

多媒体技术



多媒体程序设计基础

河南大学计算机与信息科学学院

刘扬

2017/11/28 上午11时25分17秒

目录

- 1、多媒体编程技术介绍
- 2、MM API
- 3、MCI
- 4、GDI+
- 5、DirectX
- 6、XNA & WPF
- 7、Speech SDK
- 8、OpenGL/OpenAL/OpenCV/OpenCL/OpenAI
- 9、JOGL & JMF
- 10、MM Cloud Services

1、多媒体编程技术介绍

■ 多媒体编程内容

- 1、数字图形与图像软件设计技术
 - 图像显示
 - 图形绘制
 - 图像处理
 - 图像识别
- 2、数字音频与MIDI软件开发技术
 - 音频播放
 - 音频信号处理
 - 语音识别
 - MIDI软件设计
 - 语音合成
- 3、数字视频与动画软件开发技术
 - 视频播放
 - 视频编辑与同步
 - 动画生成

常见的多媒体编程技术

■ WINDOWS MMAPI

- GDI, GDI+, 图像获取:WIA, Windows Image Acquisition, Graphics and Imaging(gdi32.dll, 284个函数)(GdiPlus.dll)
- Audio, Video, and Joystick control(winmm.dll, 133个函数)

■ MCI COM

■ MM SDK

- Windows Media Services SDK
- Windows Media Player SDK
- Windows Media Format SDK
- Windows Media Encoder SDK
- Microsoft Secure Audio Path SDK
- Speech SDK
- Windows SDK(Platform SDK)
- Media Foundation
- RTC Client API
- NetMeeting SDK

■ DirectX, XNA

■ OpenGL

■ JMF

■ JOGL

[查](#) · [论](#) · [编](#) · [历](#)

微软应用程式开发接口与框架

[隐藏▲](#)

图形接口	桌面视窗管理员 ▪ DirectX ▪ Direct3D ▪ GDI ▪ WPF ▪ Windows色彩系统 ▪ Windows Image Acquisition ▪ Windows Imaging Component
音乐与音效	DirectSound ▪ DirectMusic ▪ DirectX外挂 ▪ XACT ▪ Speech API
多媒体	DirectShow ▪ DirectX Media Objects ▪ DirectX Video Acceleration ▪ Windows Media ▪ Media Foundation ▪ Image Mastering API
Web	MSHTML ▪ 微软XML剖析器 ▪ RSS Platform ▪ JScript ▪ VBScript ▪ 浏览器协助工具物件 ▪ XMLHTTP ▪ SideBar小工具
资料存取	资料存取元件集 ▪ 延伸储存引擎 ▪ ADO.NET ▪ ADO.NET Entity Framework ▪ ADO.NET Data Services ▪ Microsoft Sync Framework ▪ Jet数据库引擎 ▪ OLE DB
网络	Winsock (LSP) ▪ Winsock Kernel ▪ Filtering Platform ▪ 网络驱动程序接口规格(NDIS) ▪ Windows Rally ▪ 智慧型背景传输服务 ▪ Windows Vista 对等通讯 API ▪ Active Directory Service Interface
通讯	讯息 API ▪ 电话 API
管理	Win32 console ▪ Windows Script Host ▪ WMI ▪ Windows PowerShell ▪ 工作排程器 ▪ Offline Files ▪ Shadow Copy ▪ Windows Installer ▪ Windows错误回报 ▪ 事件记录器 ▪ Common Log File System
元件与模型	COM ▪ COM+ ▪ ActiveX ▪ Distributed COM ▪ .NET Framework
函式库	MFC ▪ ATL ▪ WTL
驱动程式开发	Windows Driver Model (Broadcast Driver Architecture) ▪ Windows Driver Foundation (KMDF ▪ UMDf)
安全性	Crypto API (CAPICOM) ▪ Windows CardSpace ▪ Data Protection API ▪ Security Support Provider Interface
.NET	.NET Framework ▪ ASP.NET ▪ ASP.NET AJAX ▪ ADO.NET ▪ Remoting ▪ WPF ▪ WF ▪ WCF ▪ Windows CardSpace ▪ XNA ▪ Silverlight ▪ Task Parallel Library
软件工厂	EFx Factory ▪ Enterprise Library ▪ 复合使用者接口 ▪ CCF ▪ CSF
行程间通讯	MSRPC ▪ 具名管道 ▪ 内存对映档 ▪ 动态资料交换 ▪ MailSlot
可协助性	Active Accessibility ▪ 使用者接口自动化
文字与多语系支援	Text Services Framework ▪ Text Object Model ▪ 输入法编辑器 ▪ 语言接口套件 ▪ 多语系使用者接口 ▪ Uniscribe
游戏开发	Direct3D ▪ D3DX ▪ DirectSound ▪ DirectInput ▪ DirectPlay ▪ DirectMusic ▪ Managed DirectX ▪ Microsoft XNA

v · d · e

Java

[hide]

Java platform

[Java Language](#) · [JVM](#) · [Java Platform, Micro Edition](#) · [Java SE](#) · [Java EE](#) · [Java Card](#)

Sun technologies

[Squawk](#) · [Java Development Kit](#) · [OpenJDK](#) · [Java Virtual Machine](#) · [JavaFX](#)

Platform technologies

[Applets](#) · [Servlets](#) · [MIDlets](#) · [jsp](#) · [Web Start \(jnlp\)](#)

Major third-party technologies

[JRockit](#) · [GNU Classpath](#) · [Kaffe](#) · [TopLink](#) · [Apache Harmony](#) · [Apache Struts](#) · [Spring framework](#) · [Hibernate](#) · [JBoss application server](#) · [Tapestry](#) · [Jazelle](#)

History

[Java version history](#) · [Criticism of Java](#) · [Java Community Process](#) · [Sun Microsystems](#) · [Free Java implementations](#)

Major programming languages

[Java Tcl](#) · [Jython](#) · [JRuby](#) · [BeanShell](#) · [Clojure](#) · [Groovy](#) · [Rhino](#) · [Scala](#) · [Processing](#) · [more...](#)

Java conferences

[JavaOne](#)



2、多媒体API函数

- **API是应用程序编程接口 (Application Program Interface)的缩写，这是一组供应用程序使用的命令，用以向计算机的操作系统请求或执行更低级的设备访问操作。**
- **多媒体API函数**
 - 与MCI有关的高级函数是：
 - mciSendString ()： 传送指令字符串给MCI;
 - mciExecute ()： 可视为mciSendString的简化函数，如果无法执行，会以一个对话框显示错误信息;
 - mciGetErrorString ()： 解释MCI错误代码所表示的意思。

■ 波形音频函数

- 波形音频函数包括高层波形音频函数和低层音频函数
- 高层波形音频函数

函数	描 述
MessageBeep	播放对应给定的系统报警波形声音
PlaySound	播放对应给定的文件名或注册项的波形声音
sndPlaySound	播放对应给定的文件名或注册项的波形声音

■ 波形音频函数

● 低层波形音频函数

-低层波形音频函数以wav为前缀，共有6类，提供如下服务：

查询音频设备

打开和关闭设备驱动程序

分配和准备音频数据块

管理音频数据块

应用MMTIME结构

处理错误

Audio, Video, and Joystick control (winmm.dll)

- joyGetDevCaps
- joyGetNumDev
- joyGetNumDevs
- joyGetPos
- joyGetPosEx
- joyGetThreshold
- joyReleaseCapture
- joySetCapture
- joySetThreshold
- auxGetDevCaps
- auxGetNumDevs
- auxGetVolume
- auxOutMessage
- auxSetVolume
- mciExecute
- mciGetCreatorTask
- mciGetDeviceID
- mciGetDeviceIDFromElementID
- mciGetErrorString
- mciGetYieldProc
- mciSendCommand
- mciSendString
- mciSetYieldProc
- CloseDriver
- DefDriverProc
- DrvGetModuleHandle
- GetDriverModuleHandle

- `midiConnect`
- `midiDisconnect`
- `midiInAddBuffer`
- `midiInClose`
- `midiInGetDevCaps`
- `midiInGetErrorText`
- `midiInGetID`
- `midiInGetNumDevs`
- `midiInMessage`
- `midiInOpen`
- `midiInPrepareHeader`
- `midiInReset`
- `midiInStart`
- `midiInStop`
- `midiInUnprepareHeader`
- `midiOutCacheDrumPatches`
- `midiOutCachePatches`
- `midiOutClose`
- `midiOutGetDevCaps`
- `midiOutGetErrorText`
- `midiOutGetID`
- `midiOutGetNumDevs`
- `midiOutGetVolume`
- `midiOutLongMsg`
- `midiOutMessage`
- `midiOutOpen`
- `midiOutPrepareHeader`
- `midiOutReset`
- `midiOutSetVolume`
- `midiOutShortMsg`
- `midiOutUnprepareHeader`

- midiStreamClose
- midiStreamOpen
- midiStreamOut
- midiStreamPause
- midiStreamPosition
- midiStreamProperty
- midiStreamRestart
- midiStreamStop
- mixerClose
- mixerGetControlDetails
- mixerGetDevCaps
- mixerGetID
- mixerGetLineControls
- mixerGetNumDevs
- mixerMessage
- mixerOpen
- mixerSetControlDetails
- mmioAdvance
- mmioAscend
- mmioClose
- mmioCreateChunk
- mmioDescend
- mmioFlush
- mmioGetInfo
- mmioInstallIOProc
- mmioInstallIOProcA
- mmioOpen
- mmioRead
- mmioRename
- mmioSeek
- mmioSendMessage
- mmioSetBuffer
- mmioSetInfo
- mmioStringToFOURCC
- mmioWrite

- sndPlaySound
- timeBeginPeriod
- timeEndPeriod
- timeGetDevCaps
- timeGetSystemTime
- timeGetTime
- timeKillEvent
- timeSetEvent
- waveInAddBuffer
- waveInClose
- waveInGetDevCaps
- waveInGetErrorText
- waveInGetID
- waveInGetNumDevs
- waveInGetPosition
- waveInMessage
- waveInOpen
- waveInPrepareHeader
- waveInReset
- waveInStart
- waveInStop
- waveInUnprepareHeader
- waveOutBreakLoop
- waveOutClose
- waveOutGetDevCaps
- waveOutGetErrorText
- waveOutGetID
- waveOutGetNumDevs
- waveOutGetPitch
- waveOutGetPlaybackRate
- waveOutGetPosition
- waveOutGetVolume
- waveOutMessage
- waveOutOpen
- waveOutPause
- waveOutPrepareHeader
- waveOutReset
- waveOutRestart
- waveOutSetPitch
- waveOutSetPlaybackRate
- waveOutSetVolume
- waveOutUnprepareHeader
- waveOutWrite

