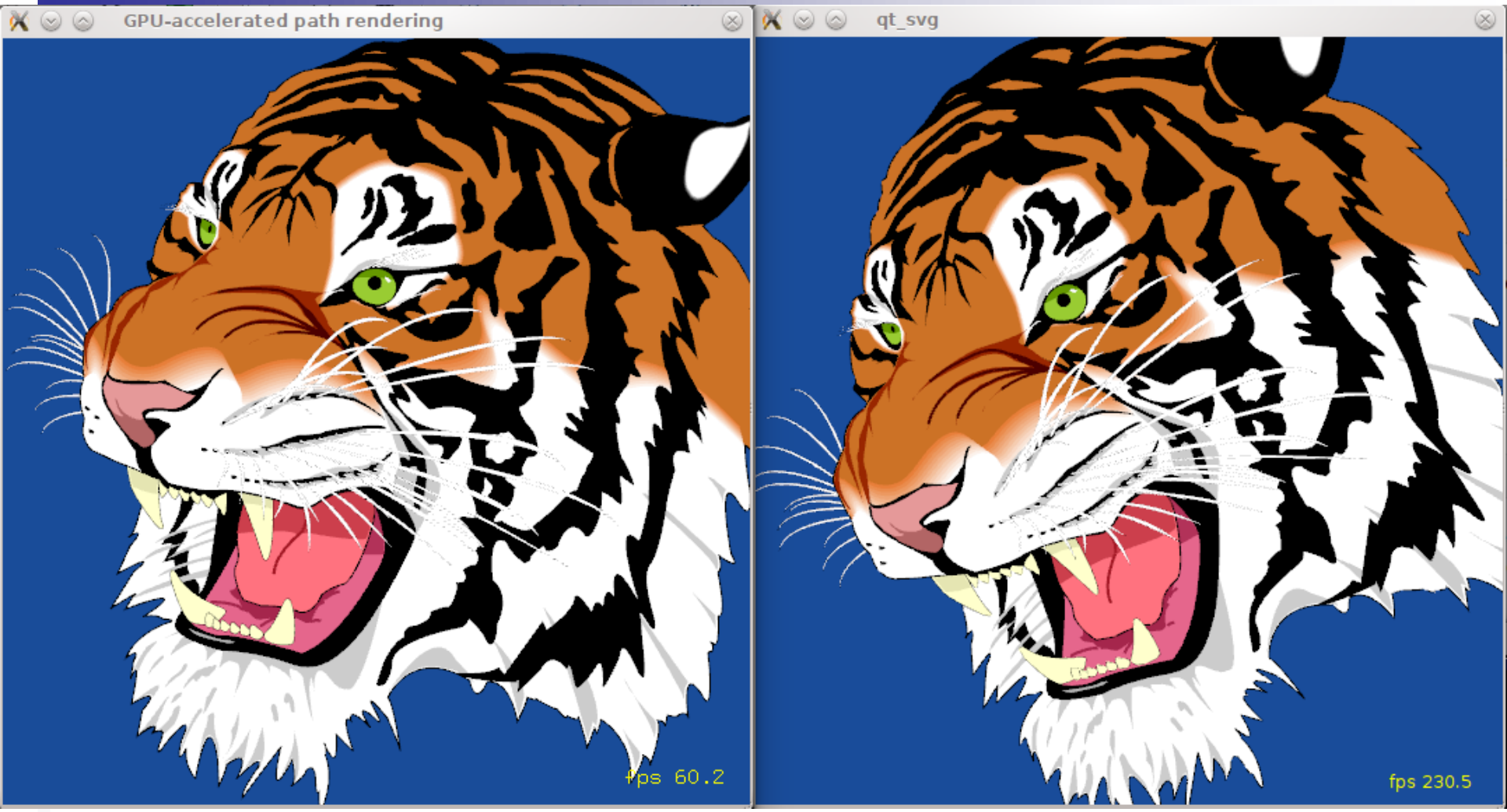
# PATH RENDERING

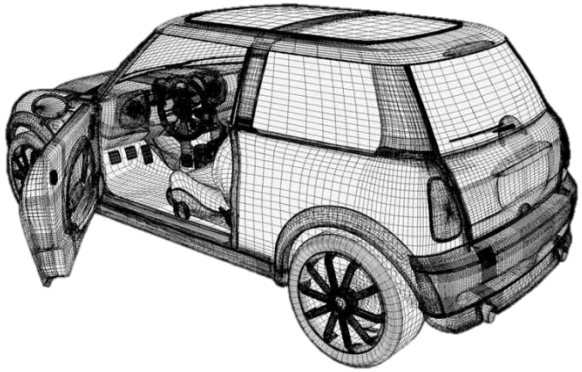


# AGENDA

- Introducción
- Path Rendering
- Características del Path Rendering
- Estructura del NV_PATH_RENDERING
- Stencil then Cover (StC)
- Ventajas de la GPU
- Ideas Finales

# GRÁFICOS VECTORIALES

- Empleado para describir diversos tipos de gráficos
  - Mallados para el despliegue



