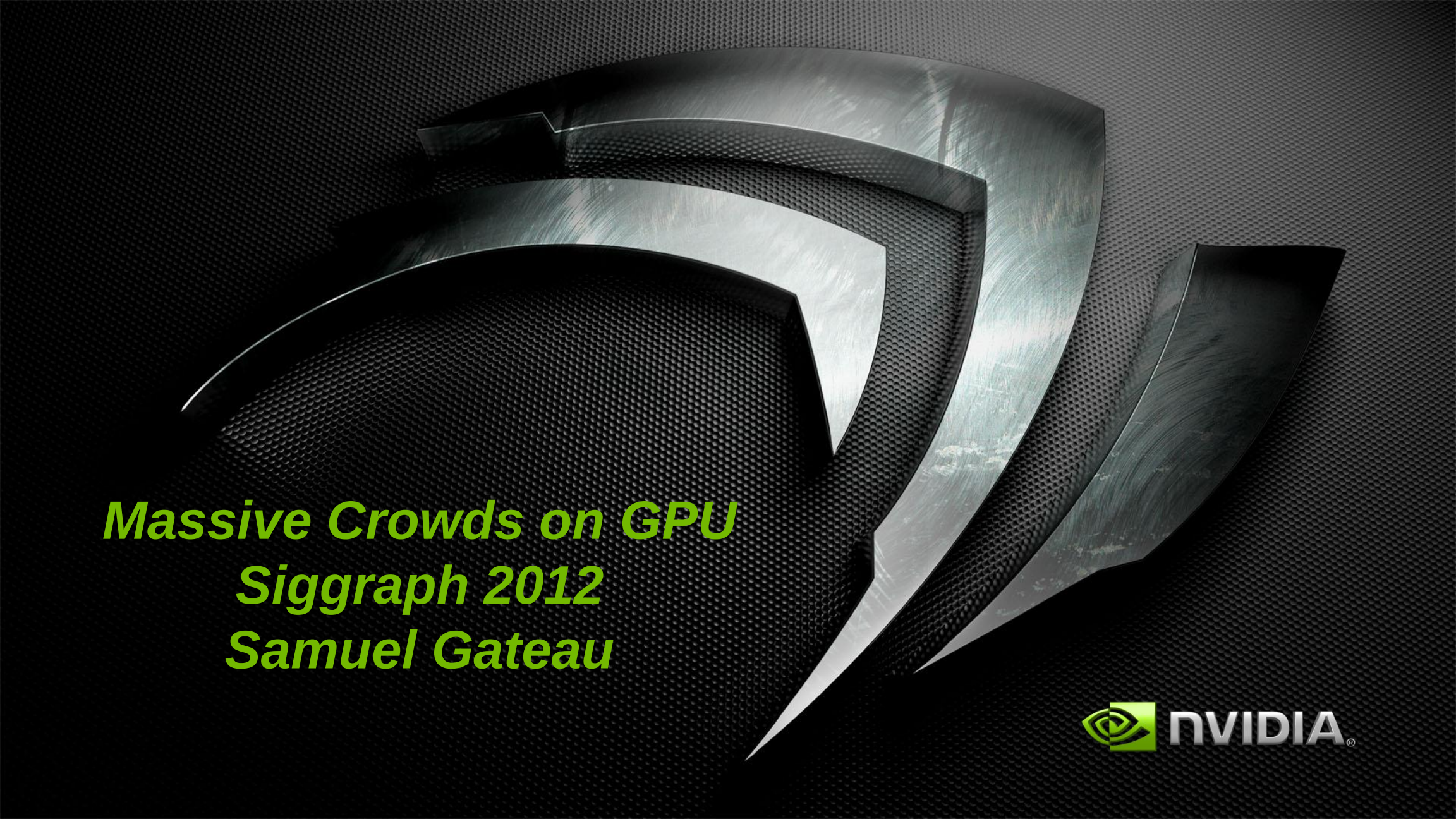

Massive Crowds on GPU
Siggraph 2012
Samuel Gateau
NVIDIA

Context: Autodesk Project Geppetto

Autodesk People Power Geppetto for 3DS Max

http://labs.autodesk.com/utilities/geppetto/

Project Geppetto Preview

Project Geppetto assembles all the components you'd need to create, manage and control large crowds of characters. Our research goal is to solve three big problems:

1. Creating believable motion easily
2. Creating the look of any kind of human population or demographic and allowing for cultural influences
3. Creating an efficient framework for interacting with tens of thousands of such characters

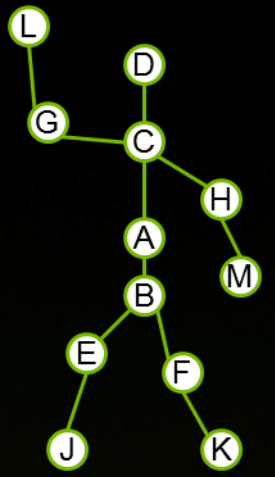
Context: NVIDIA Project Goals

- Explore ways to accelerate the Geppetto project using the GPU as much as possible
- Accelerate overall crowd rendering performances for real-time interaction and editing
  - Execute all the rendering processing steps efficiently on the gpu
  - Prove scaling
- Optimize the data representation in memory
  - For best parallel computation performances
  - To minimize the footprint

Rendering characters

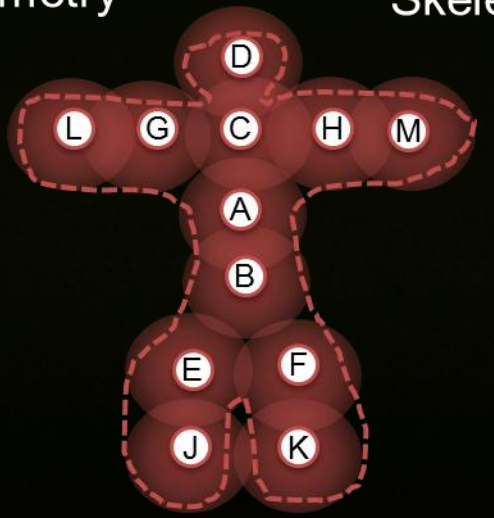
- A Character is a regular 3D model Geometry plus…
  - Skin
  - Skeleton
  - Repertoire
    - Clip library
- Every frame for every characters
  - Animation of the Skeleton
  - Skinning the Geometry
  - Drawing the Skinned Geometry