3、GDI+

- GDI+ 是 Microsoft® Windows® XP 操作系统的子系统，负责在屏幕和打印机上显示信息。顾名思义，GDI+ 是 GDI（Windows 早期版本提供的图形设备接口）的后续版本。GDI+ 是一种应用程序编程接口（API），通过一套部署为托管代码的类来展现。这套类被称为 GDI+ 的“托管类接口”。

Graphics and Imaging (gdi32.dll)

AbortDoc	CreateDC	ExtCreateRegion	GetEnhMetaFile	GetTextColor	PolyPolygon	SetMapperFlags
AbortPath	CreateMetaFile	ExtEscape	GetEnhMetaFileBits	GetTextExtentExPoint	PolyPolyline	SetMetaFileBitsEx
AddFontResource	CreatePalette	ExtFloodFill	GetEnhMetaFileDescription	GetTextExtentPoint	PolyTextOut	SetMetaRgn
AngleArc	CreatePatternBrush	ExtSelectClipRgn	GetEnhMetaFileHeader	GetTextExtentPoint32	PlInRegion	SetStretchBlt
AnimatePalette	CreatePen	ExtTextOut	GetEnhMetaFilePaletteEntries	GetTextFace	PlVisible	SetPaletteEntries
Arc	CreatePenIndirect	FillPath	GetFontData	GetTextMetrics	RealizePalette	SetPixel
ArcTo	CreatePolygonRgn	FillRgn	GetFontLanguageInfo	GetViewportExtEx	Rectangle	SetPixelFormat
BeginPath	CreatePolyPolygonRgn	FltBrushOrgEx	GetGlyphOutline	GetViewportOrgEx	RgnInRegion	SetPixelV
BitBlt	CreateRgnRgn	FlattenPath	GetGraphicMode	GetWindowExtEx	RgnVisible	SetPolyFillMode
CancelDC	CreateRgnRgnIndirect	FloodFill	GetICProfile	GetWindowOrgEx	RemoveFontResource	SetRgnRgn
CheckColorsInUsage	CreateRoundRectRgn	FrameRgn	GetKerningPairs	GetWinMetaFileBits	ResetDC	SetROP2
ChoosePixelFormat	CreateScalableFontResource	GdiComment	GetLogColorSpace	GetWorldTransform	ResizePalette	SetStretchBltMode
Chord	CreateSolidBrush	GdiFlush	GetMapMode	IntersectClipRect	RestoreDC	SetSystemPaletteUse
CloseEnhMetaFile	DeleteColorSpace	GdiGetBatchLimit	GetMetaFile	InvertRgn	RoundRect	SetTextAlign
CloseFigure	DeleteDC	GdiSetBatchLimit	GetMetaFileBitsEx	LineDD	SaveDC	SetTextCharacterExtra
CloseMetaFile	DeleteEnhMetaFile	GetArcDirection	GetMetaRgn	LineDDa	ScaleViewportExtEx	SetTextColor
ColorMatchToTarget	DeleteMetaFile	GetAspectRatioFilterEx	GetStretchBlt	LineTo	ScaleWindowExtEx	SetTextJustification
CombineRgn	DeleteObject	GetBitmapBits	GetNearestColor	LPltoDP	SelectClipPath	SetViewportExtEx
CombineTransform	DescribePixelFormat	GetBitmapDimensionEx	GetNearestPaletteIndex	MaskBlt	SelectClipRgn	SetViewportOrgEx
CopyEnhMetaFile	DlgDirList	GetBkColor	GetObject_Rename	ModifyWorldTransform	SelectObject	SetWindowExtEx
CopyMetaFile	DlgDirListComboBox	GetBkMode	GetObjectType	MoveToEx	SelectPalette	SetWindowOrgEx
CreateBitmap	DlgDirSelectComboBoxEx	GetBoundRect	GetOutlineTextMetrics	OffsetClipRgn	SetAbortProc	SetWinMetaFileBits
CreateBitmapIndirect	DlgDirSelectEx	GetBrushOrgEx	GetPaletteEntries	OffsetRgn	SetArcDirection	SetWorldTransform
CreateBrushIndirect	DPltoLP	GetCharBkWidths	GetPath	OffsetViewportOrgEx	SetBitmapBits	StartDoc
CreateColorSpace	DrawEscape	GetCharBkWidthsFloat	GetPixel	OffsetWindowOrgEx	SetBitmapDimensionEx	StartPage
CreateCompatibleBitmap	Ellipse	GetCharacterPlacement	GetPixelFormat	PaintRgn	SetBkColor	StretchBlt
CreateCompatibleDC	EndDoc	GetCharWidth	GetPolyFillMode	PatBlt	SetBkMode	StretchDIBits
CreateDC	EndPage	GetCharWidth32	GetRasterizerCaps	PathToRegion	SetBoundRect	StrokeAndFillPath
CreateDIBitmap	EndPath	GetCharWidthFloat	GetRegionData	Pie	SetBrushOrgEx	StrokePath
CreateDIBPatternBrush	EnumEnhMetaFile	GetClipBox	GetRgnBox	PlayEnhMetaFile	SetColorAdjustment	SwapBuffers
CreateDIBPatternBrushPt	EnumFontFamilies	GetClipRgn	GetROP2	PlayEnhMetaFileRecord	SetColorSpace	TextOut
CreateDIBSection	EnumFontFamiliesEx	GetColorAdjustment	GetSticObject	PlayMetaFile	SetDeviceGammaRamp	TranslateCharsetInfo
CreateDiscardableBitmap	EnumFonts	GetColorSpace	GetStretchBltMode	PlayMetaFileRecord	SetDIBColorTable	UnrealizeObject
CreateEllipticRgn	EnumICProfiles	GetCurrentObject	GetSystemPaletteEntries	PlgBlt	SetDIBits	UpdateColors
CreateEllipticRgnIndirect	EnumMetaFile	GetCurrentPositionEx	GetSystemPaletteUse	PolyBezier	SetDIBitsToDevice	WidenPath
CreateEnhMetaFile	EnumObjects	GetDCOrgEx	SetTextAlign	PolyBezierTo	SetEnhMetaFileBits	
CreateFont	EqualRgn	GetDeviceCaps	SetTextCharacterExtra	PolyDraw	SetGraphicMode	
CreateFontIndirect	Escape	GetDeviceGammaRamp	SetTextCharacterExtra	Polygon	SetICMode	
CreateHalftonePalette	ExcludeClipRect	GetDIBColorTable	SetTextCharset	Polyline	SetICProfile	
CreateHatchBrush	ExtCreatePen	GetDIBits	SetTextCharsetInfo	PolylineTo	SetMapMode	

3、GDI+ 的组成

■ 二维矢量图形

- 矢量图形包括坐标系统中的系列点指定的绘图基元（例如，直线、曲线和图形）。
- GDI+ 提供了存储基元自身相关信息的类（和结构）、存储基元绘制方式相关信息的类，以及实际进行绘制的类。例如，**Rectangle** 结构存储矩形的位置和尺寸；**Pen** 类存储有关线条颜色、线条粗细和线型的信息；而 **Graphics** 类具有用于绘制直线、矩形、路径和其他图形的方法。还有几种 **Brush** 类，它们存储有关如何使用颜色或图案来填充封闭图形和路径的信息。

■ 图像处理

- GDI+ 提供了 **Bitmap** 类，可用于显示、操作和保存位图。

■ 版式

- 版式关系到使用各种字体、字号和样式来显示文本。可以使文本在屏幕上显示进行像素的锯齿消除,平滑。

基于类的接口结构

- GDI+ 的托管类接口包含大约 60 个类、50 个枚举和 8 个结构。Graphics 类是 GDI+ 的核心功能，它是实际绘制直线、曲线、图形、图像和文本的类。
- 许多类与 Graphics 类一起使用。例如，Graphics.DrawLine 方法接收 Pen 对象，该对象中存有所要绘制的线条的属性（颜色、宽度、虚线线型和外观）。Graphics.FillRectangle 方法可以接收指向 LinearGradientBrush 对象（它使用 Graphics 对象以渐变色填充矩形）的指针。Font 和 StringFormat 对象影响 Graphics 对象绘制文本的方式。Matrix 对象存储并操作 Graphics 对象的全局变形，该对象用于旋转、缩放和翻转图像。

■ Color 结构

■ 公共属性

- A 获取此 Color 结构的 alpha 分量值
- B 获取此 Color 结构的蓝色分量值。
- G 获取此 Color 结构的绿色分量值。
- R 获取此 Color 结构的红色分量值。

■ 公共方法

- GetBrightness 获取此 Color 结构的“色调-饱和度-亮度”（HSB）的亮度值。
- GetHue 获取此 Color 结构的“色调-饱和度-亮度”（HSB）的色调值，以度为单位。
- GetSaturation 获取此 Color 结构的“色调-饱和度-亮度”（HSB）的饱和度值。
- ToArgb 获取此 Color 结构的 32 位 ARGB 值。
- ToString 已重写。将此 Color 结构转换为可读的字符串。

Shlwapi.dll

Header: Declared in Shlwapi.h.

Import Library: Shlwapi.lib.

- ColorHLSToRGB

- Converts colors from hue-luminance-saturation (HLS) to red-green-blue (RGB) format.

- COLORREF ColorHLSToRGB(

- WORD wHue, //HLS hue value.

- WORD wLuminance, //HLS luminance value.

- WORD wSaturation //HLS saturation value.

-);//Returns the RGB value.

- ColorRGBToHLS

- Converts colors from red-green-blue (RGB) to hue-luminance-saturation (HLS) format.

- VOID ColorRGBToHLS(

- COLORREF clrRGB, //[in] RGB color.

- WORD *pwHue, //[out] HLS hue value.

- WORD *pwLuminance, //[out] HLS luminance value.

- WORD *pwSaturation //[out] HLS saturation value.

-);//No return value.