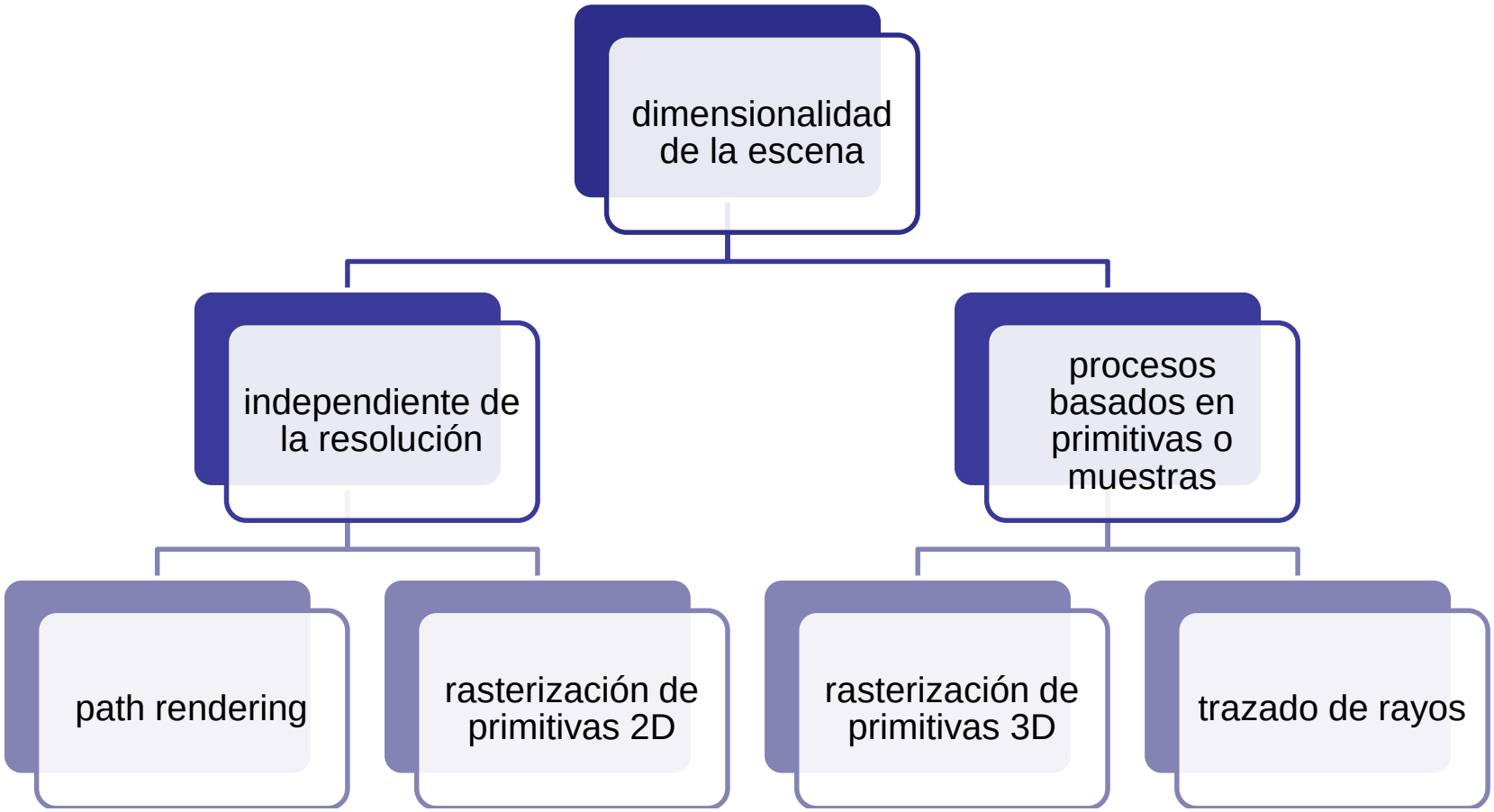
  - Plotting o render caligráfico

**HP Design Jet**

**Scalar Vector Graphics**

  - Gráficos 2D de resolución independiente

# Taxonomía del Rendering




**Ejemplos:**
**Postscript, PDF, SVG, TrueType, Adobe Flash, Microsoft Silverlight**

**Ejemplos:**
**GDI, Xlib**

**Ejemplos:**
**OpenGL, Direct3D**

**Ejemplos:**
**Mental Ray**

# PATH RENDERING

**Impresión e Intercambio de Documentos**


Adobe PostScript

Open XML Paper (XPS)

**Fuentes de Resolución Independiente**

OpenType

TrueType

**Aplicaciones Web Interactivas**

Flash

Microsoft Silverlight


Scalable Vector Graphics

HTML 5

**Interfaces de Programación 2D**


Java 2D API


QtGui API


Mac OS X 2D API


Khronos API

**Aplicaciones de Oficina e Imágenes**




Adobe Illustrator

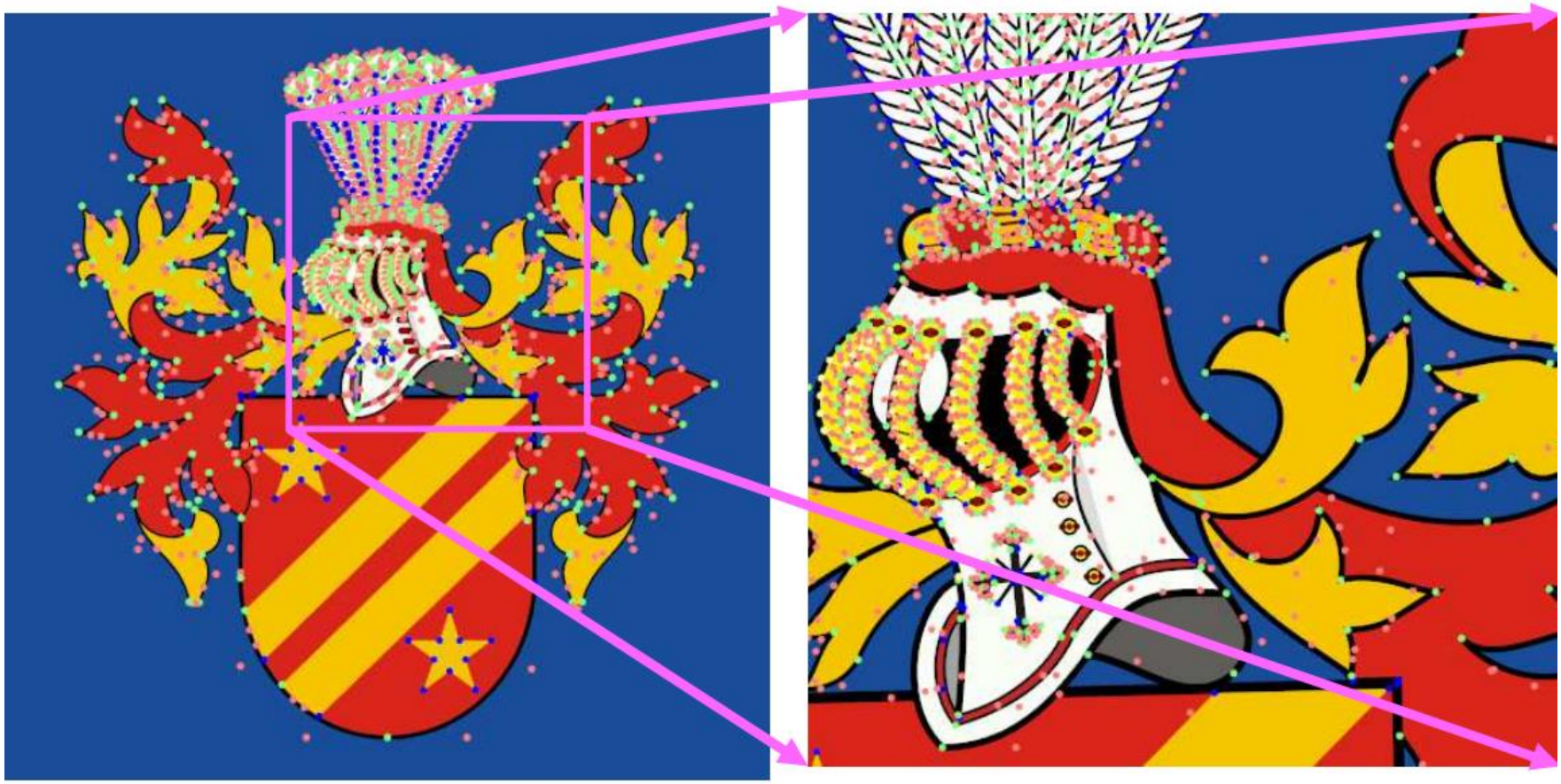
Open Source Inkscape

# PATH RENDERING

- Enfoque resolución para el despliegue de gráficos 2D
- Especificado por objetos del tipo path (*outlines*)
- Secuencia de comandos: líneas, curvas y arcos
- Cóncavos, auto-intersectados y complejos

# PATH RENDERING

- Contenido definido por puntos de control



# INICIOS DEL PATH RENDERING

- John Warnock y Douglas Wyatt, Xerox PARC
  - Paper presentado en SIGGRAPH ’82 “A Device Independent Graphics Imaging Model for Use with Raster Devices”
- Warnock y Geschke fundaron Adobe Systems en Diciembre de 1982
  - $20.1 billones en el mercado (NVIDIA+AMD = $14.3 billones)

John Warnock

Charles Geschke

Computer Graphics Volume 16, Number 3 July 1982

A Device Independent Graphics Imaging Model for Use with Raster Devices

John Warnock and Douglas K. Wyatt

Xerox Palo Alto Research Centers
3333 Coyote Hill Road
Palo Alto, CA 94304

**Abstract**

In building graphic systems for use with raster devices, it is difficult to develop an intuitive, device independent model of the imaging process, and to preserve that model over a variety of device implementations. This paper describes an imaging model and an

**Raster Devices**

The class of raster devices encompasses a wide range of displays, plotters, and printers. These include full color (24 bit per pixel) displays, grey level displays, simple low resolution binary (1 bit per pixel) displays, electrostatic plotters, high resolution film recorders, and laser printers. Raster devices, because of their potential ability

