].y;
```



NVIDIA<sup>™</sup>

# Infinite Light/Infinite Viewer

---

```
DP3  R0.x,Rn,c[LIGHT_POSITION] ;
DP3  R0.y,Rn,c[LIGHT_HALF_ANGLE_VECTOR] ;
MOV  R0.w,c[LIGHT_SPECULAR].w;
LIT  R0,R0;
MAD  Rc.xyz,R0.x,c[LIGHT_AMBIENT],Rc;
MAD  Rc.xyz,R0.y,c[LIGHT_DIFFUSE],Rc;
MAD  Rx.xyz,R0.z,c[LIGHT_SPECULAR],Rx;
/* use Rc above for single color */
```



NVIDIA

## Spotlight/Local Viewer 1/4

---

```
/* light direction/distance vectors */  
ADD R0, -Re, c[LIGHT_POSITION];  
DP3 R0.w, R0, R0;  
RSQ R1.w, R0.w;  
MUL R1, R0, R1.w;           /* direction */  
DST Rd, R0.w, R1.w;         /* distance */
```



nVIDIA™

## Spotlight/Local Viewer 2/4

---

```
/* half-angle vector */  
ADD RH,Rv,Rl;  
DP3 RH.w,RH,RH;  
RSQ RH.w,RH.w;  
MUL RH,RH,RH.w;
```



nVIDIA™

## Spotlight/Local Viewer 3/4

---

```
/* distance attenuation */  
DP3  Rd.w,Rd,c[LIGHT_ATTENUATION];  
RCP  Rd.w,Rd.w;  
  
/* spotlight cone attenuation */  
DP3  R0.y,R1,-c[LIGHT_SPOT_DIRECTION];  
ADD  R0.x,R0.y,-c[LIGHT_SPOT_DIRECTION].w;  
MOV  R0.w,c[LIGHT_ATTENUATION].w;  
LIT  R0,R0;  
MUL  Rd,Rd.w,R0.z;
```



NVIDIA

## Spotlight/Local Viewer 4/4

---

```
DP3  R0.x,Rn,Rl;  
DP3  R0.y,Rn,RH;  
MOV  R0.w,c[LIGHT_SPECULAR].w;  
LIT  R0,R0;  
MUL  R0,R0,Rd.w;  
MAD  Rc.xyz,R0.x,c[LIGHT_AMBIENT],Rc;  
MAD  Rc.xyz,R0.y,c[LIGHT_DIFFUSE],Rc;  
MAD  Rx.xyz,R0.z,c[LIGHT_SPECULAR],Rx;  
/* use Rc above for single color */
```



NVIDIA

# Lighting Output

---

```
/* diffuse only */
```

```
MOV o[COL0],Rc;
```

```
/* diffuse and specular */
```

```
MOV o[COL0],Rc;
```

```
MOV o[COL1],Rx;
```



NVIDIA™



## Global Texture Generation State 1/2

---

```
/* reflection vector */  
MUL R0,Rn,c[EYE_POSITION].w;  
DP3 Rr.w,Rn,Rv;  
MAD Rr,Rr.w,R0,-Rv;
```



NVIDIA™

## Global Texture Generation State 2/2

---

```
/* sphere map vector */  
ADD R0,c[CONSTANT0].yyzy,Rr;  
DP3 R0.w,R0,R0;  
RSQ R0.w,R0.w;  
MUL R0.xyz,R0,c[CONSTANT0].wwyy;  
MAD Rs,R0.w,R0,c[CONSTANT0].wwyy;
```



NVIDIA<sup>™</sup>

## Per Texture 1/4

---

```
/* initialize */  
MOV R0,v[TEX0];
```



nVIDIA™

## Per Texture 2/4

---

```
/* object space plane */  
DP4 R0.x,v[OPOS],c[TEXTURE_OBJECT_PLANE_X];  
DP4 R0.y,v[OPOS],c[TEXTURE_OBJECT_PLANE_Y];  
DP4 R0.z,v[OPOS],c[TEXTURE_OBJECT_PLANE_Z];  
DP4 R0.w,v[OPOS],c[TEXTURE_OBJECT_PLANE_W];
```

```
/* eye space plane */  
DP4 R0.x,Rh,c[TEXTURE_EYE_PLANE_X];  
DP4 R0.y,Rh,c[TEXTURE_EYE_PLANE_Y];  
DP4 R0.z,Rh,c[TEXTURE_EYE_PLANE_Z];  
DP4 R0.w,Rh,c[TEXTURE_EYE_PLANE_W];
```



nVIDIA™

## Per Texture 3/4

---

```
/* sphere map */
```

```
MOV R0.xy, Rs;
```

```
/* normal vector */
```

```
MOV R0.xyz, Rn;
```

```
/* reflection vector */
```

```
MOV R0.xyz, Rr;
```



NVIDIA™

## Per Texture 4/4

---

```
/* texture matrix and output */  
DP4 o[TEX0].x,R0,c[TEXTURE_MATRIX_X];  
DP4 o[TEX0].y,R0,c[TEXTURE_MATRIX_Y];  
DP4 o[TEX0].z,R0,c[TEXTURE_MATRIX_Z];  
DP4 o[TEX0].w,R0,c[TEXTURE_MATRIX_W];
```



NVIDIA™

# Constants

---

- **Constants required by the template code is organized into blocks by functionality.**
- **Feel free to reorganize as you see fit**
- **You have 96 locations**



**nVidia**™

# Constant Map 0-11

---