Bitmap Class

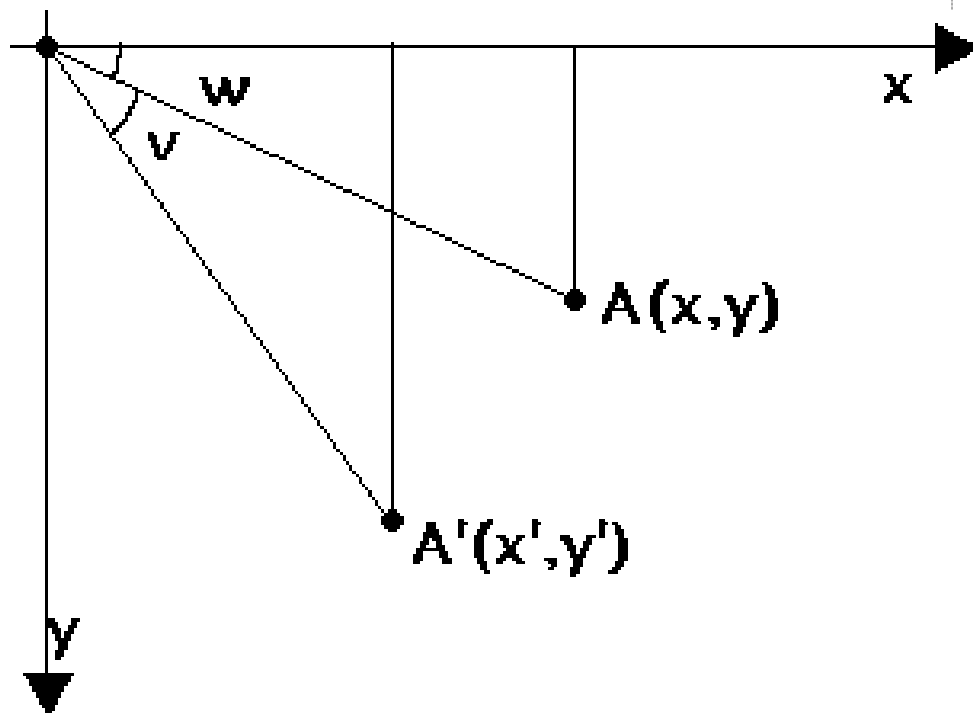
■ Bitmap Class Methods

- FromStream Creates a Bitmap object based on a stream.
- GetPixel Gets the color of a specified pixel in this bitmap.
- SetPixel Sets the color of a specified pixel in this bitmap.
- 可用 **Bitmap** 类来加载和显示光栅图像，还可以利用 **Metafile** 类来加载和显示矢量图像。**Bitmap** 和 **Metafile** 类从 **Image** 类中继承。要显示矢量图像，需要有 **Graphics** 对象和 **Metafile** 对象。要显示光栅图像，需要有 **Graphics** 对象和 **Bitmap** 对象。**Graphics** 对象提供了 **DrawImage** 方法，该方法接收 **Metafile** 或 **Bitmap** 对象作为参数。
- 可用多种图形文件格式（BMP、GIF、JPEG、EXIF、PNG、TIFF 和 ICON）中构造 **Bitmap** 对象。

坐标系统和变形

- GDI+ 提供了全局变形和页面变形，以便您可以使绘制的项目变形（旋转、缩放、平移，等等）。两种变形还允许您使用多种坐标系统。
- GDI+ 使用三个坐标空间：全局、页面和设备。进行 `myGraphics.DrawLine(myPen, 0, 0, 160, 80)` 调用时，传递到 `DrawLine` 方法的点（(0, 0) 和 (160, 80)）位于全局坐标空间中。由于度量单位是像素，所以设备坐标与页面坐标是相同的。如果将度量单位设置为像素以外的其他单位（例如英寸），设备坐标将不同于页面坐标。

坐标系统和变形



- $\tan(w) = y/x, \quad r^2 = x^2 + y^2$

- 旋转角度 v

- $y' = r \sin(w+v)$
 $= \sqrt{x^2 + y^2} \sin(\arctan(y/x) + v)$

- $x' = r \cos(w+v)$
 $= \sqrt{x^2 + y^2} \cos(\arctan(y/x) + v)$

- 特例

- 原点对称 ($v=180$):
 $x' = -x; \quad y' = -y$

- x轴对称: ($x' = x; \quad y' = -y$)

- y轴对称: ($x' = -x; \quad y' = y$)

坐标系统和变形

- 将全局坐标映射到页面坐标的变形称为“全局变形”，保存在 Graphics 类的 Transform 属性中。
- `myGraphics.TranslateTransform(100, 50);`
- `//全局变形是在 x 方向平移 100 个单位、在 y 方向平移 50 个单位。`
- `myGraphics.DrawLine(myPen, 0, 0, 160, 80);`
- `//使用该 Graphics 对象来绘制前图中显示的线条(0, 0)——(60, 80)`

- 将页面坐标映射到设备坐标的变形称为“页面变形”。Graphics 类提供了用于操作页面变形的 PageUnit 和 PageScale 属性。Graphics 类还提供了两个只读属性：DpiX 和 DpiY，可用于检查显示设备每英寸的水平点和垂直点。Graphics 类的 PageUnit 属性可用于指定像素以外的其他度量单位。
- `myGraphics.PageUnit = GraphicsUnit.Inch;`
`myGraphics.DrawLine(myPen, 0, 0, 2, 1);`
- //从 (0, 0) 至 (2, 1) 绘制线条，其中点 (2, 1) 位于点 (0, 0) 的右边 2 英寸和下边 1 英寸处

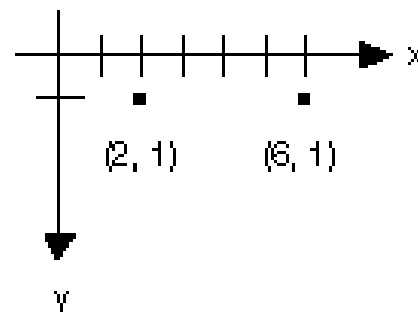
变形的矩阵表示形式

- $m \times n$ 矩阵是以 m 行和 n 列排列的一组数字。 $m \times n$ 矩阵可以与 $n \times p$ 矩阵相乘，得到的结果是 $m \times p$ 矩阵。第一个矩阵的列数必须与第二个矩阵的行数相同。
- 可以将平面中的点视为 1×2 矩阵，可将该点乘以 2×2 矩阵来变形该点。或者加上 1×2 矩阵进行位移

点 (2, 1) 的几个变形

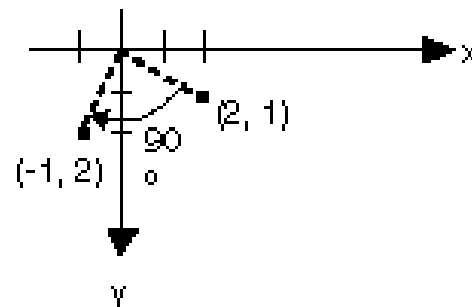
按比例缩放

$$\begin{bmatrix} 2 & 1 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 1 \end{bmatrix}$$



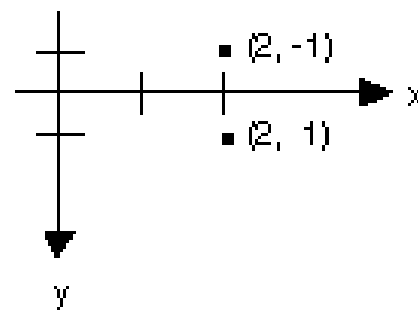
旋转 90°

$$\begin{bmatrix} 2 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 2 \end{bmatrix}$$



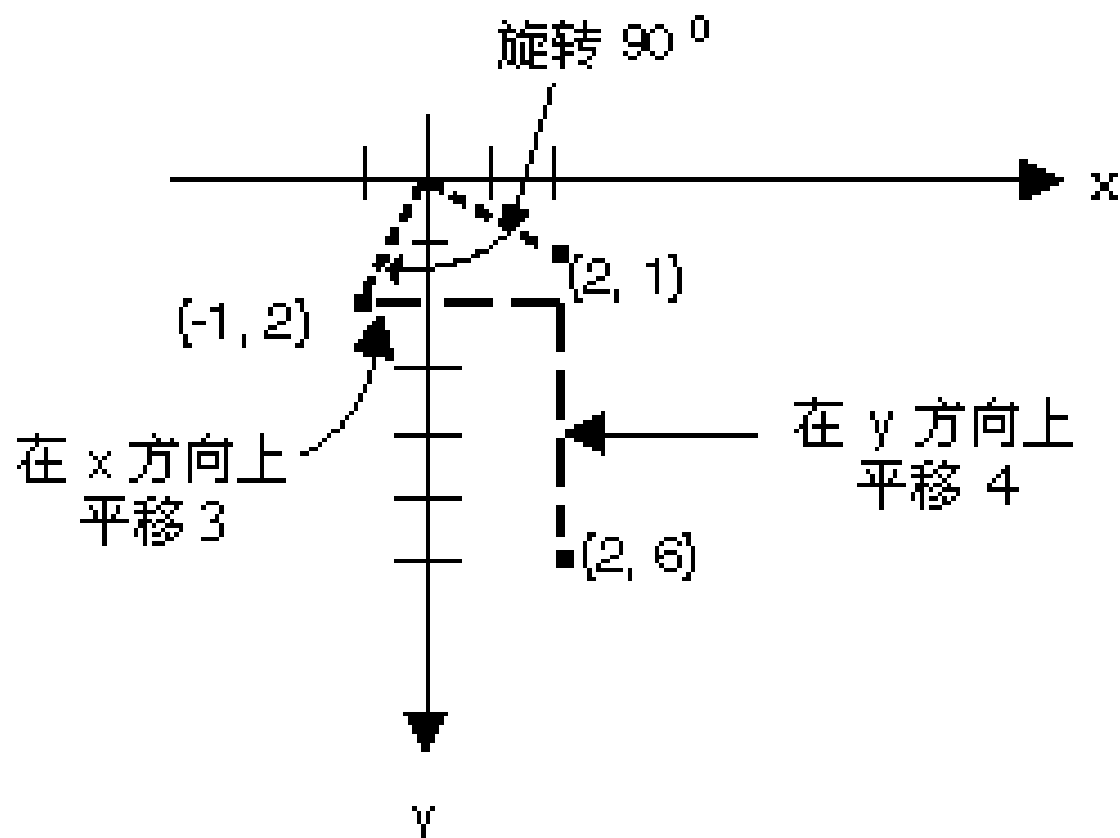
跨 x 轴反射

$$\begin{bmatrix} 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 2 & -1 \end{bmatrix}$$



仿射变形

$$\begin{bmatrix} 2 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} + \begin{bmatrix} 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 6 \end{bmatrix}$$



仿射变形

- 可仿射变形存储于一对矩阵 3×3 矩阵, 要使其起作用, 平面上的点必须存储于有虚拟第 3 坐标的 1×3 矩阵。通常的技术是使所有的第 3 坐标等于 1。例如, 矩阵 $[2 \ 1 \ 1]$ 代表点 $(2, 1)$ 。下图显示了表示为与单个 3×3 矩阵相乘的仿射变形 (旋转 90° 度; 在 x 方向上平移 3 个单位, 在 y 方向上平移 4 个单位)。

$$\begin{bmatrix} 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 3 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 6 & 1 \end{bmatrix}$$

线性部分

平移部分

$$\begin{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 3 & 4 \end{bmatrix} & 1 \end{bmatrix}$$

仿射变形

- GDI+ 中，您可以在 `Matrix` 对象中存储仿射变形。由于表示仿射变形的矩阵第三列总是 $(0, 0, 1)$ ，因此构建 `Matrix` 对象时，只需指定前两列中的 6 个数字。语句 `Matrix myMatrix = new Matrix(0, 1, -1, 0, 3, 4)` 构建了前图中显示的矩阵。
- `Matrix` 类提供了建立复合变形的几种方法：`Multiply`、`Rotate`、`RotateAt`、`Scale`、`Shear`和 `Translate`。