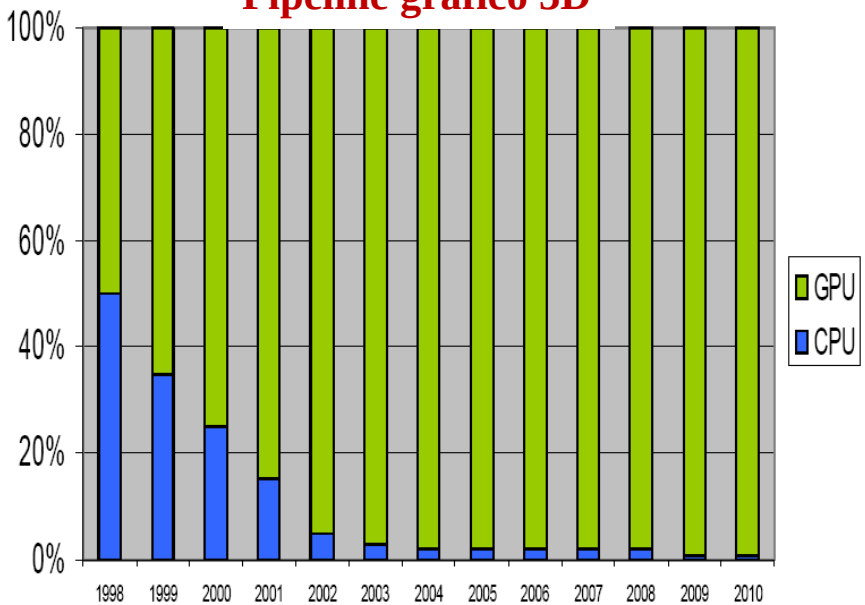
# NV_PATH_RENDERING

## Objetivo principal

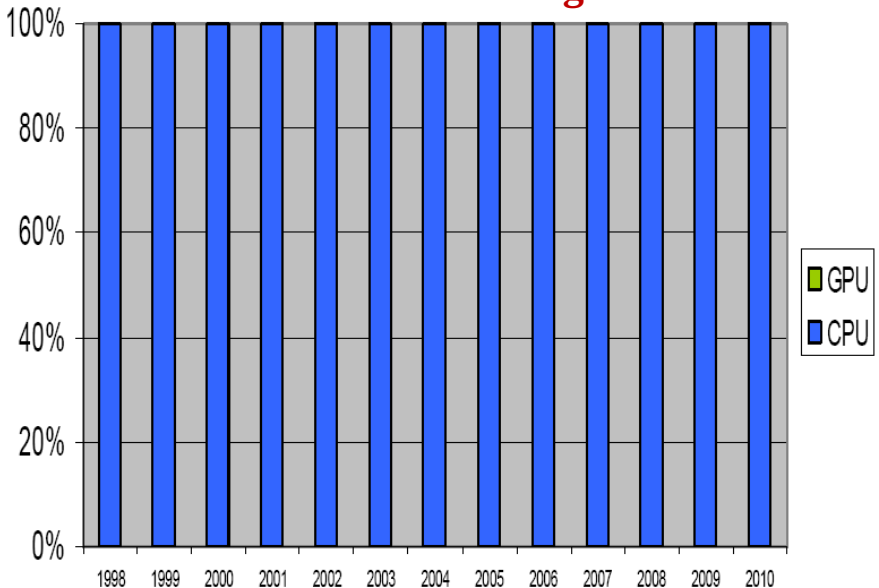
## Realizar el proceso de *rendering* de un camino (*path*) en la GPU

- Extensión OpenGL soportada por las GPUs de NVIDIA con soporte CUDA. Aparición Junio 2011

**Pipeline gráfico 3D**




**Path Rendering**



# CARACTERÍSTICAS DE NV_PATH_RENDERING

- Independiente de la resolución
- Algoritmo del Pintor
- Primitivas de rendering: caminos (*paths*). Componentes de una primitiva: puntos de control.
- Convexas, cóncavas, auto-intersectadas y con orificios
- *Rendering*: relleno (*filling*) y *stroking*
- Parámetros para una línea:
  - Ancho
  - Patrón de punteado
  - *Capping*
  - *Join style*

# ESTRUCTURA DE NV_PATH_RENDERING

1. Manejo del objeto Path
   - Basados en OGL, empleando una variable Gluint
   - glIsPath, glGenPathsNV, glDeletePathsNV

2. Especificación de los datos de entrada

3. Parámetros para el objeto Path

4. Render del Path

# 2. Especificación de los datos de entrada

- Idea: Especificar las primitivas de un path

a) Cadena de caracteres: SVG y PS (glPathStringNV)

b) Arreglo de comandos (glPathCommands, glPathSubCommands, glPathCoords, glPathSubCoords)

c) Empleando fuentes: glyphs (glPathGlyphNV, glPathGlyphRangeNV)

d) Combinación lineal de paths existentes: interpolación de 1 o más paths (glInterpolatePathsNV, glCombinePathsNV)

e) Tranformación lineal de paths existentes (glTransformPathNV)

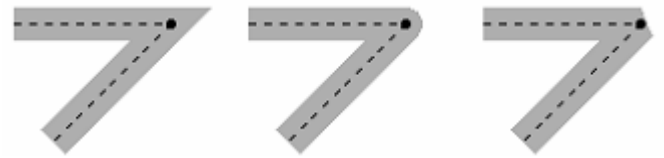
# PATH STRING

- GL_PATH_FORMAT_SVG_NV
  - BNF de la especificación SVG 1.1
  - String codificado en ASCII. Ejemplos:
    - “M100,180 L40,10 L190,120 L10,120 L160,10 z”
    - “M300 300 C 100 400,100 200,300 100,500 200,500 400,300 300Z”
- GL_PATH_FORMAT_PS_NV
  - Gramática para crear paths provista por PostScript
  - Codificación más compacta que SVG: binario y ASCII-85
  - Ejemplos
    - “100 180 moveto 40 10 lineto 190 120 lineto 10 120 lineto 160 10 lineto closepath”
    - “300 300 moveto 100 400 100 200 300 100 curveto 500 200 500 400 300 300 curveto closepath”

# 3. Parámetros para el Objeto Path

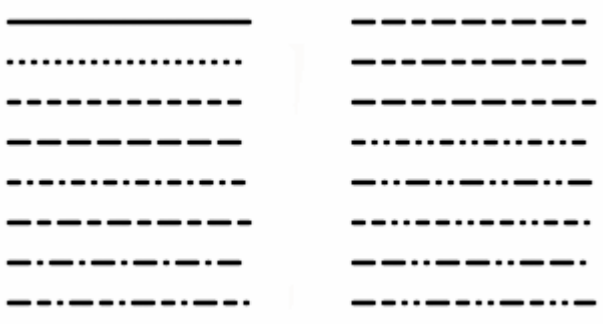
- Parámetros para el relleno: *fill mode*, *fill mask* y *fill cover mode*
- Parámetros para el trazo
  - Ancho: número punto flotante
  - Forma de fin de línea: plana, cuadrada, redonda, triangular
  - Estilo de las uniones: puntiaguda, redondeada, truncada
  - Diversos estilos de punteado

GL_FLAT GL_ROUND_NV GL_SQUARE_NV



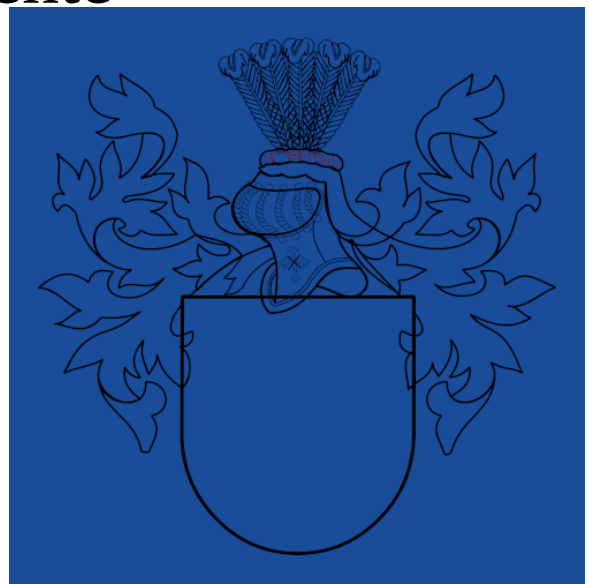
GL_MITER_NV GL_ROUND_NV GL_BEVEL_NV

Diversos esquemas de punteados



# 4. Render del Path

- Paradigma de 2 pasadas “stencil, then cover (StC)”
- Dos operaciones básicas: Stencil y Cover
  - glStencilFillPathNV y glStencilStrokePathNV
  - glCoverFillPathNV y glCoverStrokePathNV
- Obtener un *stroking* correcto es costoso computacionalmente