Geometry



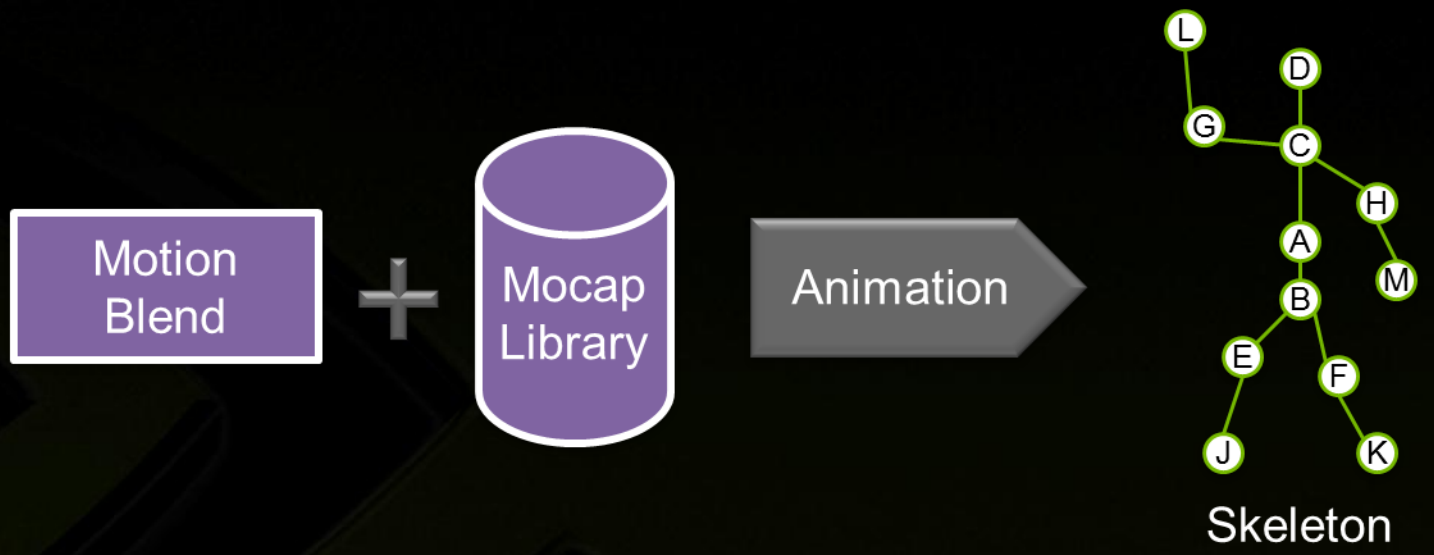
Skeleton

Skinned Geometry



Skin

Animation: Update Skeleton Pose



Update the Skeleton pose from a library of Mocaps (motion capture) and the Motion Blend parameters for the current frame

Animation steps

Steps	Details	Geppetto	Prototype
Animate Motion Blend	Standard frame animation for the motion blend parameters for the current frame and character	CPU	CPU

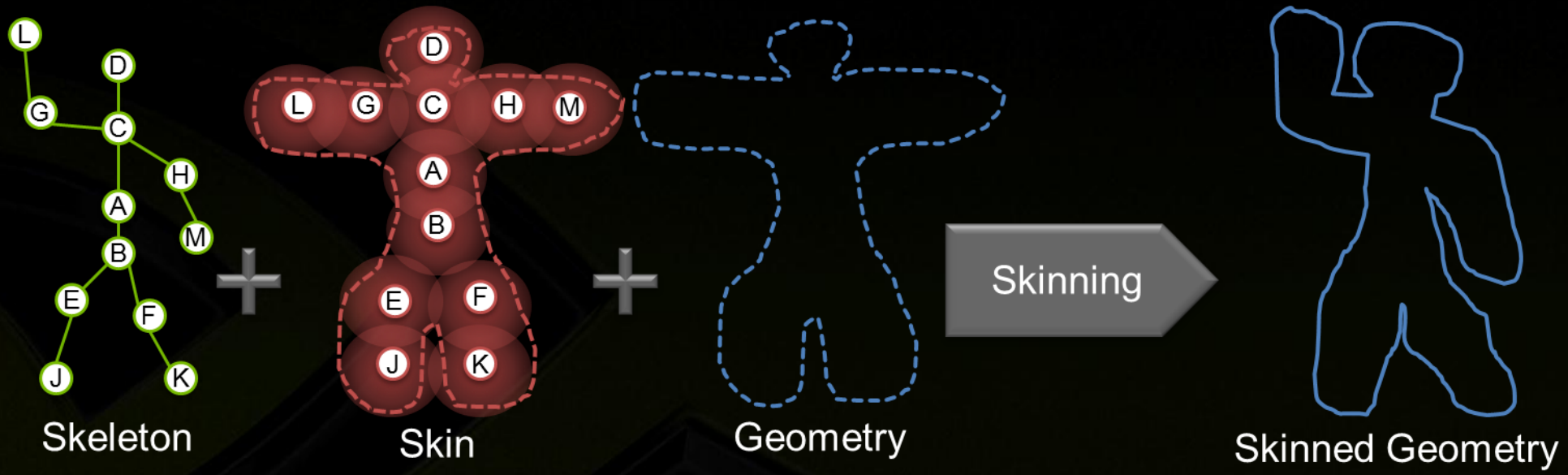
Animation steps

Steps	Details	Geppetto	Prototype
Animate Motion Blend	Standard frame animation for the motion blend parameters for the current frame and character	CPU	CPU
Evaluate Skeleton Local Pose	For each joint: Blend the joint values coming from several mocaps (up to 4)	CPU	GPU

Animation steps

Steps	Details	Geppetto	Prototype
Animate Motion Blend	Standard frame animation for the motion blend parameters for the current frame and character	CPU	CPU
Evaluate Skeleton Local Pose	For each joint: Blend the joint values coming from several mocaps (up to 4)	CPU	GPU
Evaluate Skeleton Absolute Pose	For each joint: Accumulate transformation from joint to the skeleton root	CPU	GPU

Skinning: Update Geometry Surface



For every vertex of the geometry, displace vertex position & tangent space according to the Skin weighting and Skeleton pose

Linear blending of deformed position

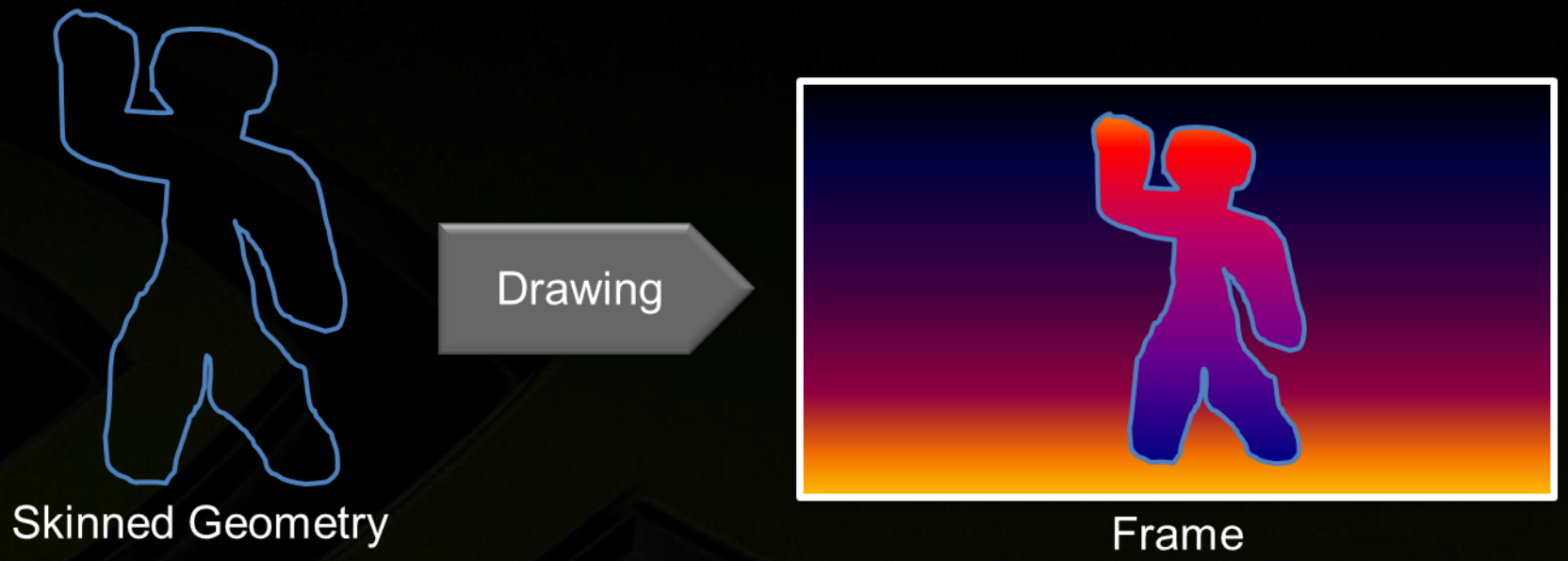
Skinning steps

Steps	Details	Geppetto	Prototype
Evaluate Bind-Pose Matrices	For each Skin Cluster: Combine the joint pose matrix with the skin binding matrix	CPU	GPU