仿射变形

- `Matrix myMatrix = new Matrix();` //复合变形
- `myMatrix.Rotate(30);` //先旋转 30 度
- `myMatrix.Scale(1, 2, MatrixOrder.Append);`
- //在 y 方向上缩放 2 倍
- `myMatrix.Translate(5, 0, MatrixOrder.Append);` //x 方向平移 5 个单位

$$\begin{bmatrix} \cos 30^\circ & 2\sin 30^\circ & 0 \\ -\sin 30^\circ & 2\cos 30^\circ & 0 \\ 5 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 0.866 & 1.0 & 0 \\ -0.5 & 1.73 & 0 \\ 5 & 0 & 1 \end{bmatrix}$$

全局变形和局部变形

- 全局变形是应用于由给定的 Graphics 对象绘制的每个项目的变形。要创建全局变形，可构建 Graphics 对象，然后操作其 Transform 属性。Transform 属性是 Matrix 对象，因此它能够保存仿射变形的任何序列。存储在 Transform 属性中的变形被称为全局变形。
- Graphics 类提供了建立复合全局变形的几种方法：
MultiplyTransform、RotateTransform、ScaleTransform 和 TranslateTransform。
- `myGraphics.DrawEllipse(myPen, 0, 0, 100, 50);` //在创建全局变形之前绘制椭圆
- `myGraphics.ScaleTransform(1, 0.5f);` //在 y 方向上缩放 0.5 倍
- `myGraphics.TranslateTransform(50, 0, MatrixOrder.Append);` //再在 x 方向平移 50 个单位

全局变形和局部变形

- `myGraphics.RotateTransform(30, MatrixOrder.Append);` // 旋转 30 度。
- `myGraphics.DrawEllipse(myPen, 0, 0, 100, 50);` // 在创建全局变形之后绘制椭圆
- // 椭圆围绕坐标系统的原点旋转，该原点在工作区的左上角。与椭圆围绕其自身中心旋转相比，这样会产生不同的结果。

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 50 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 30^\circ & \sin 30^\circ & 0 \\ -\sin 30^\circ & \cos 30^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos 30^\circ & \sin 30^\circ & 0 \\ -0.5 \sin 30^\circ & 0.5 \cos 30^\circ & 0 \\ 50 \cos 30^\circ & 50 \sin 30^\circ & 1 \end{bmatrix}$$

按比例缩放 平移 旋转

全局变形和局部变形

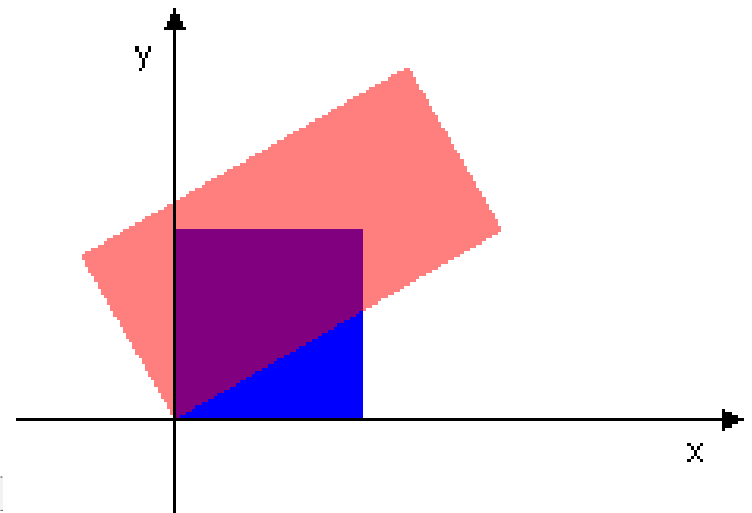
- 局部变形是应用于要绘制的特定项目的变形。例如，GraphicsPath 对象具有 Transform 方法，此方法可用于将该路径的数据点变形。
- `Matrix myMatrix = new Matrix();`
- `myMatrix.Rotate(45);`
- `myGraphicsPath.Transform(myMatrix);`
- `myGraphics.DrawRectangle(myPen, 10, 10, 100, 50);`
- `//绘制一个没有变形的矩形`
- `myGraphics.DrawPath(myPen, myGraphicsPath);`
- `//绘制一个有旋转变形路径的矩形`

全局变形和局部变形

- 全局变形可与局部变形合并，以得到多种结果。例如，全局变形可用于修正坐标系统，而局部变形可用于旋转和缩放在新坐标系统上绘制的对象。
- `Matrix myMatrix = new Matrix(1, 0, 0, -1, 0, 0);`
- `myGraphics.Transform = myMatrix;`
- `myGraphics.TranslateTransform(200, 150, MatrixOrder.Append); // Create the path.`
- `GraphicsPath myGraphicsPath = new GraphicsPath();`
- `Rectangle myRectangle = new Rectangle(0, 0, 60, 60);`
- `myGraphicsPath.AddRectangle(myRectangle); // Fill the path on the new coordinate system. No local transformation`
- `myGraphics.FillPath(mySolidBrush1, myGraphicsPath); // Set the local transformation of the GraphicsPath object.`

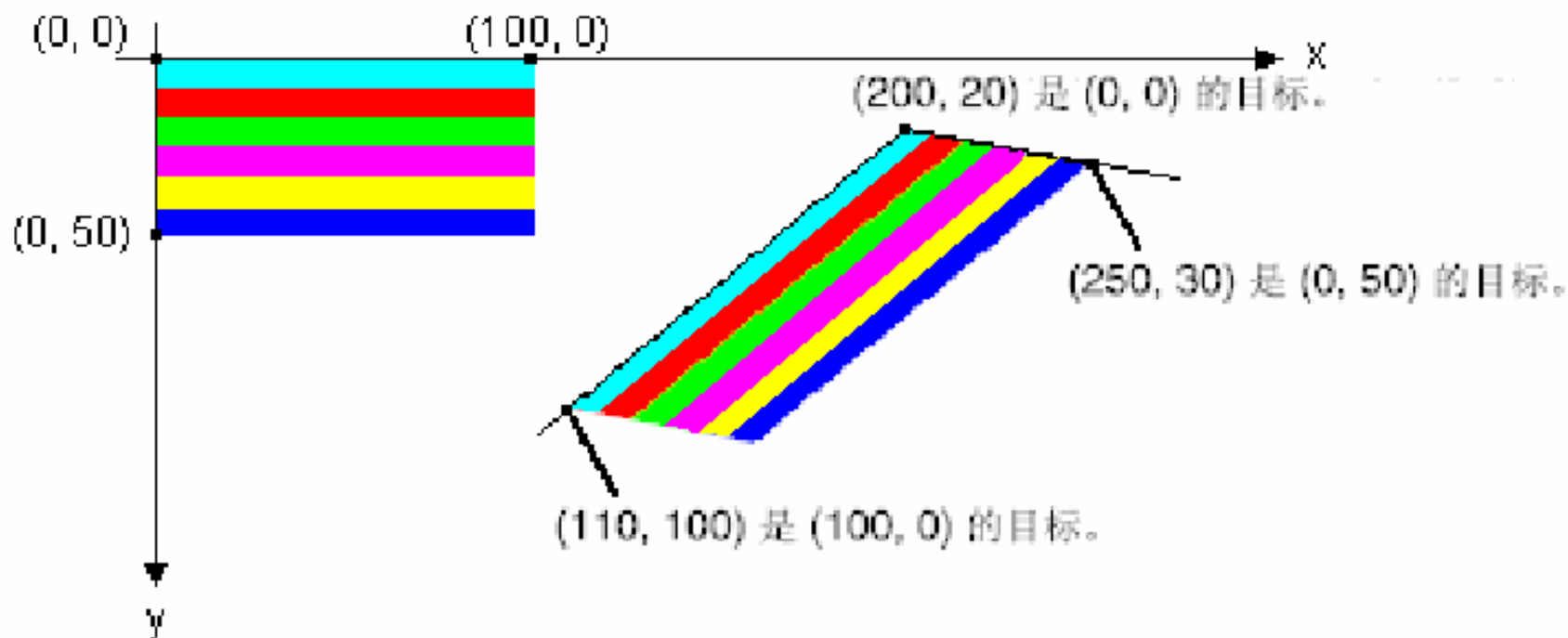
全局变形和局部变形

- `Matrix myPathMatrix = new Matrix();`
- `myPathMatrix.Scale(2, 1);`
- `myPathMatrix.Rotate(30, MatrixOrder.Append);`
- `myGraphicsPath.Transform(myPathMatrix);`
- `// Fill the transformed path on the new coordinate system.`
- `myGraphics.FillPath`
`(mySolidBrush2,`
`myGraphicsPath);`



旋转、反射和扭曲图像

- 通过指定原始图像的左上角、右上角和左下角的目标点可旋转、反射和扭曲图像。这三个目标点确定将原始矩形图像映射为平行四边形的仿射变换。



旋转、反射和扭曲图像

- `Point[] destinationPoints =`
- `{ new Point(200, 20), //左上角`
- `new Point(110, 100), //右上角`
- `new Point(250, 30)}; //左下角`
- `Image image = new Bitmap("C://ly.bmp");`
- `e.Graphics.DrawImage(image, 0, 0); //原图`
- `e.Graphics.DrawImage(image, destinationPoints); //变形`

3、媒体控制接口(MCI)

■ 概述

- 媒体控制接口 (MCI, Media Control Interface) 在控制音频、视频等多媒体外围设备方面，提供了与设备无关的应用程序。
- 由于MCI的设备无关性，系统升级十分方便，从而大大提高了应用系统的开发效率。
- MCI设备驱动器可以直接或通过Windows提供的低级多媒体函数控制媒体硬件。

■ MCI编程接口

- 命令-消息接口

- 应用信息传递方式与MCI设备进行通信。

- 如

- WORD wDeviceID;

- MCI_PLAY_PARMS mciPlayParms;

- MciSendCommand(wDeviceID,MCI_PLAY,0,(D
WORD)(LPVOID)&mciPlayParms);

- 是为需要C语言接口以控制多媒体设备的应用程序而设计的。

■ MCI命令集

- 命令—字符串接口中的命令描述了MCI命令集，每一条命令在命令—消息接口中都有对应的命令消息。例如：close命令字符串等同于MCI_CLOSE命令消息。
- MCI命令可分成四类：
 - 系统命令
 - 需求命令
 - 基本命令
 - 扩展命令