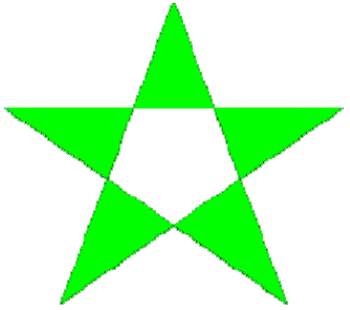




# STENCIL THEN COVER (STC)

- Primer paso (stencil):
  - El stencil es marcado en la ruta del objeto path
  - Tanto para el relleno como para el trazado
  - Proceso de rasterización
  - No se actualizan colores en el framebuffer
- Segundo paso (cover):
  - Coloreado de píxeles con la información del stencil
  - Limpia los valores del stencil para repetir la operación

# EJEMPLO DE UN PATH RENDERING

- Dibujar una estrella cóncava de 5 puntas



estilo par-impar

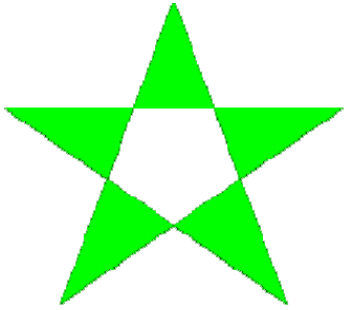
estilo completo

- Especificación de la estrella por un string

```
Gluint pathObj = 42;
const char* pathString = "M100,180,L40,10,L190,120,L10,
   120,L160,10 z"
glPathStringNV(pathObj, GL_PATH_FORMAT_SVG_NV,
   strlen(pathString), pathString);
```

# EJEMPLO DE UN PATH RENDERING

- Dibujar una estrella cóncava de 5 puntas



estilo par-impar

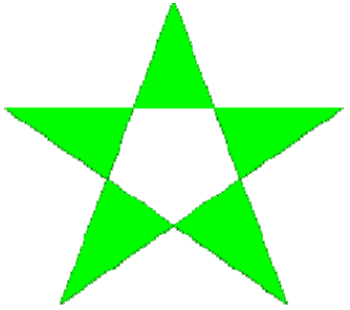
estilo completo

- Especificación de la estrella por datos

```
static const Glubyte pathCommandsObj[5] = {GL_MOVE_TO_
  NV,GL_LINE_TO_NV,GL_LINE_TO_NV,GL_LINE_TO_NV,GL_LINE
  _TO_NV,GL_CLOSE_PATH_NV};
static const GLshort pathVertices[5][2] = {{100,180},
  {40,10},{190,120},{10,120},{160,10}};
glPathCommandsNV(pathObj, 6, pathCommands, GL_SHORT,10,
  pathVertices);
```

# EJEMPLO DE UN PATH RENDERING

- Dibujar una estrella cóncava de 5 puntas



estilo par-impar

estilo completo

- Especificación de la estrella por comandos PostScript

```
const char * psPathString = "100 180 moveto"
"40 10 lineto 190 120 lineto 10 120 lineto 160 10
   lineto closepath";
glPathStringNV(pathObj, GL_PATH_FORMAT_PS_NV, (Glsizei)
   strlen(psPathString), psPathString);
```

# Ejemplo de un Path Rendering

- Inicialización
- Limpiar el *stencil buffer* a 0 y el *color buffer* a negro

```
glClearStencil(0);
glClearColor(0,0,0,0);
glStencil(~0);
glClear(GL_COLOR_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);
```

- Especificar la transformación del path

```
glMatrixLoadIdentityEXT(GL_PROJECTION);
glMatrixOrthoEXT(GL_PROJECTION,0,200,0,200,-1,1);
glMatrixLoadIdentityEXT(GL_MODELVIEW);
```

- Realizar el despliegue

# EJEMPLO DE UN PATH RENDERING

- Desplegar la estrella al estilo “completo”

```
glStencilFillPathNV(pathObj, GL_COUNT_UP_NV, 0x1F);
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_NOTEQUAL, 0, 0x1F);
glStencilOp(GL_KEEP, GL_KEEP, GL_ZERO);
glColor3ub(0,255,0);
glCoverFillPathNV(pathObj, GL_BOUNDING_BOX_NV);
```

- Desplegar la estrella al estilo “par-impar”

```
glStencilFunc(GL_NOTEQUAL, 0, 0x1);
```

# EJEMPLO DE UN PATH RENDERING

- Agregando trazos de borde

```
glPathParameteriNV(pathObj, GL_PATH_JOIN_STYLE_NV,
GL_ROUND_NV);
glPathParameteriNV(pathObj, GL_PATH_STROKE_WIDTH_NV,
6.5);
```

- Colocar 0x1 en el stencil de acuerdo al path

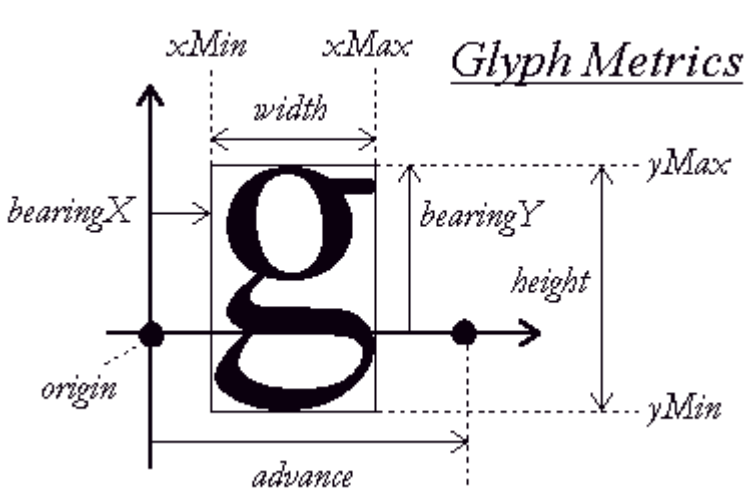
```
glStencilStrokePathNV(pathObj, 0x1, ~0);
```

- Colorear el trazo de color amarillo, cubrir el path

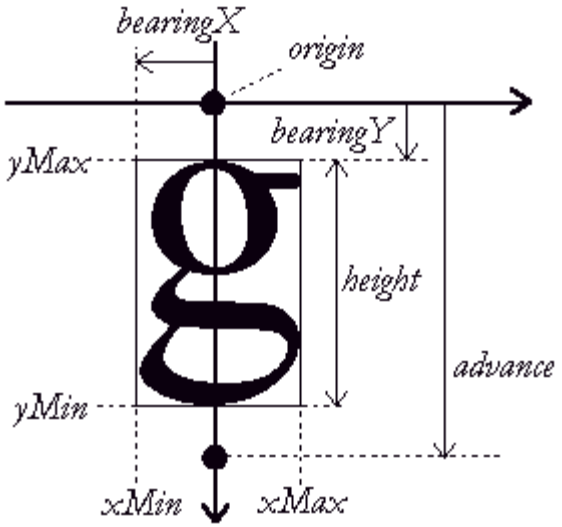
```
glColor3f(1,1,0);
glCoverStrokePathNV(pathObj, GL_BOUNDING_BOX_NV);
```

# TEXTO EN NV_PATH_RENDERING

- Basado en la métrica FreeType2




**métrica horizontal**




**métrica vertical**

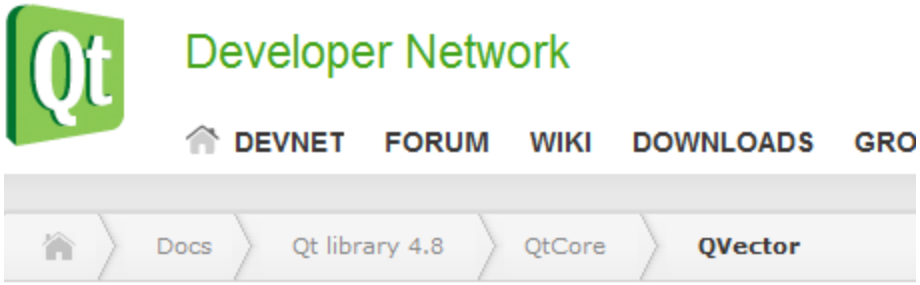
- También provee las métricas por fuente