Skinning steps

Steps	Details	Geppetto	Prototype
Evaluate Bind-Pose Matrices	For each Skin Cluster: Combine the joint pose matrix with the skin binding matrix	CPU	GPU
Evaluate Skinned Surface	For each vertex: For each influencing cluster: Blend the deformed position (and tangent space)	CPU Update a skinned vertex buffer cache	GPU Skinning is happening in the rendering pass

Drawing: Display skinned geometry

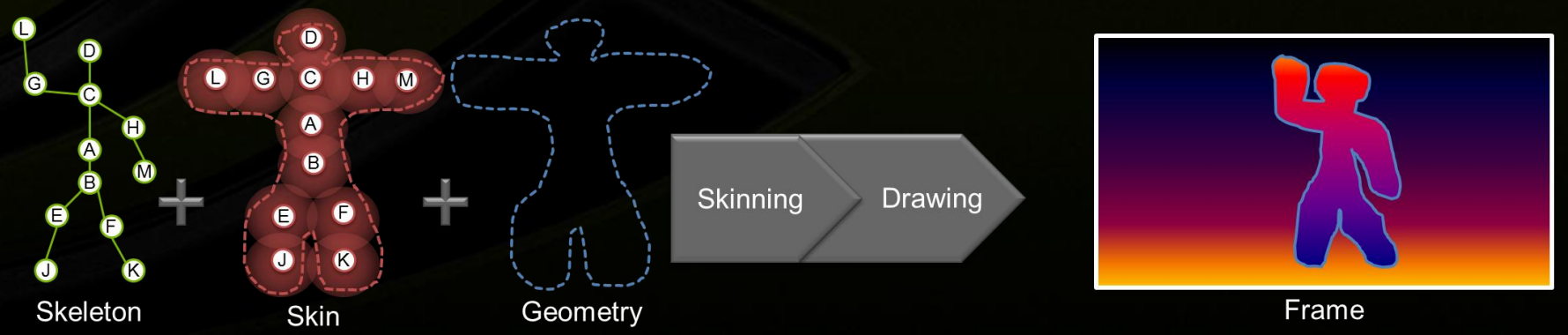


Display the skinned geometry with the material properties at the correct location in the frame

- Shadow pass
- Beauty pass

Drawing step

Steps	Details	Geppetto	Prototype
Drawing skinned surface	Once we provide the skinned position and tangent space, follow the standard Vertex Shader pipeline	GPU Display the skinned vertex buffer	GPU Include skinning in the vertex shader

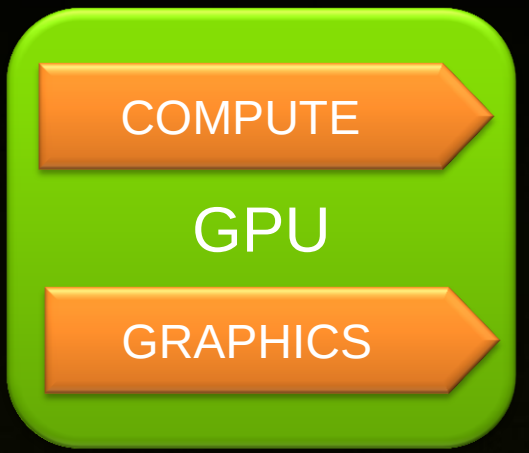


Full processing per character per frame

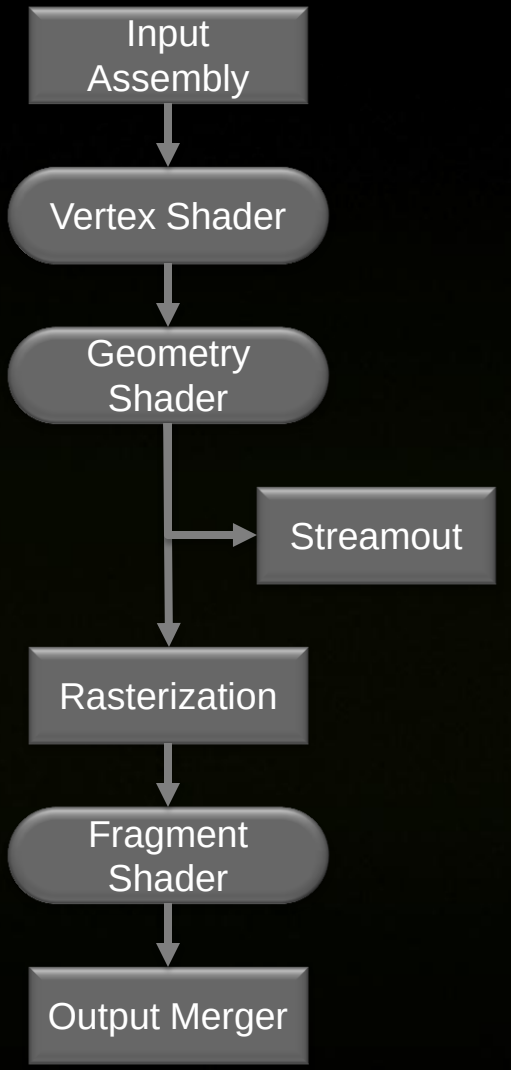
Steps	Complexity	Scale	Cost
Animate Motion Blend	Once	1	0%
Evaluate Skeleton Local Pose	nbJoints x nbBlendedMocaps	~30x2	1%
Evaluate Skeleton Absolute Pose	nbJoints	~30	1%
Evaluate Bind-Pose Matrices	nbClusters	~25	1%
Evaluate Skinned Surface	nbVertices x nbInfluencingClusters x nbViews (GPU version)	~1000x4x(2)	30%
Draw Skinned Surface	nbViews	Max(nbVertices, nbPixels)	64%

How to execute Computation batches on the GPU ?

- Compute pipeline
  - Cuda
  - OpenCL
  - DirectX Compute
  - OpenGL Compute
- Graphics pipeline
  - Avoid the context switch
  - Using D3D10 or D3D11 features