■ MCI命令集

- MCI命令由一字符串组成，语法如下：

Command device_name arguments

其中：command : MCI所使用的命令；

device_name: 指设备类型、文件名或别名；

arguments : 指令所带的参数列表。

■ 关于MCI设备

- MCI设备驱动程序可以按简单和复合设备进行分类。
- 简单设备重放时不需要数据文件。
 - 视盘机和激光唱盘机(CD)都是简单设备。
- 复合设备重放时需要数据文件。
 - MIDI序列和波形音频播放机都是复合设备。
- 与复合设备相关的数据文件叫设备元素
 - MIDI文件和WAVE文件都是设备元素的例子

■ 设备名

- 对于某一给定的设备类型，可能有几种MCI驱动程序共享命令集，但使用不同的数据格式。
- 如动画设备就有几种驱动程序使用同一命令集，但使用不同的文件格式。要单独标识MCI驱动程序，MCI使用设备名。
- 设备名是在注册中的[mci]部分标出的，这一部分标明了所有Windows MCI设备驱动程序。

媒体控制接口(MCI)

■ 设备名

- 下面是典型的[mci]部分的一部分。

[mci]

waveaudio=mciwave.drv

sequencer=mciseq.drv

MMMovie=mcimmp.drv

Cdaudio=mcicda.drv

- 键名(在等号左边)是设备名, 与键名相对的值(在等号的右边)标识MCI驱动程序的文件名, 设备名通常与该驱动程序中的设备类型名是相同的。

■ 设备名

- 如果用一个已经在 [mci]部分中存在的设备名安装MCI设备驱动程序，Windows就给新驱动程序的设备名加上一个整数，以产生一个独特的设备名。
- 在前面的例子中，用cdaudio设备名安装的驱动程序其设备名为cdaudio1，而以后的cdaudio设备名应为cdaudio2

■ 打开MCI设备

- 在使用MCI设备以前，用户必须打开该设备将其初始化；
- 打开设备就将它的驱动程序装入了内存(如果该程序还没有装入)，并且建立一个设备标识符用来指定在后续命令中的设备(命令字符串接口不使用该设备的标识符)。

媒体控制接口(MCI)

■ 打开MCI设备

- 有几种方法可以指定用户要打开的设备：
 - 对于简单设备，用户只需指定设备名将其打开。
 - 对于复合设备，用户只需指定设备名、设备元素或二者同时指定来打开这个设备。
 - 如，下列命令字符串用指定设备名的方法打开一个CD音频设备

`Open cdaudio`

- 下面的命令字符串用指定设备名和设备元素的方法打开一个波形音频复合设备。

`Open bell.wav type waveaudio`

- 也可以象下面给出的例子那样只指定设备的元素来打开一个复合设备。

`Open bells.wav`

■ 打开MCI设备

- 打开一个复合设备时，如果只指定设备元素，则MCI用设备元素文件的扩展名确定要打开哪一个设备。
- 注册中包含一个与文件扩展名和相应的MCI设备类型相关的[mci extensions]部分。
- 下面是[mci extensions]部分的一部分。

[mci extensions]

wav=wavaudio

mid=sequencer

rmi=sequencer

媒体控制接口(MCI)

■ MCI设备类型

- MCI命令通过设备标识符(即设备名称)来存取和控制设备:

`Open cdaudio`

//打开CD音响播放设备

`play cdaudio`

//如果CD唱盘已经在演播器中, 而且设备正常, 唱盘将从第一道开始播放

`Pause cdaudio`

//暂停或停止唱盘的播放

`close cdaudio`

//当程序完成了对某一多媒体设备的访问时, 应关闭设备

■ MCI设备类型

- 所有MCI命令都通过多媒体API函数mciSendString传送给媒体控制接口。该函数取一个MCI命令串，传递给MCI驱动程序，然后将出错消息或返回代码返回到用户提供的缓冲区。函数原型如下：

```
DWORD mciSendString ( LPSTR ipstrCommand, LPSTR  
ipstrReturnString, WORD wReturnLenth, HANDLE hCallback);
```

- 第一个参数是MCI命令串，第二个参数是用来存放返回代码的缓冲区地址，第三个参数是缓冲区的长度，第四个参数是回调函数地址。

■ MCI设备类型

- 下面是一个调用mciSendString函数的例子：

```
#include<mmsystem.h>
```

```
char buff[55];
```

```
mciSendString("open cdaudio", buff , strlen(buff) , NULL);
```

//该命令打开了cdaudio CD-ROM播放器，若出错则错误代码返回给buff字符数组。

媒体控制接口(MCI)

■ 基于消息的MCI

- 媒体控制接口是访问多媒体设备的一种出色的、独立于设备的方式。然而基于字符串命令的本质决定了MCI并不总是像它应该做到的那样快捷。

open test.wav type waveaudio

play test.wav from 1 to 5

close test.wav

//打开test.wav文件, 从位置1至位置5播放, 然后关闭文件。

- 对于所列出的每一行, 媒体控制接口必须把消息拆成分量, 并把指令翻译成命令。这个过程要花费一定时间。尽管时间量很小, 但在耗时的操作中, 其影响可能是很大的。

■ 基于消息的MCI

- 基于消息的MCI，它使用与mciSendCommand API 函数一起的消息(实际上，是在MMSYSTEM.H中定义的常量)。通过使用消息，应用程序在访问多媒体设备时能快一点，这是因为系统不再需要解释命令字符串。
- 用基于字符串的MCI实现的每一操作可用基于消息的MCI完成。

■ 基于消息的MCI的使用

- 一条基于消息的MCI命令包括三部分；
 - 第一部分是指定要执行的MCI命令的一个常量。这些常量以标识符MCI_开头并由一个说明性的名称, 诸如MCI_OPEN或MCI_STOP
 - 第二部分是一个(或一组)用来指定MCI消息子选项的标志。这些标志确定了可以得到什么类型的信息以及如何执行MCI函数。
 - 第三部分是一个结构, 它确定了命令的附加参数。依靠消息, 该结构含有可传递给MCI驱动程序的信息, 或含有从MCI驱动程序返回的值。
 - 消息通过mciSendCommand函数传递。
 - MciSendCommand函数在MMSYSTEM.H中定义

媒体控制接口(MCI)

■ 打开MCI设备

- 在使用MCI设备之前，必须通过MCI_OPEN消息对其初始化。
- 有两种打开设备的方法，这取决于要打开的设备类型：
- 若要打开一简单设备(不需要文件名的设备，如CD音频驱动器或视盘播放器)，必须填写MCI_OPEN_PARMS结构的lpstrDeviceType元素，以指定要打开的设备：

```
DWORD result;
```

```
MCI_OPEN_PARMS      Openparms;
```

```
OpenParms.LpstrDeviceType = "cdaudio";
```

```
Result = mciSendCommand (NULL, MCI_OPEN, MCI_OPEN_TYPE,  
(DWORD)(LPVOED)&openParms);
```

```
WdeviceID = OpenParms.WDeviceID;
```

媒体控制接口(MCI)

■ 打开MCI设备

- 若要打开一复合设备(需要文件名的设备), 必须为媒体在MCI_OPEN_PARMS数据结构的lpstrElementName元素中指定文件名:
 DWORD result;
 MCI_OPEN_PARMS OpenParms;
 OpenParms.lpstrDeviceType = "Waveaudio";
 OpenParms.lpstrElementName = "test.wav"
 Result=mciSendCommand(NULL,MCI_OPEN,MCI_OPEN_TYPE|MCI_OPEN_ELEMENT, (DWORD)(LPVOID) &OpenParms);
 wDeviceID = OpenParms.wDeviceID;
- 除了MCI_OPEN_TYPE标志之外, 还必须给定MCI_OPEN_ELEMENT标志(两者用C的|运算符"或"在一起)。同样, 设备标识符将被返回到MCI_OPEN_PARMS数据结构的wDeviceID元素中。

■ 访问MCI设备

- 在设备打开之后，使用mciSendCommand函数发送附加的消息，必须使用从MCI_OPEN返回的设备标识符作为函数的第一个参数。
- 例如在CD音频设备打开之后，可以用下列代码开始播放：

```
DWORD result;
```

```
MCI_PLAY_PARMS PlayParms;
```

```
Rdsult=mciSendCommand(DeviceID,MCI_PLAY,0,  
(DWORD)(LPVOID)&PlayParms);
```

媒体控制接口(MCI)

■ 访问MCI设备

- 若是启动波形音频复合设备，可使用类似的命令。此外，在程序中应该测试mciSendCommand函数的返回值：

```
DWORD result;
```

```
Char buffer[255];
```

```
Result = mciSendCommand(...);
```

```
If (result){
```

```
    mciGetErrorString (result, buffer, sizeof(buffer));
```

```
    MessageBox( hWnd, buffer, "MCI Device", MB_OK);
```

```
    Return;}
```

- 当程序完成了对设备的访问之后，应用程序应该关闭MCI设备。要关闭一MCI设备，可对打开的设备发送MCI_CLOSE消息，该命令释放设备，从而使其它应用程序可以访问这个设备。

MMCOM组件

- 组件就是对象。JAVA中叫BEAN, C++ Builder中叫组件, Delphi中叫部件, 而在VB中叫控件。组件是对数据和方法的简单封装
- COM组件(Component Object Model, 组件对象模型)是遵循COM规范编写的一些小的二进制可执行文件, 它们可以给应用程序, 操作系统以及其他组件提供服务。开发自定义的COM组件就如同开发动态的, 面向对象的API。多个COM对象可以连接起来形成应用程序或组件系统。
 - MMC(Microsoft Multimedia Control)
 - Media Player
 - Real Player
 - NetMeeting



4、多媒体软件开发工具包(MMSDK)

- 软件开发工具包 (Software Development Kit, SDK) 是辅助开发某一类软件的相关文档、范例和工具的集合, 通过使用多媒体软件开发工具包中提供的各种接口和工具, 开发人员可以无须了解复杂的技术内幕 (如视频压缩技术原理、文字识别技术、语音识别技术等), 而方便有效地开发多媒体应用程序。
- SDK通过API (Application Programming Interface)、动态链接库(Dynamic Link Library, DLL)、导入库(LIB)等等技术实现编程
 - DOS API : 中断调用的形式 (INT 21h)
 - Windows 的三大模块就是以 DLL 的形式提供的 (Kernel32.dll, User32.dll, GDI32.dll), 里面就含有 API 函数的执行代码, API函数调用通过对应的.H和.LIB文件进行。

■ DirectX SDK 简介

- DirectX是由微软公司建立的游戏编程接口。由C++编程语言实现，遵守COM约定。其影响力已渐渐超越OpenGL并被大多数PC游戏开发商所采用。
- 当Windows系统成为个人计算机的标准操作系统后，多任务与共通的特性让软件工程师不必再为兼容性的问题伤脑筋，但这些特性的副作用就是“慢”，使得直接存取图像、声音与硬件能力受到很大的限制，甚至让高度的声光效果游戏停留在MS-DOS阶段。直到DirectX的前身“Game SDK”的出现，程序才可以直接存取硬件，游戏才得以在Windows平台上快速发展。