# TEXTO EN NV_PATH_RENDERING

- GL_SYSTEM_FONT_NAME_NV
  - Ejemplo: “Arial”, “Verdana”
  - Windows emplea los fonts directamente como un servicio
  - Linux emplea fontconfig + librerías freetype2
- GL_STANDARD_FONT_NAME_NV
  - Tres tipos: “Sans”, “Serif” y “Mono”
  - Basado en fuente de DejaVu
  - Siempre disponibles
- GL_SYSTEM_FILE_NAME_NV
  - Emplea freetype2 para la carga de las fuentes desde un archivo
  - En Windows, se requiere del DLL de freetype2
  - En Linux, se cargan directamente

# Librerías de Path Rendering

- OpenGL / Cairo
- OpenGL / Qt
- OpenGL / Skia
- OpenGL / D2D
- OpenGL / WARP




## QVector Class Reference

The QVector class is a template class that provides a dynamic array.

```
#include <QVector>
```


Project Home | Wiki | Issues | Source

Summary | People

**Project Information**

Starred by 460 users
Project feeds

**Code license**
Other Open Source
See source for details

**Labels**
Graphics, C, Embedded, 2D, Perspective, Vector, cplusplus, OpenGL, PDF, GPU, CrossPlatform

**Members**
r...@google.com,

**What it is**

Skia is a complete 2D graphic library for drawing Text, Geometries, and

- 3x3 matrices w/ perspective
- antialiasing, transparency, filters
- shaders, xfermodes, maskfilters, patheffects
- subpixel text

Device backends for Skia currently include:

- Raster
- OpenGL
- PDF
- XPS
- Picture (for recording and then playing back into another Canvas)


News | Download | Documentation | Contact | Examples

Latest news: 2012-04-29: cairo 1.12.2 release available 2012-03-23: cairo 1.12.0 release available

Cairo is a 2D graphics library with support for multiple output devices. Currently supported output targets include the X Window System (via both Xlib and XCB), Qu Win32, image buffers, PostScript, PDF, and SVG file output. Experimental backends include OpenGL, BeOS, OS/2, and DirectFB.

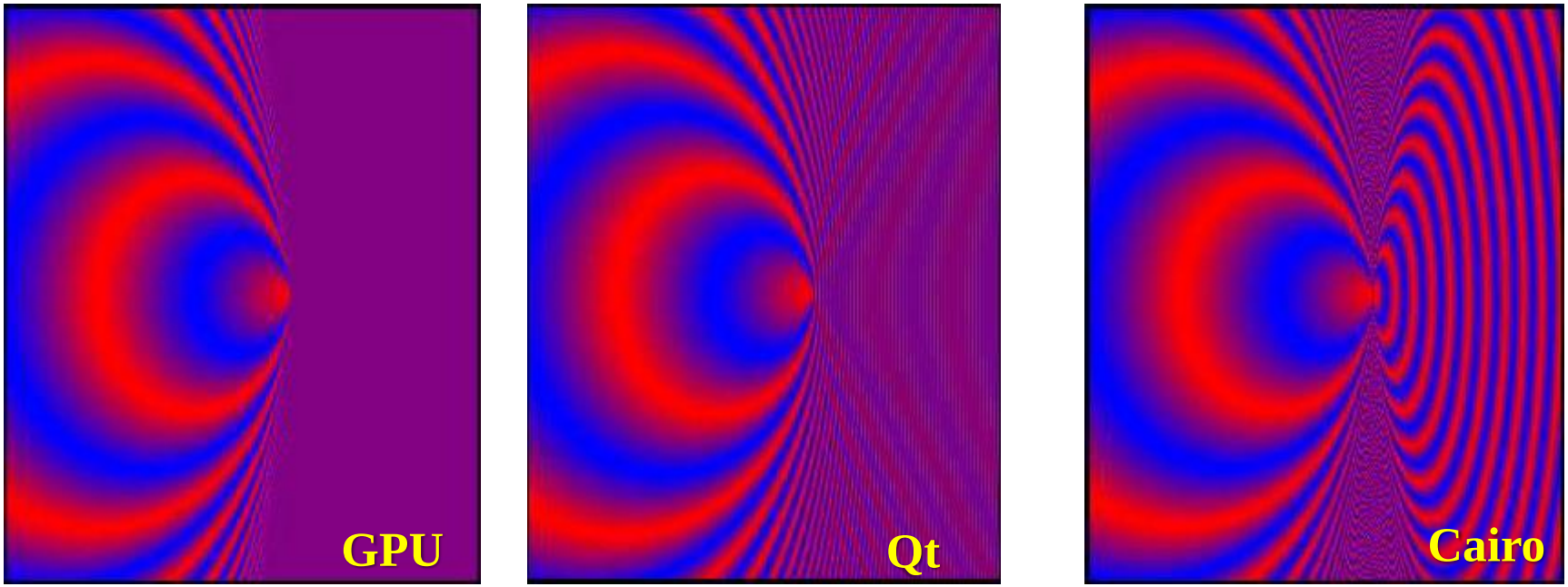
Cairo is designed to produce consistent output on all output media while taking advantage of display hardware acceleration when available (eg. through the X Render E

The cairo API provides operations similar to the drawing operators of PostScript and PDF. Operations in cairo including stroking and filling cubic Bézier splines, transfo and compositing translucent images, and antialiased text rendering. All drawing operations can be transformed by any affine transformation (scale, rotation, shear, etc.)

# VENTAJAS DE LA GPU

- Eficiencia en aplicar filtros a la imagen
  - Mapping de textura de forma rápida
  - Mipmapping
  - Filtros anisotrópicos
  - Modos de “wrap”