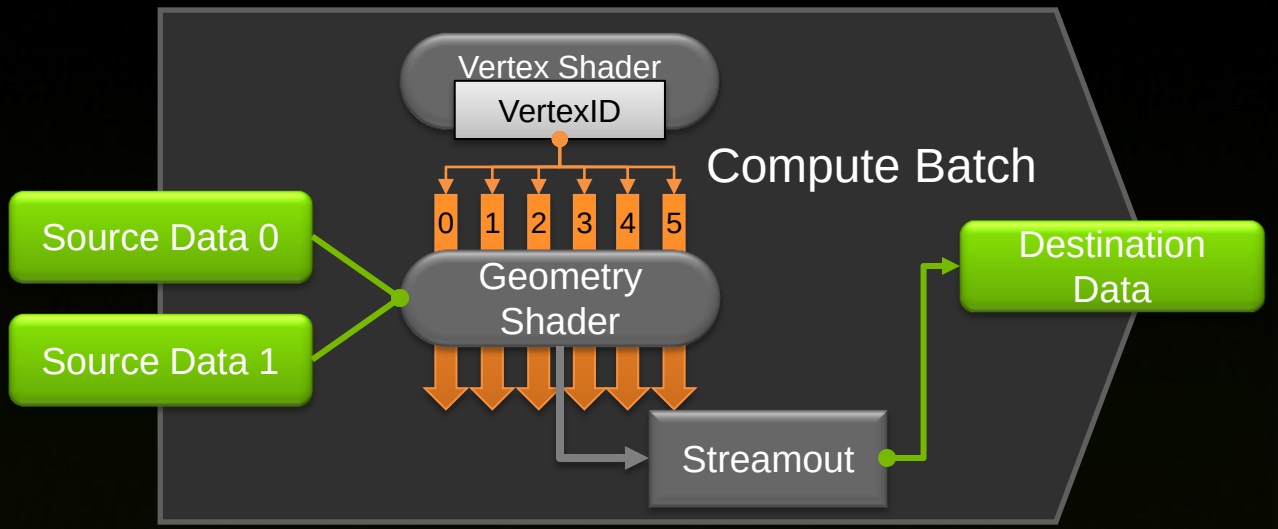


Computation batch with the Graphics Pipeline

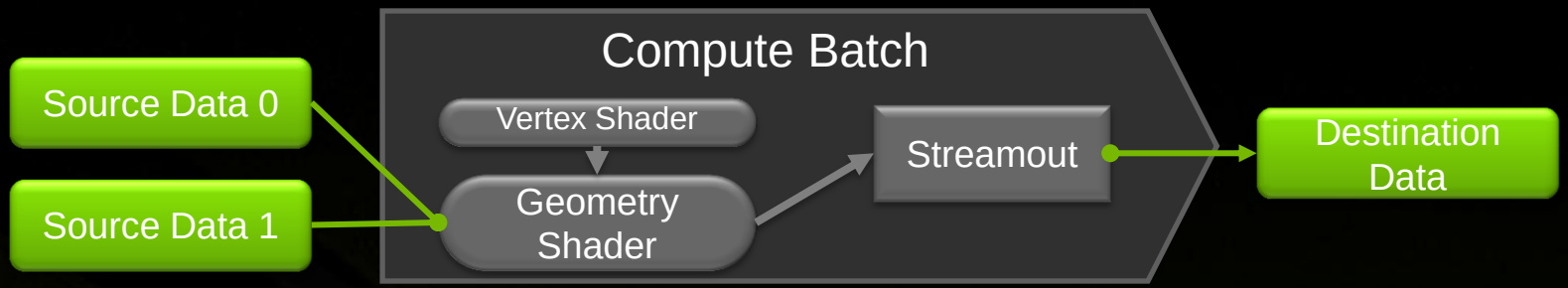


Computation Batch with the Graphics Pipeline

- Using Geometry Shader and Streamout
- Run one thread for every destination element
  - Perform the compute kernel in the Geometry Shader
  - Pass the VertexID from the vertex shader as the thread index
- Source data bound to the shader with a Shader Resource View as Texture Buffer
- Written data is bound as Streamout buffer



Computation Batch



Compute platform

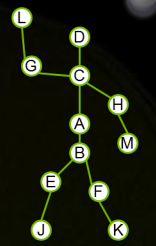
```
uint vsmain( in uint vID : SV_VertexID ) : VTXID
{
    return vID;
}

[maxvertexcount(2)]
void gsmain( point uint indices[1] : VTXID,
             inout PointStream<float4> vouts : VOUT )
{
    float4 vals[2]
    compute( index, vals );
    vals.Append( vals[0] );
    vals.Append( vals[1] );
}
```

Compute kernel

```
Buffer<float4> srcData0 : register(t0);
Buffer<float4> srcData1 : register(t1);

void compute( uint index,
              inout float4 vals[2] )
{
    // Fetch from src data
    float4 foo = srcData0.Load( index );

    // compute something
    …

    // output
    vals[0] = float4( … );
    vals[1] = float4( … );
}
```

Object Model Architecture

- Geometry
  - Vertex Buffer
  - Index Buffer
- Skeleton
  - Joint Layout
  - Joint Local Transform Buffer
  - Joint Absolute Transform Buffer
- Skin
  - Joint Layout
  - Bind Transform Buffer
  - Skin Vertex Attribute Buffer
- Deformer
  - Joint Bind Pose Transform Buffer
  - Bind Pose Map
- Clip
  - Track Header Buffer
  - Time Keys Buffer
  - Value Keys Buffer
- Repertoire
  - List of Clips grouped by motion style