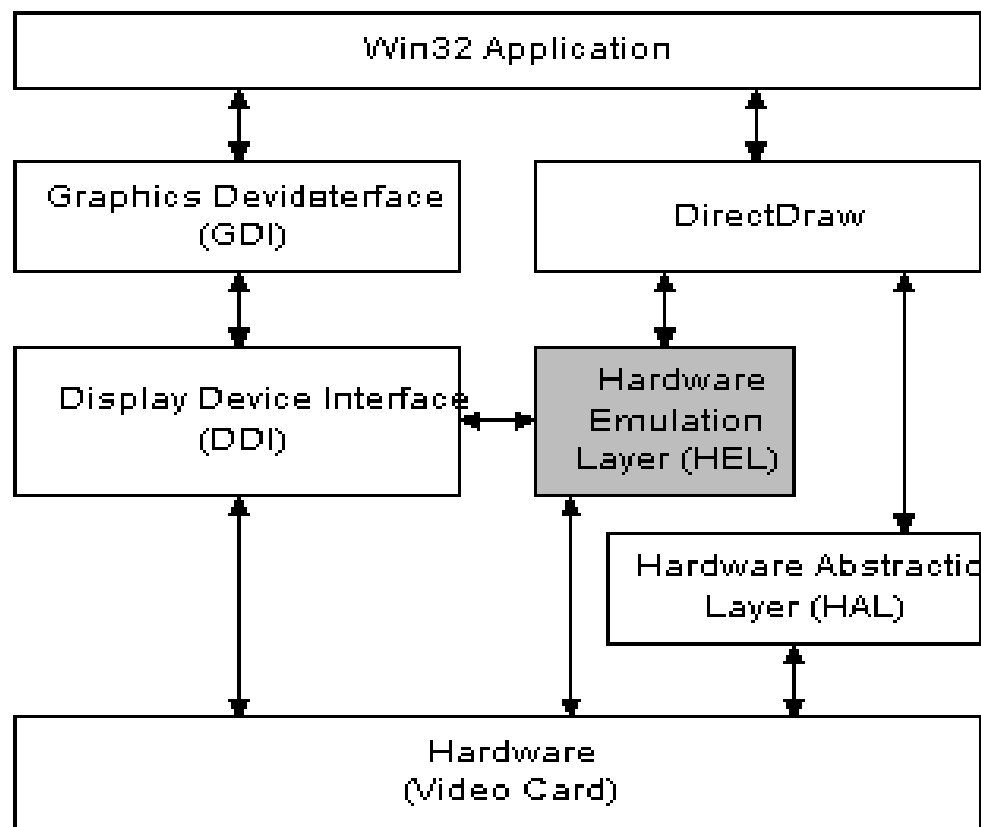
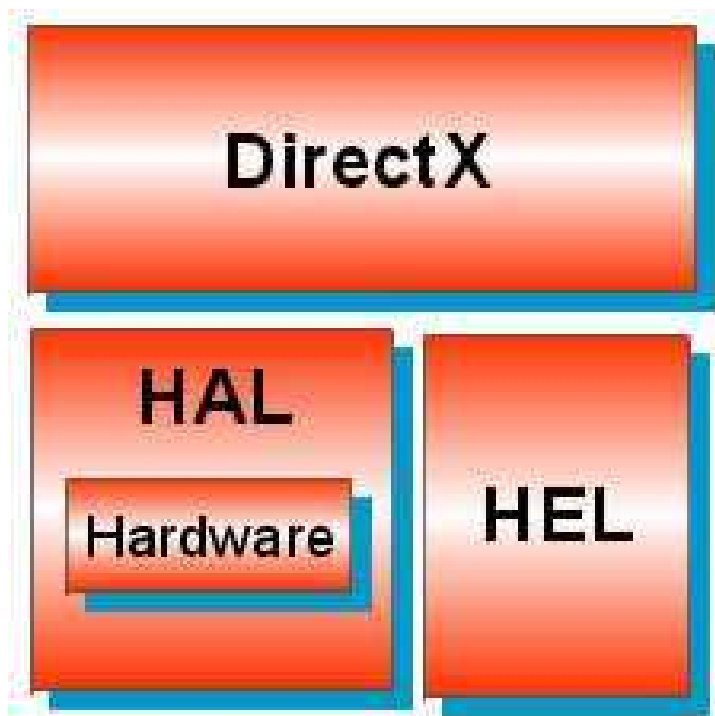
DirectX的特色

- 硬件无关性
- 直接存取显存
- 支持硬件加速
- 网络联机功能
- “DirectX”可视为一种程序设计者与硬件间的接口。程序设计者不需要花费心思去构想如何来编写底层的程序代码与硬件打交道，只须巧妙地运用DirectX中的各类组件，便可简单地制作出高效能的游戏程序。

■ DirectX硬件加速的工作原理：

- 如果硬件设备提供相应的硬件加速功能， 那就用硬件加速；
- 如果没有相应的硬件加速功能， 那就软件模拟； 之后就通知硬件设备的驱动层。
 - HAL(Hardware Abstraction Layer)： 硬件抽象层直接与硬件对话。当硬件设备支持您请求的功能时， 将使用HAL得到硬件加速。
 - HEL(Hardware Emulation Layer)： 当硬件设备不支持请求的功能时， 将使用硬件仿真层HAL， 使用HEL速度较慢， 但能保证程序得以正常运行。

DirectX结构



- HAL (Hardware Abstraction Layer)
- HEL (Hardware Emulation Layer)

DirectX组成

■ DirectX Graphics

- DirectDraw®:graphics programming API
- Direct3D®:three-dimensional (3D) graphics API
- Direct3D extensions (D3DX):math, texturing, and shaded manipulation

■ DirectX Audio

- DirectSound®:high-performance audio applications that play and capture waveform audio
- DirectMusic®:musical and non-musical soundtracks based on waveforms, MIDI sounds
- Cross-platform Audio Creation Tool (XACT):跨平台音效制作工具，是微软所提供的音效API，为DirectX 10的部份功能。XACT音乐音效编辑器可以产生XAP档案，能在XNA中编辑与播放音效

- DirectInput®:process data from a keyboard, mouse, joystick, or other game controller API
- DirectPlay®:multiplayer networked games
- DirectSetup:one-call installation of the DirectX components
- DirectShow®:high-quality capture and playback of multimedia streams

DirectShow

- DirectShow是微软公司在ActiveMovie和Video for Windows (VFW) 的基础上推出的新一代基于COM的流媒体处理的开发包，与DirectX开发包一起发布。
- DirectShow技术是建立在DirectDraw和DirectSound组件基础之上的。
 - DirectDraw对显卡进行控制以显示视频
 - DirectSound对声卡进行控制以播放声音

DirectShow

- DirectShow可提供高质量的多媒体流的捕获和回放功能；支持多种媒体格式，包括
 - ASF (Advanced Systems Format)
 - MPEG (Motion Picture Experts Group)
 - AVI (Audio-Video Interleaved)
 - MP3 (MPEG Audio Layer-3)
 - WAV声音文件
- 可以从硬件上捕获媒体数据流。
- 可以自动检测并使用视频和音频加速硬件。

DirectShow Filters

- basic units of DirectShow programs .
- establish connections with other filters.
- negotiate the types of connections.
- receive some basic messages(run, stop, and pause).



Filter type

1. Source Filter: produce streams.
2. Transform Filter:
 - parse a stream of data,
 - encode /decode,
 - add a text overlay to a video sequence,
 - create a "tee" in the stream.
3. Renderer Filter: translate a DirectShow stream into some form of output.
 - write a stream to a file on the disc.
 - send Audio streams to the speakers or Video streams to a window on the desktop.

Filter Graph

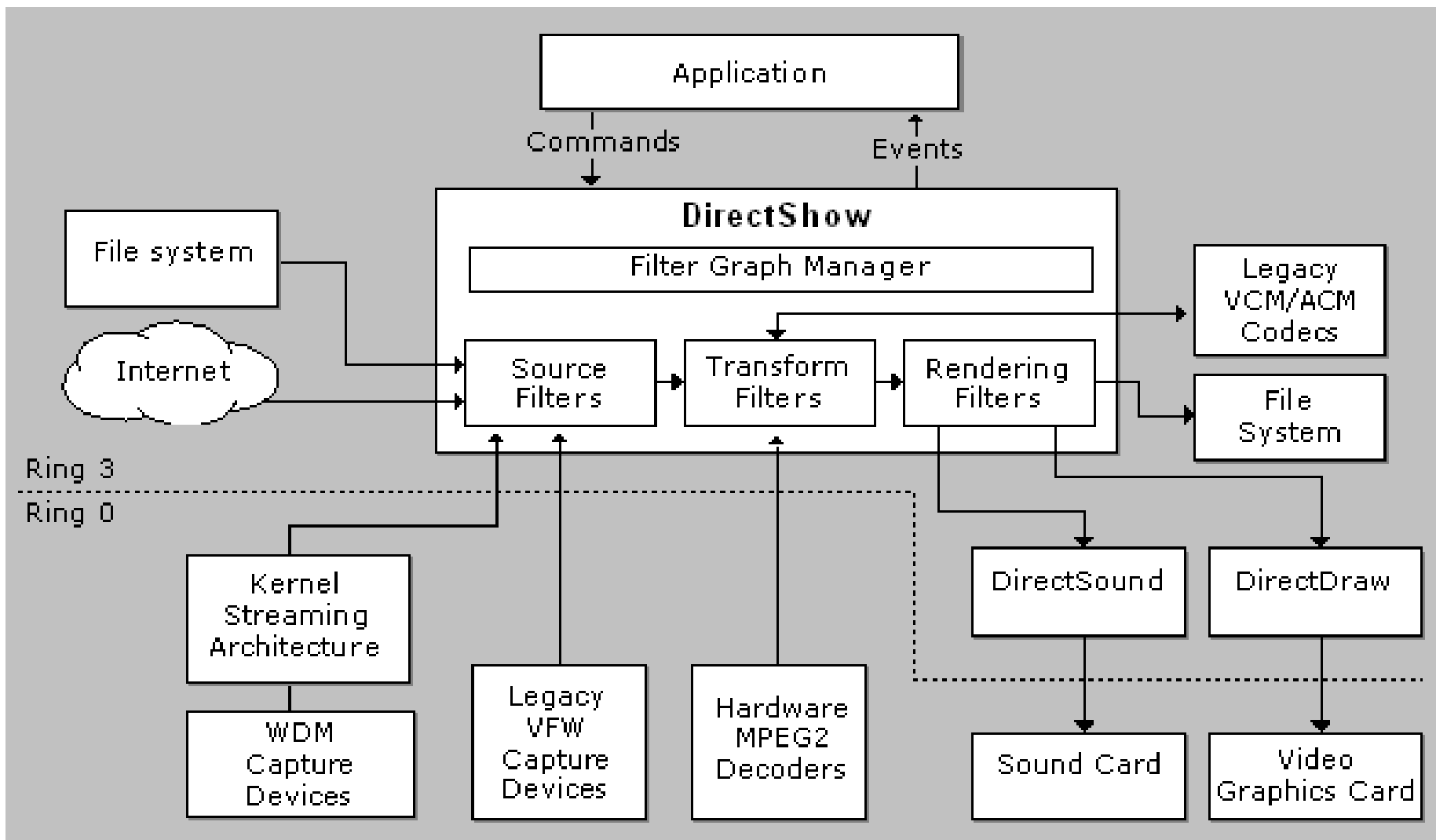
- The DirectShow filter graph organizes a group of filters into a functional unit.