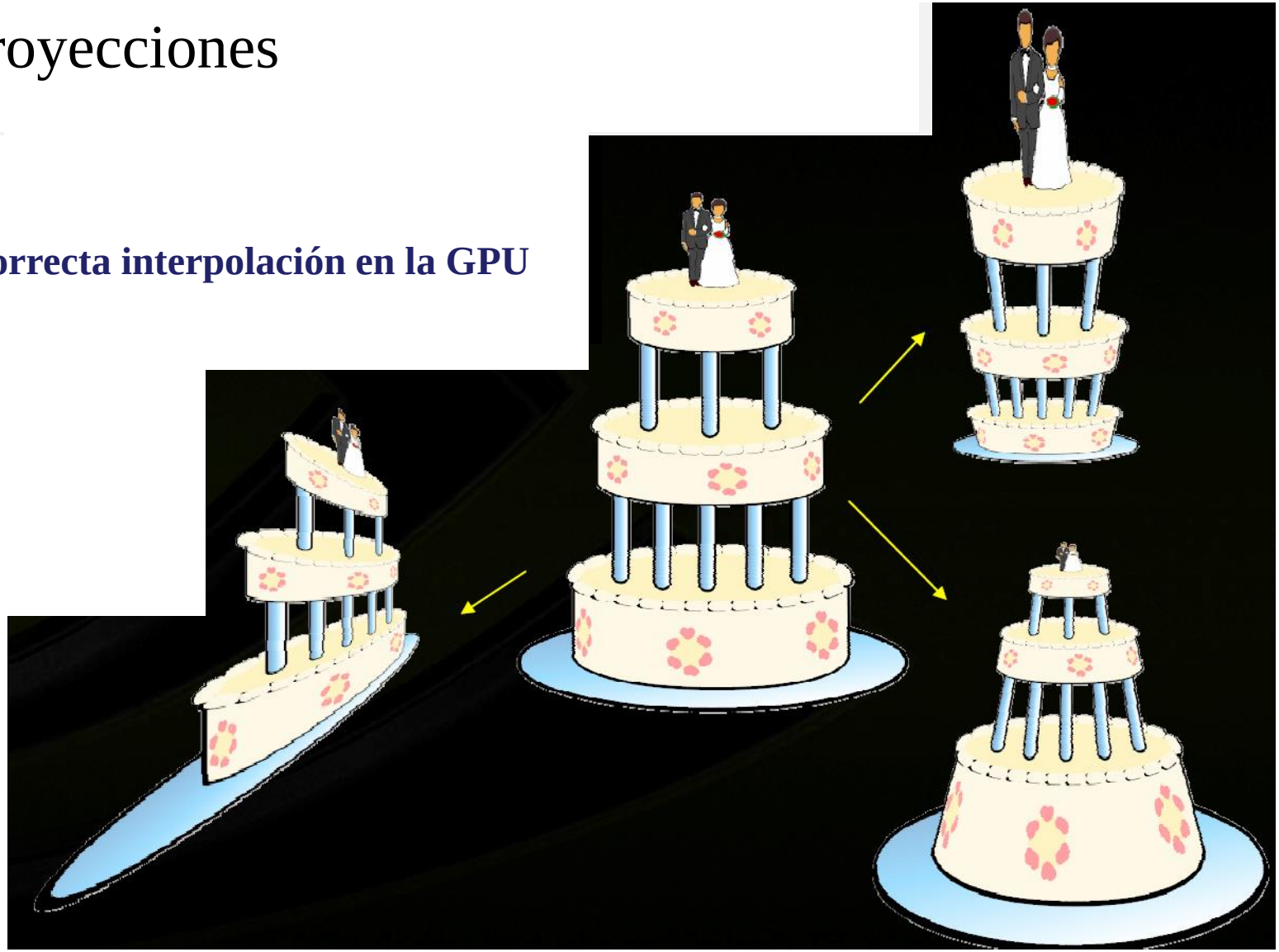
**Patrón de Moiré**



# VENTAJAS DE LA GPU

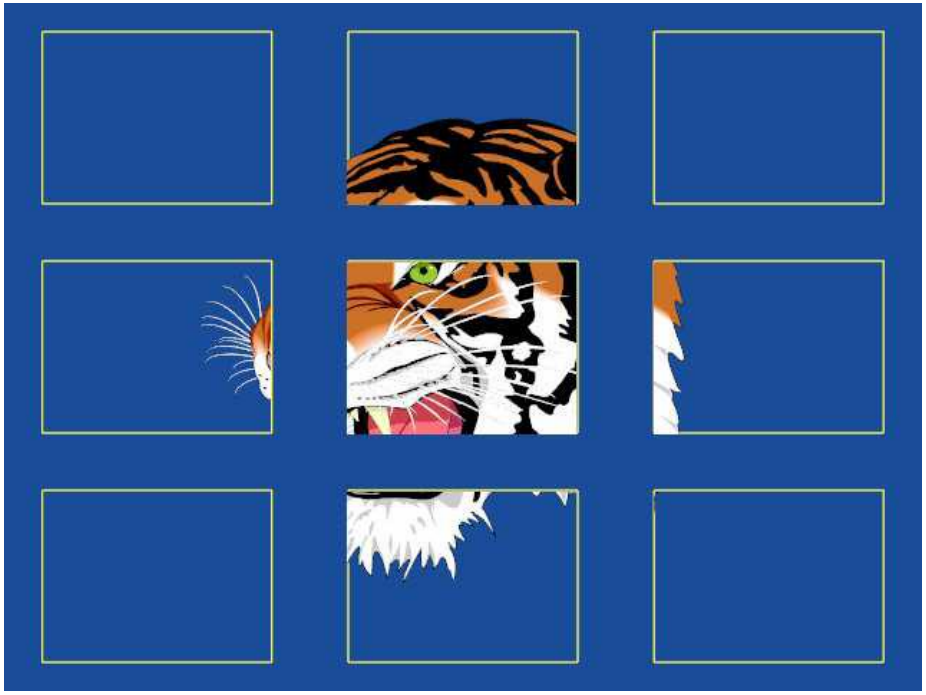
- Proyecciones

**Correcta interpolación en la GPU**

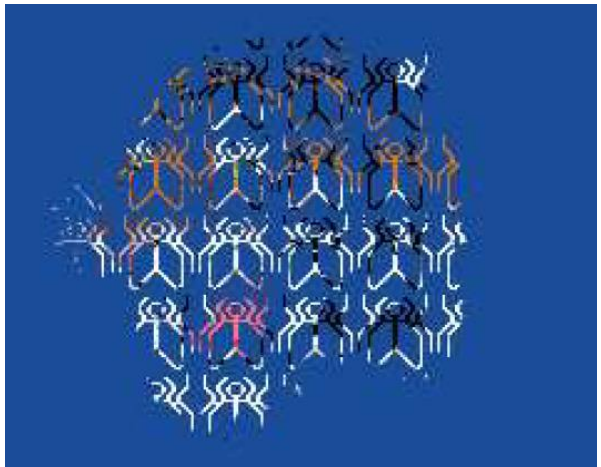


# VENTAJAS DE LA GPU

- Aplicación del scissor para varias regiones

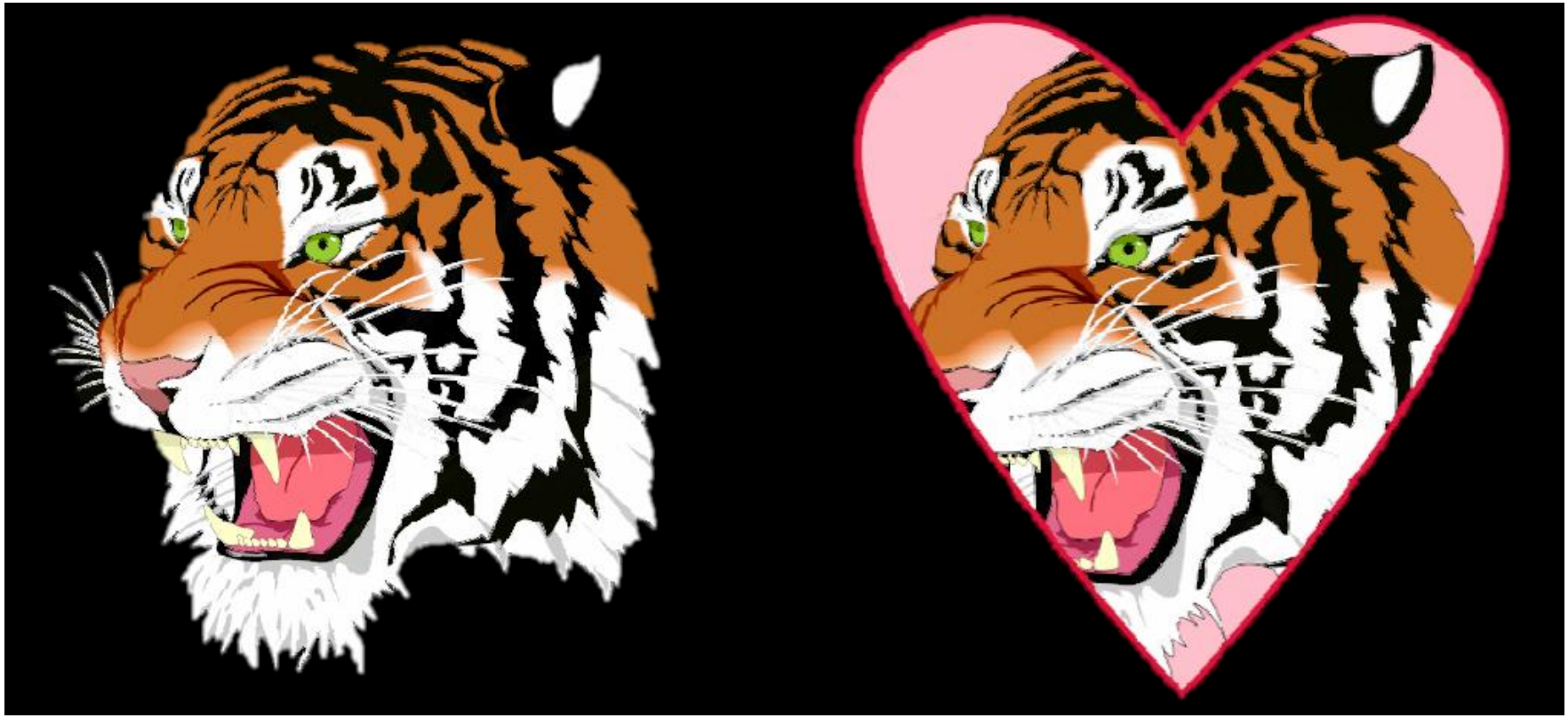


- Patrones de polígonos (punteados)