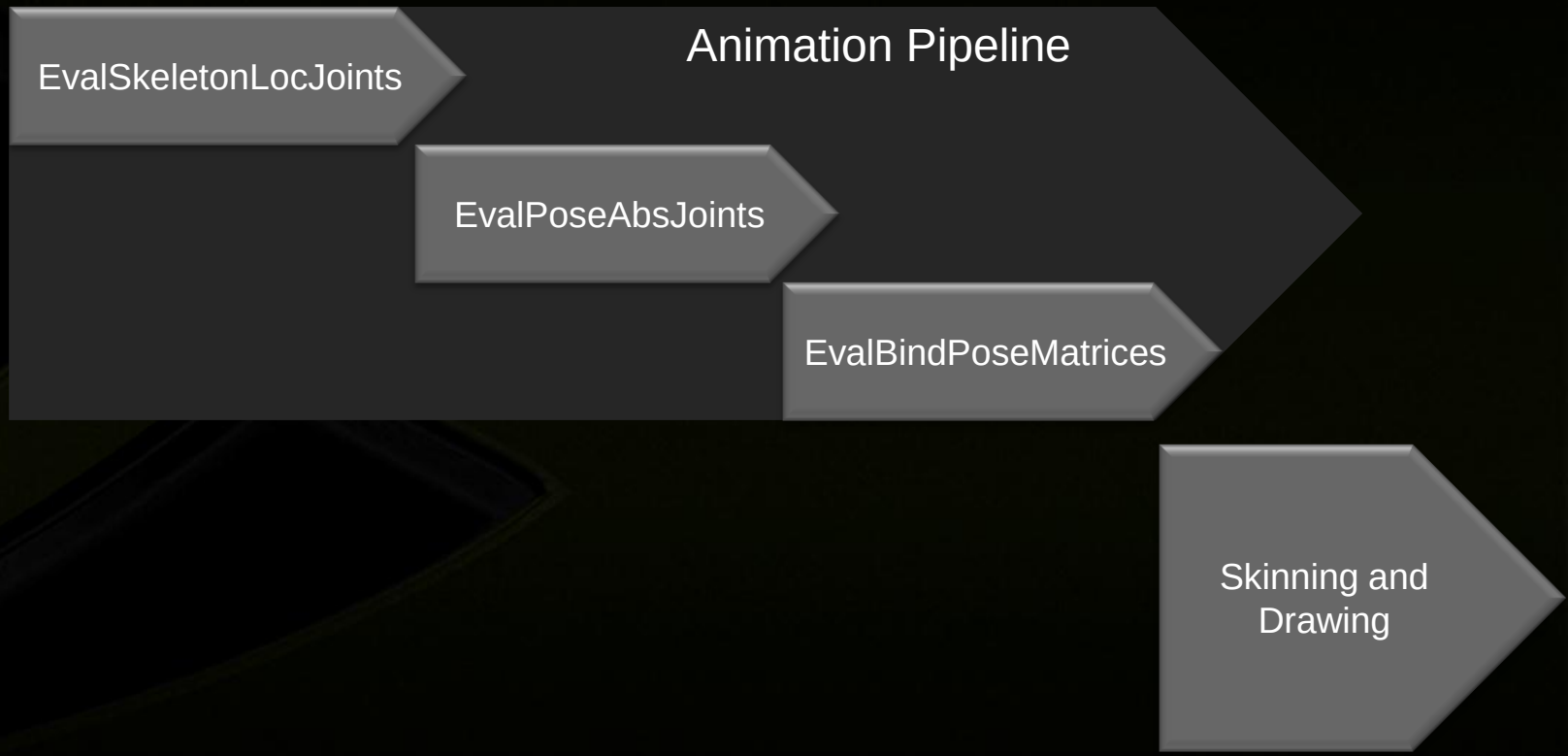


Skeleton

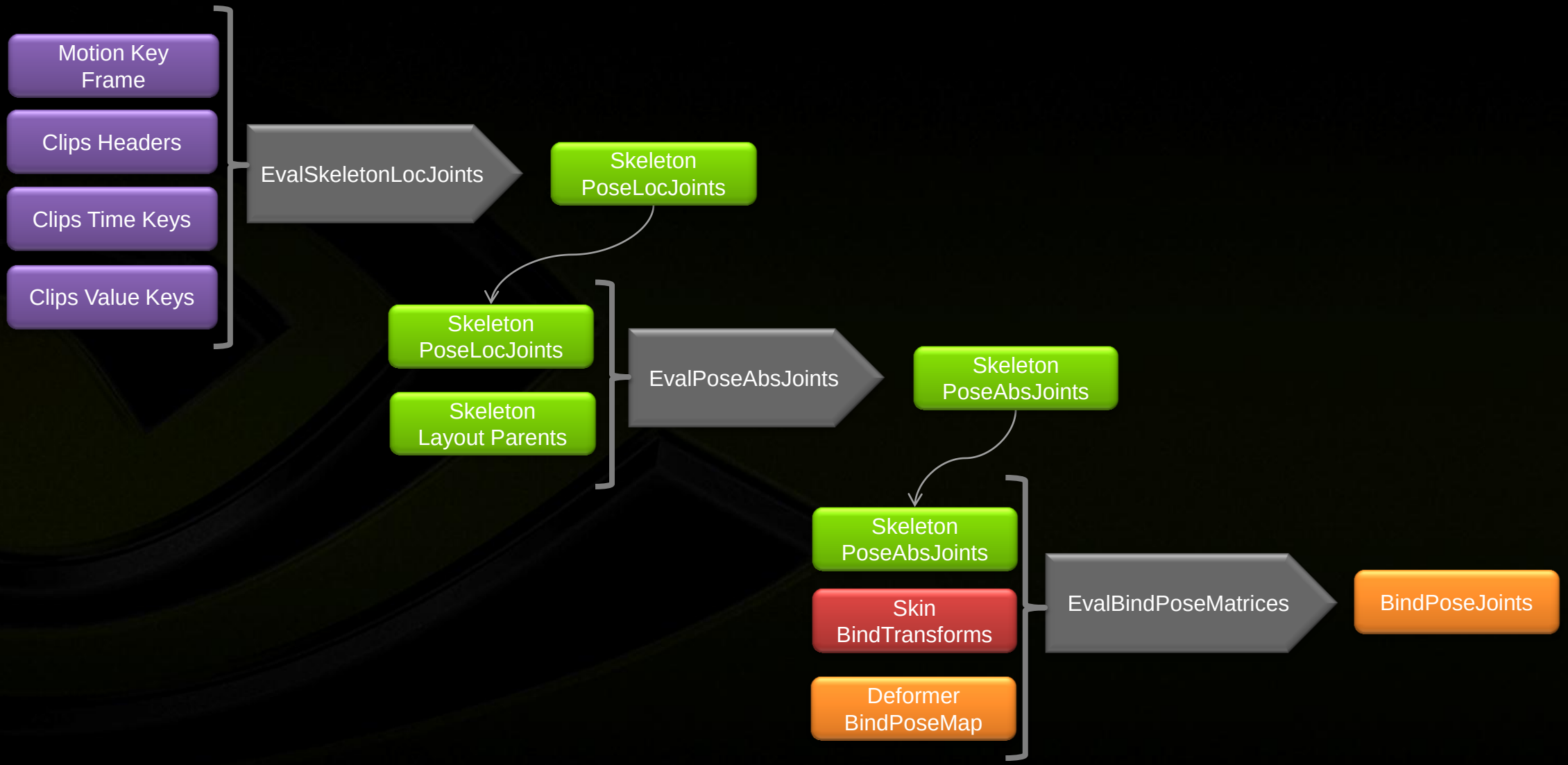
Skin

Full processing per character per frame

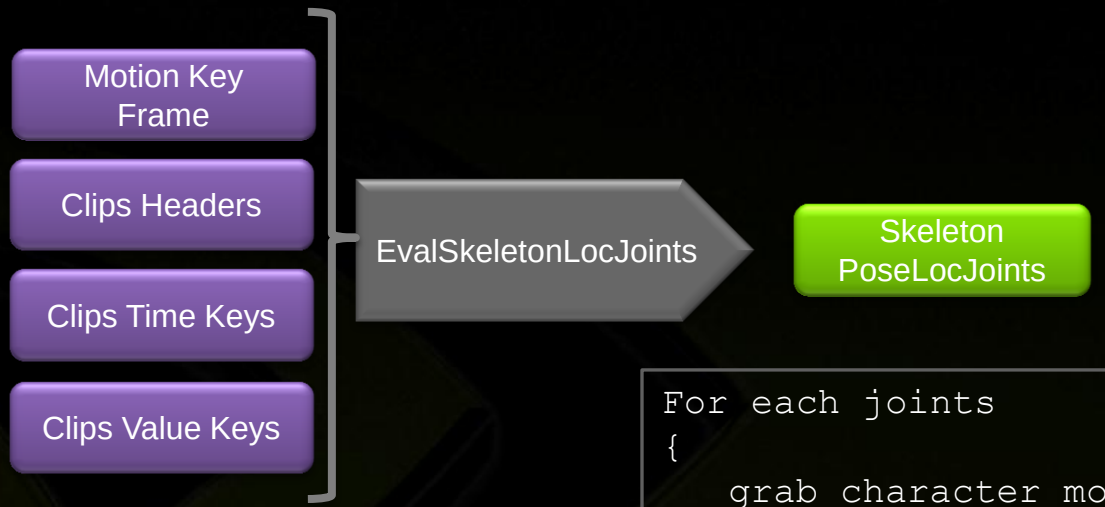
Steps
Animate Motion Blend
Evaluate Skeleton Local Pose
Evaluate Skeleton Absolute Pose
Evaluate Bind Pose matrices
Evaluate Skinned Surface
Draw Skinned Surface



Animation pipeline



Evaluate Skeleton Pose local joints




```
For each joints
{
   grab character motion key frame
   blend different poses translation and rotations
   write out local translation and rotation
}
```

Evaluate Skeleton Pose absolute joints


```
float4x4 locPoseTransform( uint jointId )
{
    return PoseLocJoints[ jointId ]
}

float4x4 evalPoseAbsJoint ( uint jointId )
{
    int currentId = jointId;
    int currentParentId = ParentJoints[ currentParentId ];
    float4x4 transform = locPoseTransform( currentId );
    while ( currentParentId != currentId )
    {
        currentId = currentParentId;
        currentParentId = ParentJoints[ currentParentId ];
        transform = mul( locPoseTransform( currentId ), transform );
    }
    return transform;
}
```

Evaluate BindPose Matrices


```
For each Joint in BindPoseMatrices
{
    int    PoseJointID = BindPoseMap[ JointID ];
    Joint = PoseJoints[ PoseJointID ] * BindTransforms[ JointID ].invBindMat;
}
```