Example;



- A Filter Graph can be created into 2 ways;
 - 1) Manual Connect
 - 2) Intelligent Connect

DirectShow Overview



Using DirectShow

- Through the Windows Media Player Control

Windows Media Player Control provides control of video and audio in Web pages and other applications.

- With Prebuilt Filters

DirectShow provides a standard set of prebuilt binary filters as part of the DirectShow SDK.

- Writing Custom Filters

SDK enables you to create your own filters using the DirectShow class library. DirectShow filters can be written in any language that can generate objects adhering to COM. The base classes for DirectShow are written in C++.

- Create and modify filter graphs using a visual, drag-and-drop interface.
- Simulate programmatic calls to build a graph, such as `IGraphBuilder::RenderFile`.
- Run, pause, stop, and seek a graph.
- See what filters are registered on your computer, and view registry information for each filter.
- View filter property pages.
- View the media types of pin connections.

5、XNA

- XNA是基于DirectX的3D游戏开发环境,美国微软于 8 月 13 日发表针对业余创作者所设计的游戏开发套件“XNA Game Studio Express”,提供没有专门开发器材的一般 PC Windows XP 使用者开发跨 Xbox 360 与 PC 平台游戏的管道. XNA是Microsoft的下一代软件开发平台,致力于帮助开发者更快地开发更好的游戏.
 - X--DirectX和Xbox;
 - N--Next Generation;
 - A--Architecture

■ Microsoft. Xna

- . Framework
- . Graphics
- . Content
- . Input
- . Audio

■ Microsoft.Xna.Framework Classes/Structs

- Game
- ContentManager
- GraphicsDeviceManager
- GameComponent
- DrawableGameComponent
- Vector2
- Vector3
- Point
- Matrix
- BoundingBox
- BoundingSphere
- Texture2D
- SpriteFont

■ Game Class

- Initialize()
- LoadContent()
- UnloadContent()
- Update()
- Draw()
- Components
- Separate out a generic/reusable class library

Render a 2D Texture

- 1. Include the texture in the Content project. Supported formats: bmp, .dds, .dib, .hdr, .jpg, .pfm, .png, .ppm, and .tga
- 2. Initialize the SpriteManager class within the LoadContent() or Initialize() method:

```
SpriteManager.Initialize(this);
```

- 3. Create data member to store texture:

```
private Texture2D mTexture;
```

- 4. Load the texture in the LoadContent() method:

```
mTexture =  
Content.Load<Texture2D>(@"Content\Textures\Skybox\back");
```

- 5. Render the texture in the Draw() method:

```
SpriteManager.DrawTexture2D(mTexture, Vector2.Zero, Color.White);
```

Draw Text

- 1. Initialize the SpriteManager class as required before any calls can be made to the SpriteManager.
- 2. Add a call to SpriteManager.DrawString in the Draw() method:

```
SpriteManager.DrawString("Hello World", 40.0f, 40.0f,  
                          Color.White);
```

- Variety of overloads to the DrawString method:
 - Change the font
 - The blend color
 - Rotation
 - Sorting

View the Game's Framerate

- 1. Add the following statements to the Game-derived constructor, LoadContent(), or Initialize() method:

```
FpsComponent fps = new FpsComponent(this);  
fps.Location = FpsComponent.ScreenLocation.TitleBar;  
Components.Add(fps);
```

- FpsComponent display locations:
 - Titlebar
 - UpperLeft
 - UpperRight
 - LowerLeft
 - LowerRight

Collect Keyboard Input

- 1. Add a using statement for the Bespoke.Games.Framework.Input namespace;
- 2. (Optional) Create a data member to store the keyboard component:

```
private KeyboardComponent mKeyboardComponent;
```

- 3. Add the following statements to the Game-derived constructor, LoadContent(), or Initialize() method:

```
mKeyboardComponent = new KeyboardComponent(this);
```

```
Components.Add(mKeyboardComponent);
```

- 4. Add keyboard queries to the Update() method:

```
if (mKeyboardComponent.WasKeyPressedThisFrame(Keys.Escape))  
{Exit();}
```

Initialize a 3D Camera

- 1. Add the following statements to the Game-derived constructor, LoadContent(), or Initialize() method:

```
mCamera = new CameraComponent(this);  
Services.AddService(typeof(ICamera), mCamera);  
Components.Add(mCamera);  
mCamera.KeyboardComponent = mKeyboardComponent;  
mCamera.GamePadComponent = mGamePadComponent;  
mCamera.Position = new Vector3(0.0f, 20.0f, 200.0f);  
mCamera.Orientation = Vector3.Up;  
mCamera.Direction = Vector3.Forward;  
mCamera.LookAtOffset = Vector3.Forward;  
mCamera.NearPlaneDistance = 1.0f;  
mCamera.FarPlaneDistance = 100000.0f;  
mCamera.FieldOfView = MathHelper.PiOver4;  
mCamera.AspectRatio = GraphicsDevice.PresentationParameters.BackBufferWidth /  
GraphicsDevice.PresentationParameters.BackBufferHeight;  
mCamera.UpdateProjectionMatrix();
```

Draw a Reference Grid

- 1. Initialize a camera
- 2. Add the following statements to the Game-derived constructor, LoadContent(), or Initialize() method:

```
GridComponent grid = new GridComponent(this);
```

```
Components.Add(grid);
```

- You can modify the size (number of cells), scale (spacing between each line) and the color of the grid

Render a 3D Model

- 1. Include the model in your Content project (this is typically a subproject within your Game project)
 - Supported Formats:
 - .fbx (Autodesk)
 - .x (DirectX Surface)

Be certain that associated textures reside in the proper locations.

- 2. Add the following statements to the Game-derived LoadContent(), or Initialize() method:

```
Model tankModel = Content.Load<Model>(@"Content\Models\tank");  
  
Actor tankActor = new DynamicActor(this, "Tank", Vector3.Zero,  
    Vector3.Up, 0.05f, 1.0f, tankModel, mCamera);  
  
tankActor.Initialize();
```

- 3. Add tankActor.Update() and tankActor.Draw() calls to the corresponding Game-derived Update() and Draw() methods.

Play Sound

- 1. Build your sound project using XACT.
- 2. Include your sound project (.xap) into your Content project.
- 3. Initialize the SoundManager static class in the `LoadContent()` or `Initialize()` method:

```
SoundManager.Initialize(@"Content\Audio\SoundProject.xgs",  
    @"Content\Audio\Wave  
Bank.xwb", @"Content\Audio\Sound Bank.xsb");
```

- 4. Play sounds with `SoundManager.Play()` :
- 5. Call `SoundManager.Update()` within the main `Update()` loop.

Render a Skybox

- 1. Create your skybox textures (Terragen) and import them into your Content project (front, back, left, right, top)
- 2. Create a SkyBoxComponent data member.

```
private SkyBoxComponent mSkyBox;
```

- 3. Add the following code to your Initialize() or LoadContent() method:

```
Texture2D front =  
    Content.Load<Texture2D>(@"Content\Textures\SkyBox\front");  
  
Texture2D back =  
    Content.Load<Texture2D>(@"Content\Textures\SkyBox\back");  
  
Texture2D left =  
    Content.Load<Texture2D>(@"Content\Textures\SkyBox\left");  
  
Texture2D right =  
    Content.Load<Texture2D>(@"Content\Textures\SkyBox\right");  
  
Texture2D top = Content.Load<Texture2D>(@"Content\Textures\SkyBox\top");  
  
mSkyBox = new SkyBoxComponent(this, "SkyBox", front, back, left, right,  
    top, 1000.0f, mCamera);  
  
mSkyBox.Initialize();
```

- 4. Call mSkyBox.Draw() from the main draw loop. Call this as the first object to be rendered.

Render Terrain

- 1. Reference Bespoke.Games.Framework.Content.dll from your Content project.
- 2. Import the heightmap (.raw) into your Content project and choose the Bespoke Software – Terrain Content Importer/Processor
- 3. Import the associated texture into your Content project.
- 4. Create a TerrainComponent data member.

```
private TerrainComponent mTerrain;
```

- 5. Add the following code to your Initialize() or LoadContent() method:

```
TerrainData terrainData =  
    Content.Load<TerrainData>(@"Content\Other\TerrainHeightMap");  
  
Texture2D terrainTexture =  
    Content.Load<Texture2D>(@"Content\Textures\Terrain");  
  
mTerrain = new TerrainComponent(this, terrainData, terrainTexture, 513,  
    513, 4.0f, 6000.0f,  
    Color.White, -1000.0f, mCamera);  
  
mTerrain.Initialize();
```

- 5. Call mTerrain.Draw() from the main draw loop.