# VENTAJAS DE LA GPU

- Clipping de un path con otro path

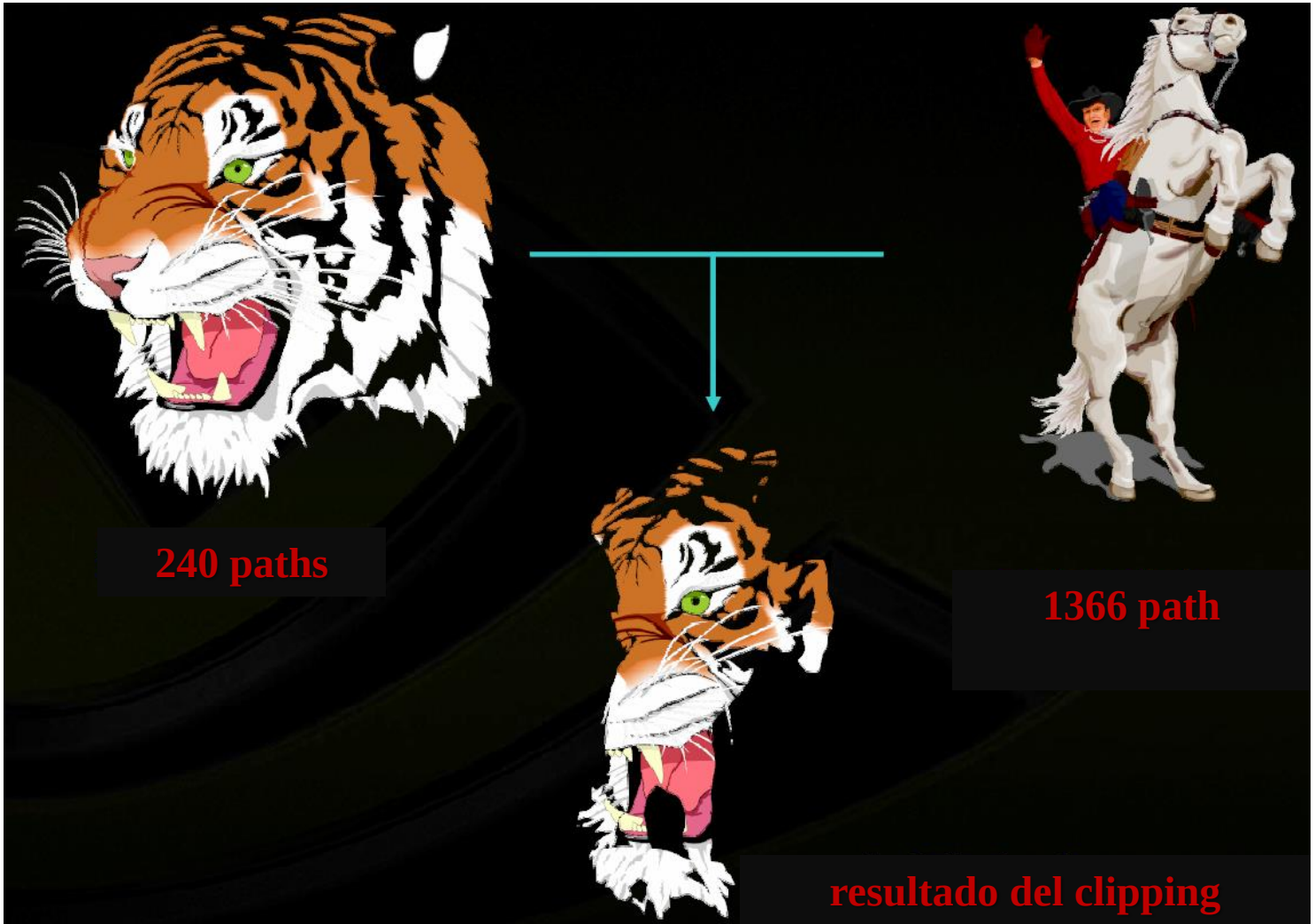


**Path original**

**Clipping de un path en forma de corazón con el path anterior**

# VENTAJAS DE LA GPU

- Clipping de un path con otro path



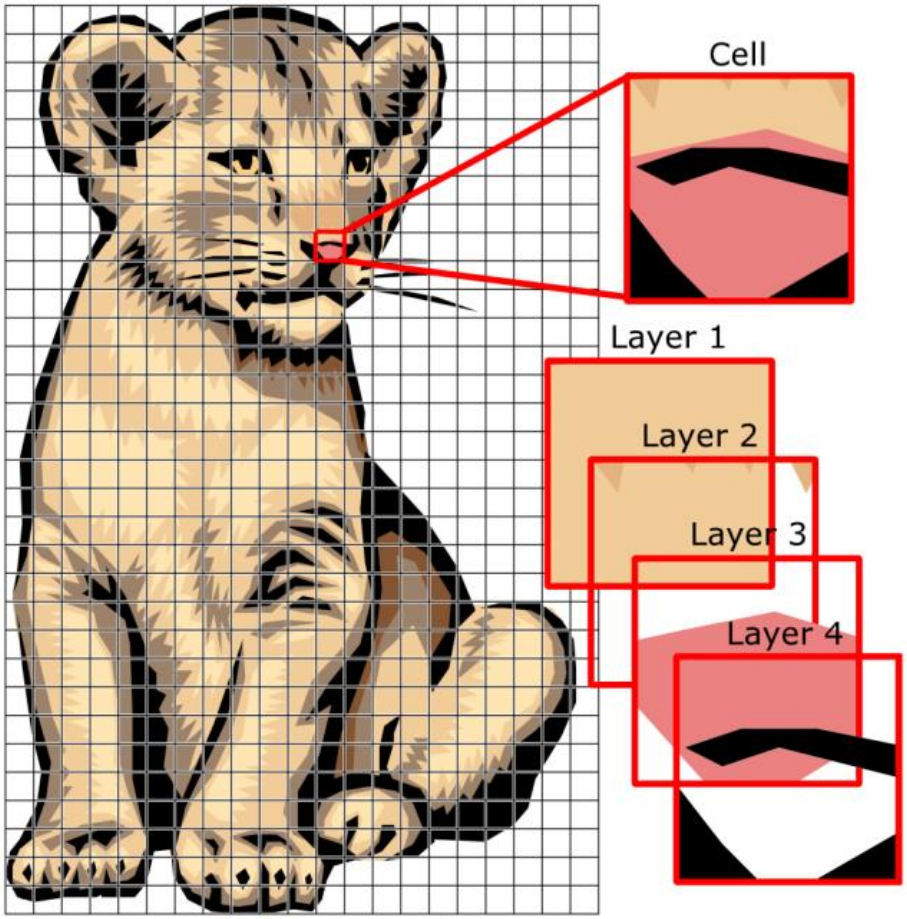
# VENTAJAS DE LA GPU

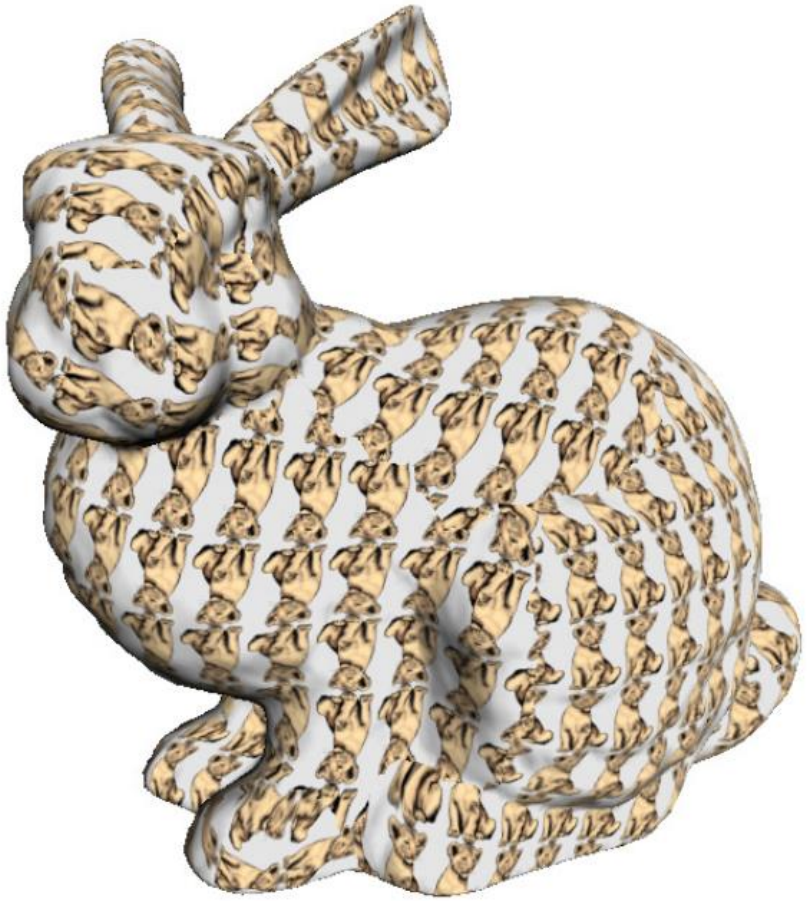
- Acceso al procesador de fragmentos
  - Shader en Cg o HLSL
- Los programas de vértices, geometría y teselación son ignorados