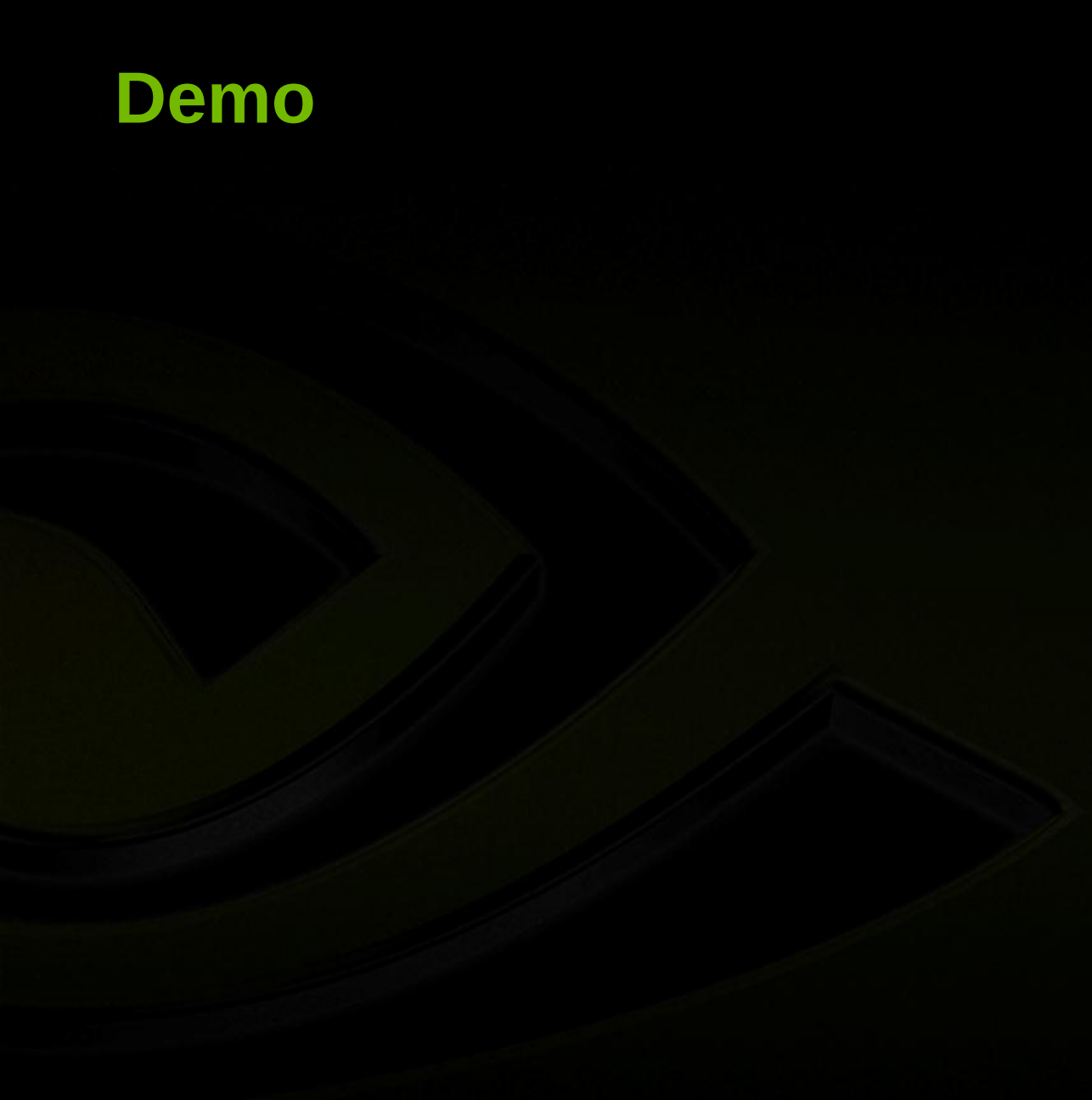
Rendering Performance

- Drawing is the bottleneck
  - Rendering many geometries cost…
    - 1000 Characters @ 5000 Triangles = 5 MTTriangles per frame
    - With Shadow pass = 10 Mtri/frame
    - 1.2BTriangles / s bandwith gives 120Hz peak perf
    - But in reality we perform at ~50%
  - The skinning (even with GPU) cost a lot
    - ~ 30% compared to no skinning
    - Not tried Dual Quaternion yet, overall hope for a theoric ~20-25% improvement

Performance Considerations

- Instancing is really an improvement if the rendering is CPU Bound by the calls to rendering pipeline
  - Otherwise, it will still be GPU Bounded
  - But very useful anyway for crowds…
- Animation
  - Performing the 3 animation steps is very small compared to rendering
  - 5000 Characters at 32 joints (and 25 clusters)
    - cost 1ms on the GPU (Quadro 5000) vs. ~20ms on CPU
    - 150 Million joints per sec
    - Could go up to 75000 characters at 60Hz

Demo



Conclusion

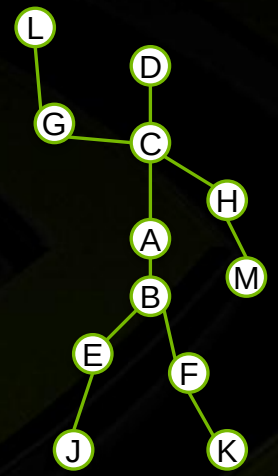
- Demonstrated a crowd animation playback pipeline running on the GPU
  - Performing at 150 Million joints per secondes
  - Equivalent to 75000 Characters of 30 bones animated at 60Hz
- Rendering is still the bottleneck costing ~50x the animation cost
  - Skinning on the GPU is a must of course
  - Instancing is a must
  - Need to explore LOD techniques
  - Need to add blend shape
- Hopefully this work will make it to Geppetto ☺

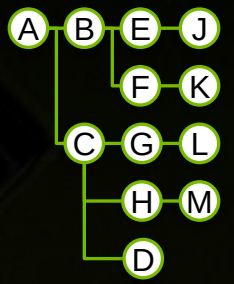
Thanks to:

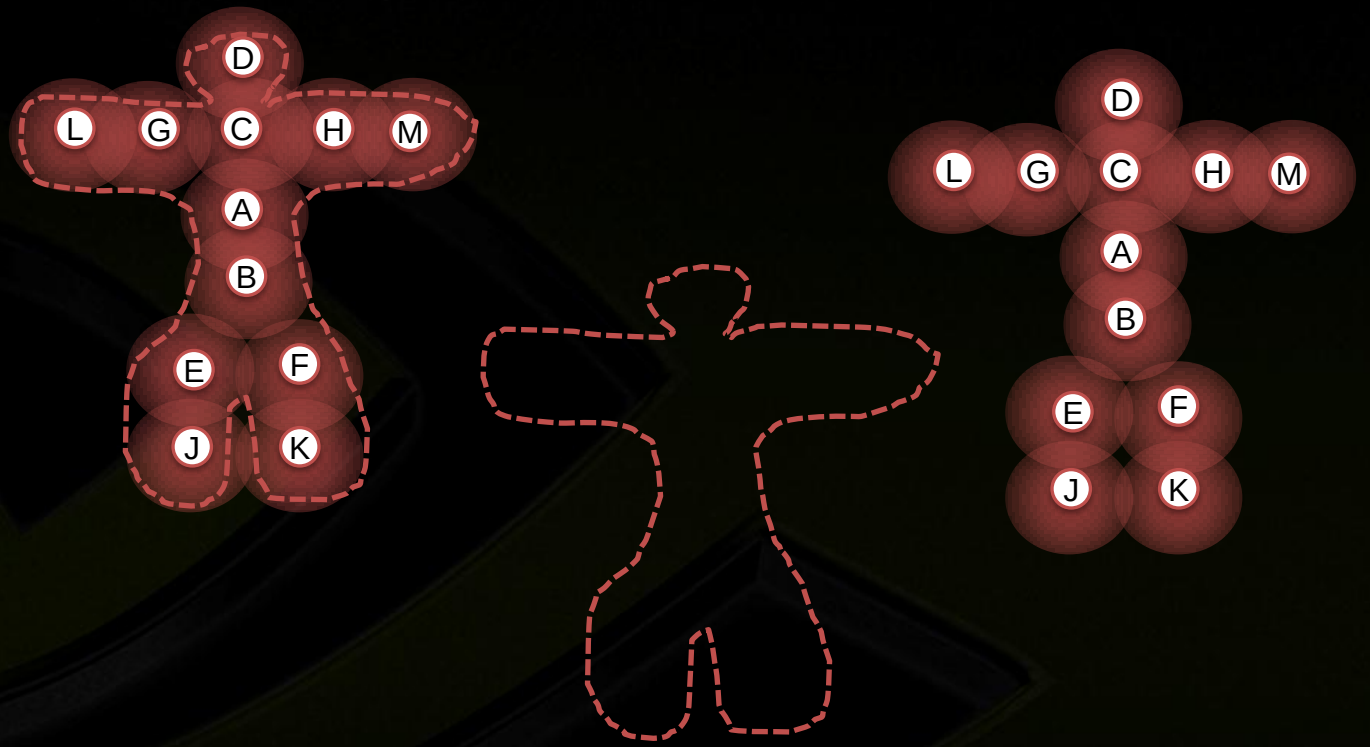
Michael Girard and Abdelhak Ouhlal at Autodesk for our cooperation

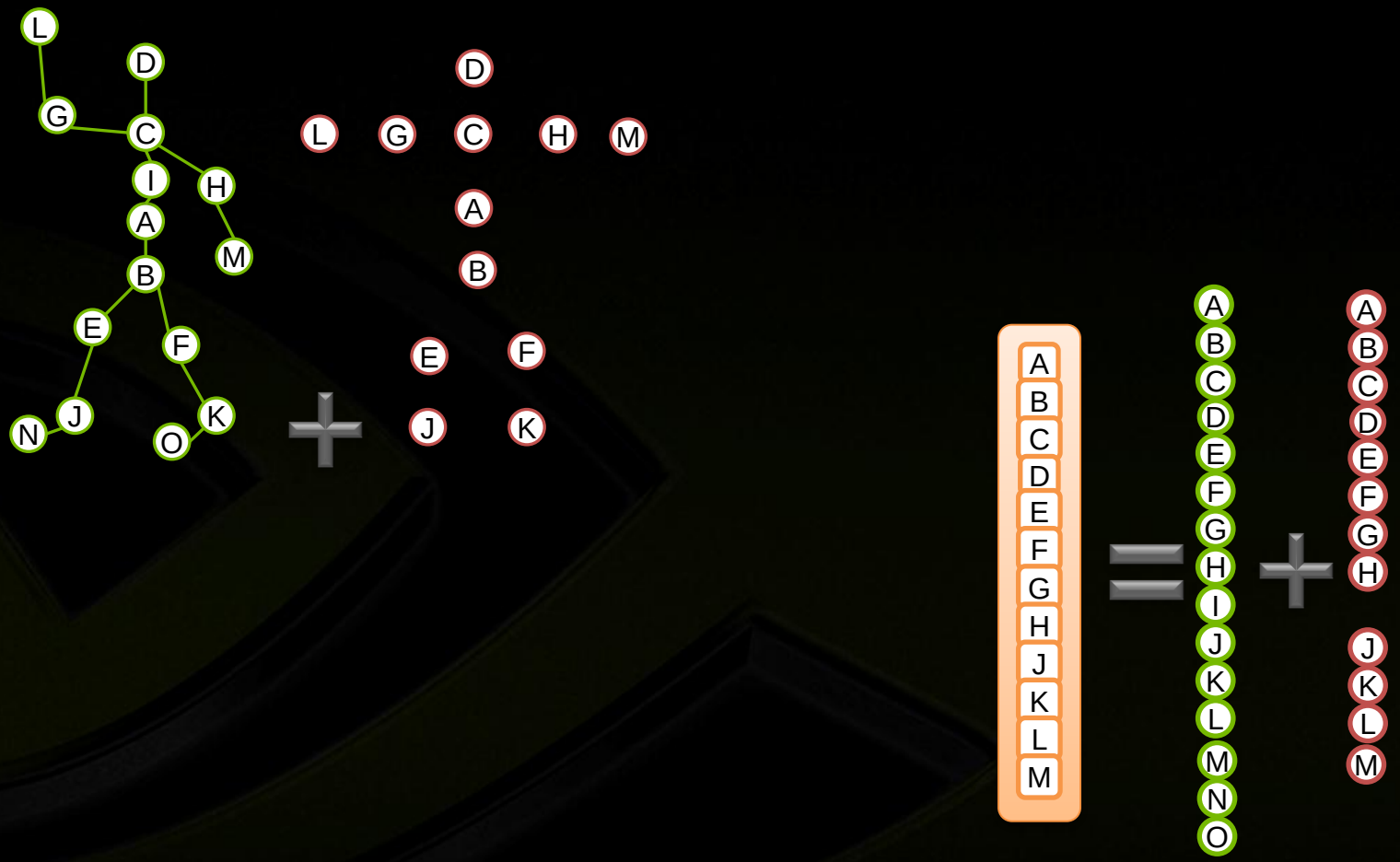
Questions ?