```
/* normal matrices */  
c[NORMAL0_MATRIX_X] /* inverse transpose Modelview Matrix */  
c[NORMAL0_MATRIX_Y]  
c[NORMAL0_MATRIX_Z]  
c[NORMAL1_MATRIX_X] /* inverse transpose Modelview1 Matrix */  
c[NORMAL1_MATRIX_Y]  
c[NORMAL1_MATRIX_Z]  
c[NORMAL2_MATRIX_X] /* inverse transpose Modelview2 Matrix */  
c[NORMAL2_MATRIX_Y]  
c[NORMAL2_MATRIX_Z]  
c[NORMAL3_MATRIX_X] /* inverse transpose Modelview3 Matrix */  
c[NORMAL3_MATRIX_Y]  
c[NORMAL3_MATRIX_Z]
```



NVIDIA™



# Constant Map 12-27

```
/* position matrices */  
c[MODELVIEW0_MATRIX_X]  
c[MODELVIEW0_MATRIX_Y]  
c[MODELVIEW0_MATRIX_Z]  
c[MODELVIEW0_MATRIX_W]  
c[MODELVIEW1_MATRIX_X]  
c[MODELVIEW1_MATRIX_Y]  
c[MODELVIEW1_MATRIX_Z]  
c[MODELVIEW1_MATRIX_W]  
c[MODELVIEW2_MATRIX_X]  
c[MODELVIEW2_MATRIX_Y]  
c[MODELVIEW2_MATRIX_Z]  
c[MODELVIEW2_MATRIX_W]  
c[MODELVIEW3_MATRIX_X]  
c[MODELVIEW3_MATRIX_Y]  
c[MODELVIEW3_MATRIX_Z]  
c[MODELVIEW3_MATRIX_W]
```

```
/* Modelview Matrix */
```

```
/* Modelview1 Matrix */
```

```
/* Modelview2 Matrix */
```

```
/* Modelview3 Matrix */
```



nVIDIA™

## Constant Map 28-35

---

```
c [PROJECTION_MATRIX_X] /* Projection Matrix */  
c [PROJECTION_MATRIX_Y]  
c [PROJECTION_MATRIX_Z]  
c [PROJECTION_MATRIX_W]
```

```
c [COMPOSITE_MATRIX_X] /* Projection*Modelview Matrix */  
c [COMPOSITE_MATRIX_Y]  
c [COMPOSITE_MATRIX_Z]  
c [COMPOSITE_MATRIX_W]
```



nVIDIA™

## Constant Map 36-39

---

```
c [TEXTURE_MATRIX_X] /* Texture Matrix */  
c [TEXTURE_MATRIX_Y]  
c [TEXTURE_MATRIX_Z]  
c [TEXTURE_MATRIX_W]
```



nVIDIA™

## Constant Map 40-47

---

```
/* light state */
c[GLOBAL_ILLUMINATION]      /* R,G,B,A (emission+global ambient) */
c[LIGHT_POSITION]           /* X,Y,Z,NA */
c[LIGHT_HALF_ANGLE_VECTOR] /* X,Y,Z,NA for infinite light+viewer */
c[LIGHT_AMBIENT]            /* R,G,B,NA (light*material) */
c[LIGHT_DIFFUSE]            /* R,G,B,NA (light*material) */
c[LIGHT_SPECULAR]          /* R,G,B,SPECULAR POWER (light*material) */
c[LIGHT_ATTENUATION]        /* K0,K1,K2,SPOT POWER */
c[LIGHT_SPOT_DIRECTION]    /* X,Y,Z,cos(CUTOFF) */
```



nVIDIA™

## Constant Map 48-49

---

```
/* point parameters */  
c[POINT_PARAMETER]      /* POINT_SIZE,MAX_CLAMP,MIN_CLAMP,NA */  
c[POINT_PARAMETER_ATTENUATION] /* K0,K1,K2,NA */
```



nVIDIA™

## Constant Map 50-57

---

```
/* texgen planes */
c [TEXTURE_OBJECT_PLANE_X]          /* X,Y,Z,W */
c [TEXTURE_OBJECT_PLANE_Y]          /* X,Y,Z,W */
c [TEXTURE_OBJECT_PLANE_Z]          /* X,Y,Z,W */
c [TEXTURE_OBJECT_PLANE_W]          /* X,Y,Z,W */
c [TEXTURE_EYE_PLANE_X]              /* X,Y,Z,W */
c [TEXTURE_EYE_PLANE_Y]              /* X,Y,Z,W */
c [TEXTURE_EYE_PLANE_Z]              /* X,Y,Z,W */
c [TEXTURE_EYE_PLANE_W]              /* X,Y,Z,W */
```



nVIDIA™

## Constant Map 58-59

---

```
/* miscellaneous */  
c[EYE_POSITION] /* 0.0,0.0,0.0,2.0 */  
c[CONSTANT0] /* -1.0,0.0,1.0,0.5 */
```



NVIDIA™

# Constant Map 60-95

---

- **More Lights**
- **More Textures**
- **More Matrices**
- **More ???**



*n*VIDIA™