- Bump mapping, shadow mapping, etc.
- Mezcla de objetos 3D y paths en un mismo buffer

# TRABAJOS EN PATH RENDERING

- Random-Access Rendering of General Vector Graphics
  - Diego Nehab, Hugues Hoppe. TOG, 27(5), 2008



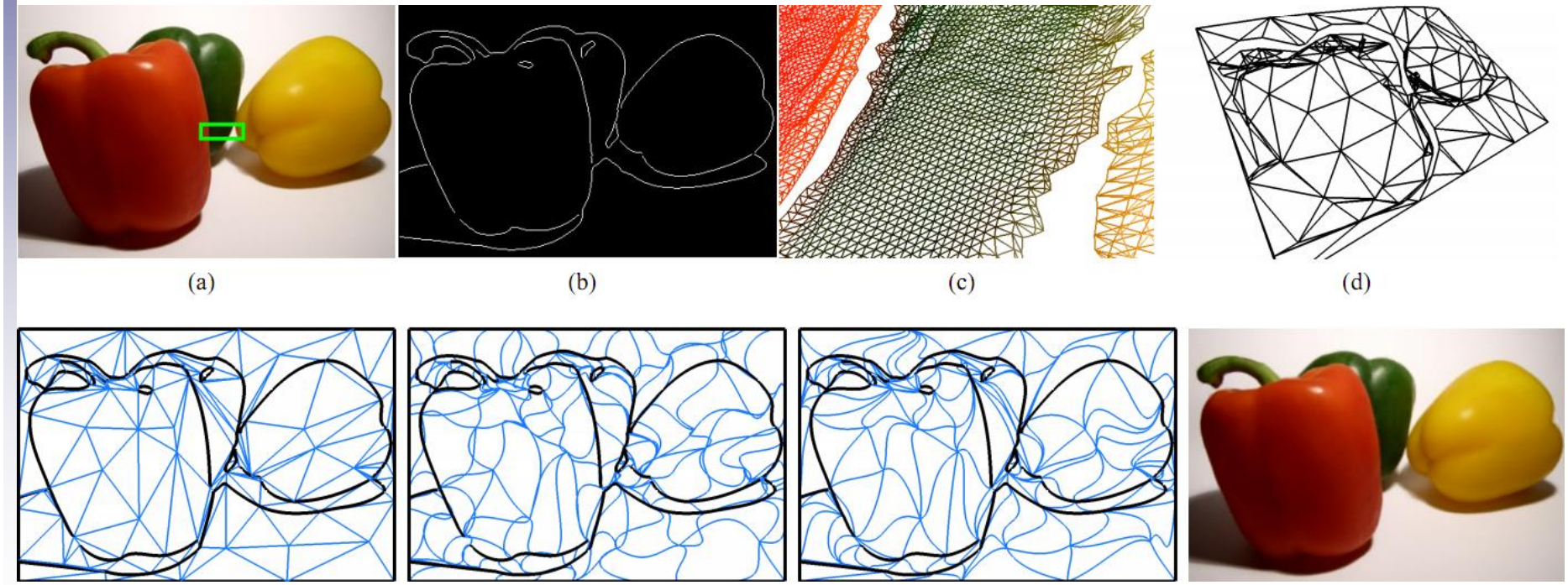



# TRABAJOS EN PATH RENDERING

- Patch-Based Image Vectorization with Automatic Curvilinear Feature Alignment
  - Tian Xa, Binbin Liao, Yizhou Yu. TOG, 28(5), 2009

# TRABAJOS EN PATH RENDERING

- Patch-Based Image Vectorization with Automatic Curvilinear Feature Alignment
  - Tian Xa, Binbin Liao, Yizhou Yu. TOG, 28(5), 2009

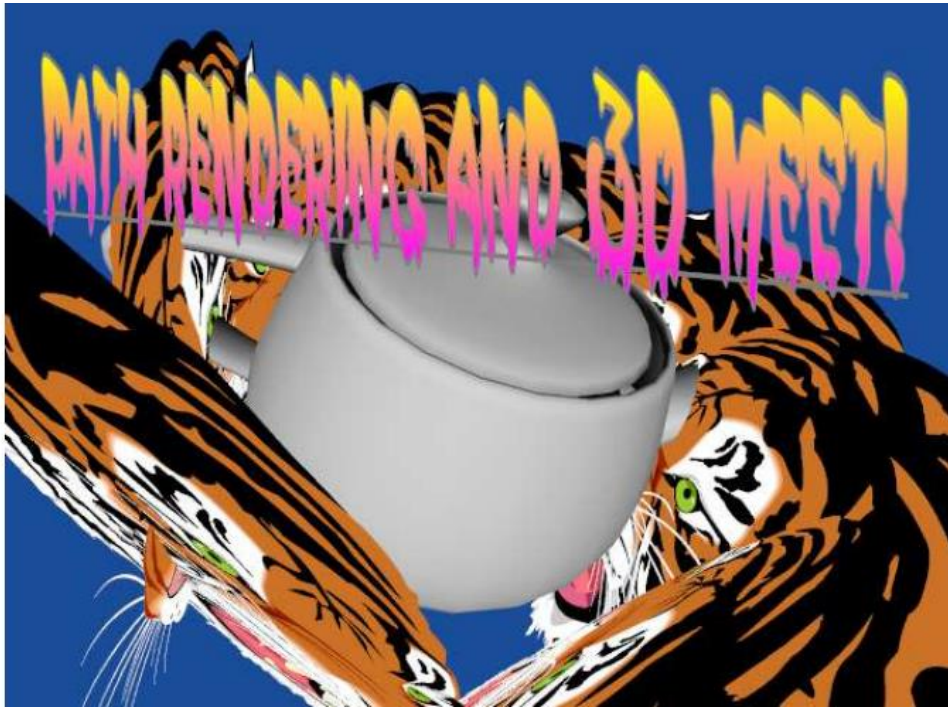


# TRABAJOS EN PATH RENDERING

- A subdivision-Based Representation for Vector Image Editing
  - Zicheng Liao, Hughes Hoppes, David Forsthy, Yizhou Yu. TVCG, Feb., 2012

# TENDENCIAS EN PATH RENDERING

- Mezclar path rendering y gráficos 3D
  - Todo acelerado en la GPU

# Tendencias en Path Rendering

- Gráficos de resolución independiente en 2D
  - Reemplazar los bitmaps
  - Tabletas, smart phones, etc.
- Pantallas más “densas”
  - Pantalla IPS (in-plane switching)
  - Touch screen grandes
  - Más píxeles a dibujar
- Más interactividad
  - PDFs estáticos vs. contenido HTML5 interactivo
  - Interacción táctil solicita baja latencia
  - Path rendering necesita ser más rápido

# IDEAS FINALES

- Path Rendering es un enfoque para gráficos 2D independientes de la resolución
- Especificación basada en paths
- Un path consiste en una serie de comandos para conectar líneas, curvas y arcos
- NV_PATH_RENDERING, extensión de OGL para GPU's de NVIDIA con capacidad CUDA
- Aprovechar las ventajas de las GPUs actuales
- Diversos estudios basados enteramente en el pipeline gráfico programable

# PATH RENDERING