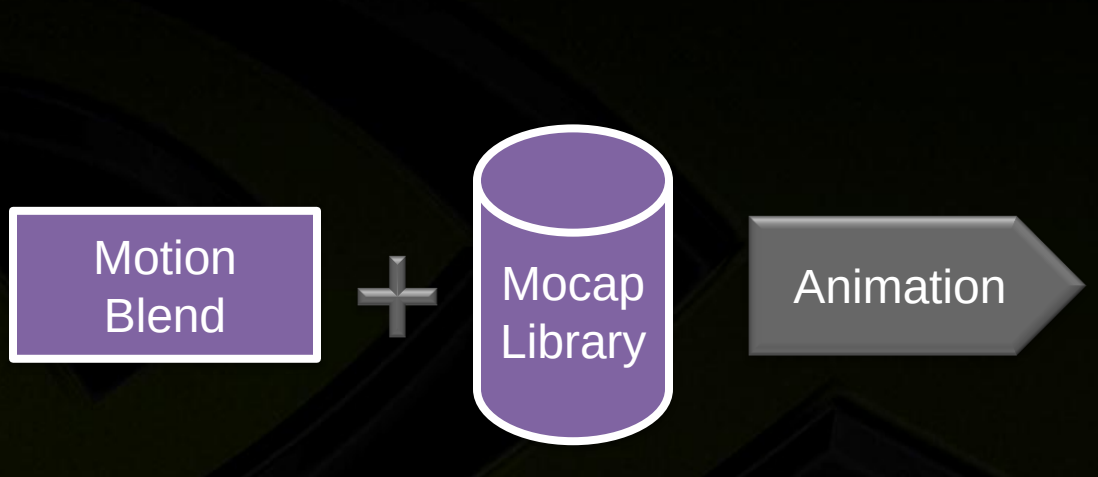
sgateau@nvidia.com

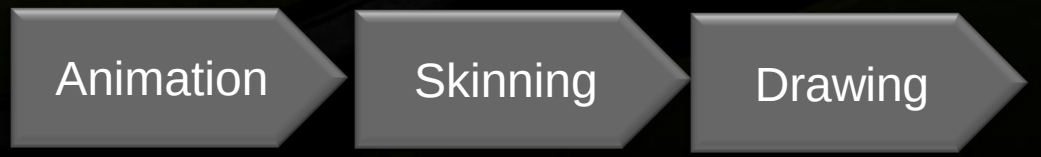

L
D
G
C
H
A
M
B
E
F
J
K

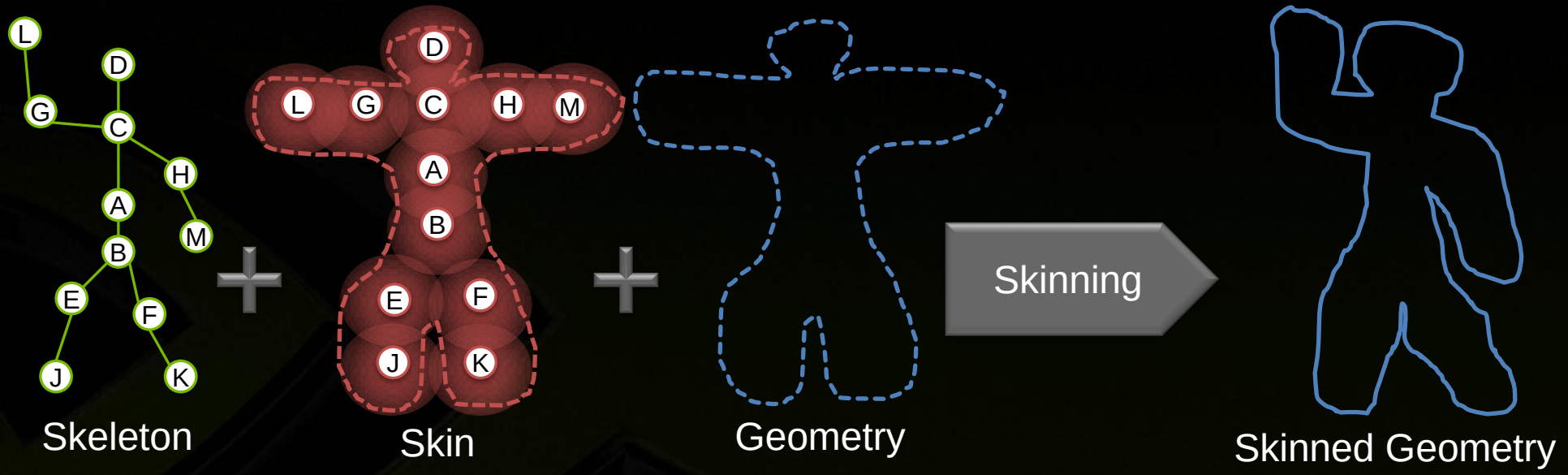

A
B
E
J
F
K
C
G
L
H
M
D

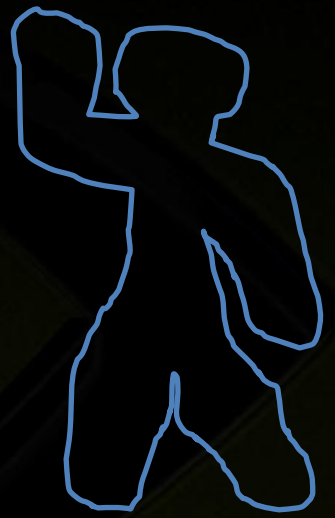

D
L
G
C
H
M
A
B
E
F
J
K
D
L
G
C
H
M
A
B
E
F
J
K


L
D
G
C
I
H
A
M
B
E
F
N
J
O
K
D
L
G
C
H
M
A
B
E
F
J
K
A
B
C
D
E
F
G
H
I
J
K
L
M
A
B
C
D
E
F
G
H
J
K
L
M


Motion Blend
Mocap Library
Animation
L
D
G
C
H
A
M
B
E
F
J
K
Skeleton


Animation
Skinning
Drawing

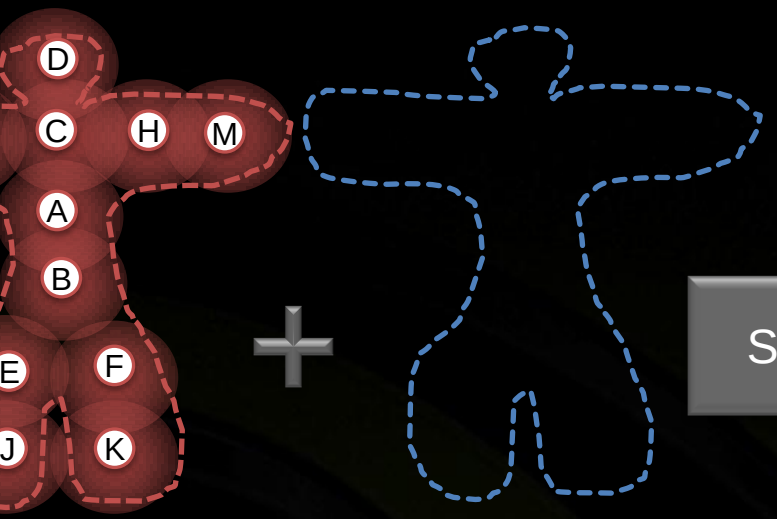

L
D
G
C
H
A
M
B
E
F
J
K
Skeleton
Skin
Geometry
Skinning
Skinned Geometry
EvalBindPose



Skinned Geometry

Drawing

Frame



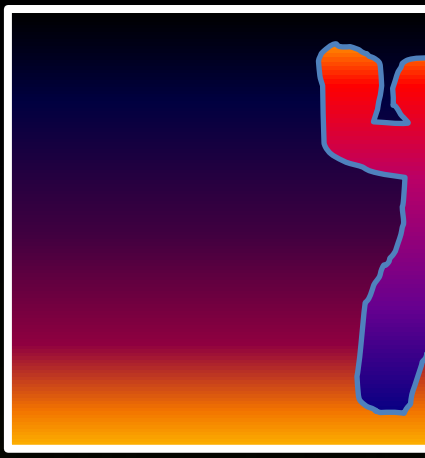
Skin

Geometry

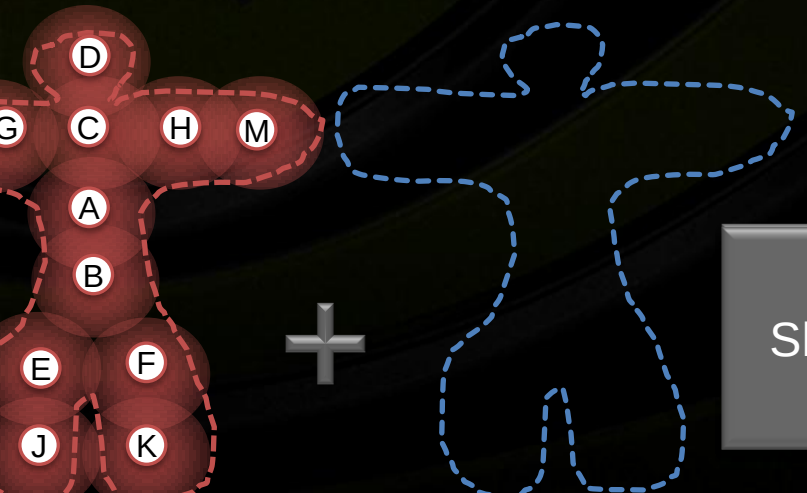
Skinning

Skinned Geometry

Drawing



Fram



Skinning

Drawing