CameraComponent

■ Keyboard

- WASD (forward, turn left, backward, turn right)
- Up Arrow (turn up), Down Arrow (turn down)

■ GamePad

- Left Thumbstick (turn up, down, left, right)
- Right Trigger (forward)
- Left Trigger (reverse)

StereoScopicChaseCameraComponent

■ Keyboard

- PageUp/PageDown (increase/decrease IPD)
- End (toggle stereoscopic rendering)

WPF & Silverlight

- WPF (Windows Presentation Foundation) 是MS .NET Framework 3.0的组成部分之一，它是一套基于XAML (eXtensible Application Markup Language)、.NET Framework、矢量绘图 (vector graphic) 技术的展示层 (presentation layer) 开发框架，为MS下一代用户界面技术，将广泛被用于下一代的Windows平台的界面开发。
- Silverlight是WPF和XAML子集，即WPF/E，支持流媒体和矢量式的绘图能力，是MS与Adobe Flash竞争的web技术。
- WPF特征：
 - 绘图服务：支持矢量绘图和3D透视图，所有的图形皆可由Direct3D产生。
 - 部署：WPF部署方式可分成两类：standalone与XAML Browser Applications (XBAP)]])
 - 互操作性：WPF可与Win32互相操作
 - 媒体播放：WPF对2D图形提供形状基元 (shape primitives)，内置笔刷 (brushes)、画笔 (pens)、几何 (geometries)，与变形 (transforms)。WPF以Direct3D提供完整的3D功能，支持了大量的图像、视频、动画格式
 - 用户界面：WPF提供了内置组件控件集合，组件的外貌样品可被完全改写。WPF将逻辑层与外观展示层进行分离，实现Designer 和 Developer分工合作。
- Game Develop.DirectX, OpenGL, WPF or XNA?
 - DirectX , C++ & Win32 MMAPI
 - Managed DirectX-->XNA , C# & .NET
 - WPF, C# & .NET

6、Microsoft Speech SDK

■ 语音技术

- 语音识别 (speech recognition, SR)
- 语音合成 (speech synthesis, SS, 即TTS)

■ 微软公司推出了Microsoft Speech SDK

- 提供关于语音处理的一套应用程序编程接口SAPI (Speech Application Programming Interface)。
- 提供了实现TTS和SR程序的基本函数，大大简化了语音编程的难度，降低了语音编程的工作量。

■ Microsoft Speech SDK

- 采用COM标准开发，底层协议都以COM组件的形式完全独立于应用程序层。
- 语音识别由识别引擎（Recognition Engine）管理。
- 语音合成由语音合成引擎（Synthesis Engine）负责。
- 程序员只需专注于自己的应用，调用相关的语音应用程序接口（SAPI）来实现语音功能。

■ 语音识别的主要接口包括：

- IspRecognizer接口：用于创建语音识别引擎的实例，在创建时通过参数选择引擎的种类。
- IspRecoContext接口：主要用于接受和发送与语音识别消息相关的事件消息，装载和卸载识别语法资源。
- IspRecoGrammar接口：通过这个接口，应用程序可以载入、激活语法规则
- IspPhrase 接口：用于获取识别的结果，包括识别的文字、识别了哪一条语法规则等。

- 语音合成主要通过ISpVoice接口来控制的
 - 通过调用其中的Speak方法可以朗读出文本内容。
 - 通过调用SetVoice/GetVoice方法（在.NET中已经转变成Voice属性）来获取或设置朗读的语音。
 - 通过调用GetVolume/SetVolume、GetRate/SetRate等方法（在.NET中已经转变成Volume和Rate属性）来获取或设置朗读的音量和语速。

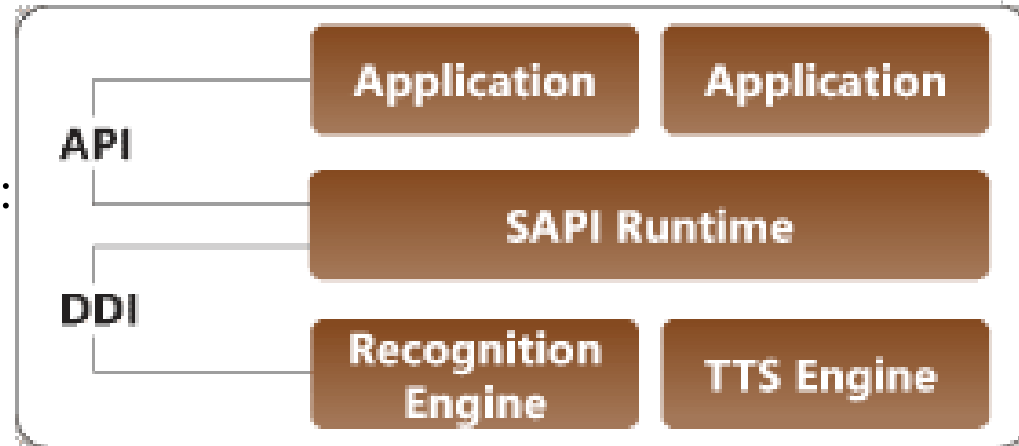
- TTS是微软出品的一套文字朗读引擎（Text-To-Speech Engine），这些英语软件就是调用它来进行英文朗读的，TTS引擎所需的Microsoft Text-to-Speech Engine与Microsoft Speech API软件都可到微软的站点去下载。
- 安装了TTS引擎后，在Windows所在目录下会生成一个SPEECH目录，其中有一个Vtxtauto.tlb文件，在编程时我们需要调用它。

- 执行选单命令“文件/新建工程/标准EXE”并确定，然后执行选单“工程”中的“引用”，单击“浏览”按钮到Windows目录下的SPEECH子目录，打开Vtxtauto.tlb文件，将“VoiceText 1.0 Type Library”添加到引用列表中，选中它并单击确定。
- 将库文件Vtxtauto引入VB后，我们可以通过选单“视图”中的“对象浏览器”来了解它所封装的类，以及各类成员函数的属性的意义、使用格式等信息。
 - 例：使用TTS引擎制作英文朗读器。

- Unified Communications Managed API (UCMA)

- .NET System.Speech

- System.Speech.Recognition: Provides access to speech recognition engines.
- System.Speech.Synthesis: Provides access to voice output engines for text-to-speech (TTS).



System. Speech. Synthesis Namespace

■ Classs

- BookmarkReachedEventArgs
- FilePrompt
- InstalledVoice
- PhonemeReachedEventArgs
- Prompt
- PromptBuilder
- PromptEventArgs
- PromptStyle
- SpeakCompletedEventArgs
- SpeakProgressEventArgs
- SpeakStartedEventArgs
- SpeechSynthesizer
- StateChangedEventArgs
- VisemeReachedEventArgs
- VoiceChangeEventArgs
- VoiceInfo

■ Enumerations

- PromptBreak
- PromptEmphasis
- PromptRate
- PromptVolume
- SayAs
- SynthesisMediaType
- SynthesisTextFormat
- SynthesizerEmphasis
- SynthesizerState
- VoiceAge
- VoiceGender

System.Speech.Recognition Namespace

■ Classes

- AudioLevelUpdatedEventArgs
- AudioSignalProblemOccurredEventArgs
- AudioStateChangedEventArgs
- Choices
- DictationGrammar
- EmulateRecognizeCompletedEventArgs
- Grammar
- GrammarBuilder
- LoadGrammarCompletedEventArgs
- RecognitionEventArgs
- RecognitionResult
- RecognizeCompletedEventArgs
- RecognizedAudio
- RecognizedPhrase
- RecognizedWordUnit
- RecognizerInfo
- RecognizerUpdateReachedEventArgs
- ReplacementText

- SemanticResultKey
- SemanticResultValue
- SemanticValue
- SpeechDetectedEventArgs
- SpeechHypothesizedEventArgs
- SpeechRecognitionEngine
- SpeechRecognitionRejectedEventArgs
- SpeechRecognizedEventArgs
- SpeechRecognizer
- SpeechUI
- StateChangedEventArgs

■ Enumerations

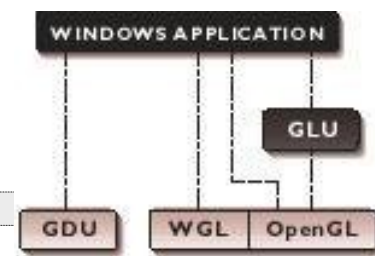
- AudioSignalProblem
- AudioState
- DisplayAttributes
- RecognizeMode
- RecognizerState
- SubsetMatchingMode

7、OpenGL/OpenAL/OpenCV/OpenCL



■ OpenGL (Open Graphics Library)

- OpenGL是个定义了一个跨编程语言、跨平台的编程接口的规格，它用于生成二维、三维图像。这个接口由近三百五十个不同的函数调用组成，用来从简单的图元绘制复杂的三维景象。而另一种编程接口系统是仅用于Microsoft Windows上的Direct3D。OpenGL常用于CAD、虚拟现实、科学可视化程序和电子游戏开发。Microsoft、SGI、IBM、DEC、SUN、HP等公司都采用OpenGL做为三维图形标准，以OpenGL为基础开发出自己的产品，其中比较著名的产品包括动画制作软件Soft Image和3D Studio MAX、仿真软件Open Inventor、VR软件World Tool Kit、CAM软件ProEngineer、GIS软ARC/INFO等
- OpenGL的高效实现（利用了图形加速硬件）存在于Windows，很多UNIX平台和MacOS。这些实现一般由显示设备厂商提供，而且非常依赖于该厂商提供的硬件。开放源代码库Mesa是一个纯基于软件的图形API，它的代码兼容于OpenGL。但是，由于许可证的原因，它只声称是一个“非常相似”的API。
- OpenGL规范由1992年成立的OpenGL架构评审委员会（ARB）维护



OpenGL特点及功能

- OpenGL实际上是一个开放的三维图形软件包，它独立于窗口系统和操作系统，以它为基础开发的应用程序可以十分方便地在各种平台间移植。
- 建模
 - OpenGL图形库除了提供基本的点、线、多边形的绘制函数外，还提供了复杂的三维物体（球、锥、多面体、茶壶等）以及复杂曲线和曲面（例如Bezier、Nurbs等曲线或曲面）绘制函数。
- 变换
 - OpenGL图形库的变换包括基本变换和投影变换。基本变换有平移、旋转、变比、镜像四种变换，投影变换有平行投影（又称正射投影）和透视投影两种变换。其变换方法与机器人运动学中的坐标变换方法完全一致，有利于减少算法的运行时间，提高三维图形的显示速度。

■ 颜色模式设置

- OpenGL颜色模式有两种，即RGBA模式和颜色索引（Color Index）。

■ 光照和材质设置

- OpenGL光有辐射光（Emitted Light）、环境光（Ambient Light）、漫反射光（Diffuse Light）和镜面光（Specular Light）。材质是用光反射率来表示。场景（Scene）中物体最终反映到人眼的颜色是光的红绿蓝分量与材质红绿蓝分量的反射率相乘后形成的颜色。

■ 纹理映射（Texture Mapping）

- 利用OpenGL纹理映射功能可以十分逼真地表达物体表面细节

■ 位图显示和图像增强

- 图像功能除了基本的拷贝和像素读写外，还提供融合（Blending）、反走样（Antialiasing）和雾（fog）的特殊图像效果处理。以上三条可是被仿真物更具真实感，增强图形显示的效果。

■ 双缓存动画（Double Buffering）

- 双缓存即前台缓存和后台缓存，简而言之，后台缓存计算场景、生成画面，前台缓存显示后台缓存已画好的画面。此外，利用OpenGL还能实现深度暗示（Depth Cue）、运动模糊（Motion Blur）等特殊效果。从而实现了消隐算法

■ *Download Address: <http://www.opengl.org/>*

7、OpenGL/**OpenAL**/OpenCV/OpenCL



■ OpenAL (Open Audio Library)

- OpenAL是开源软件的跨平台音效API，由Loki Software，使用在Windows、Linux 系统上，用来音效缓冲和收听者中编码。 <http://www.openal.org/>
- OpenAL设计给多通道三维位置音效的特效表现。其 API 风格模仿自OpenGL。

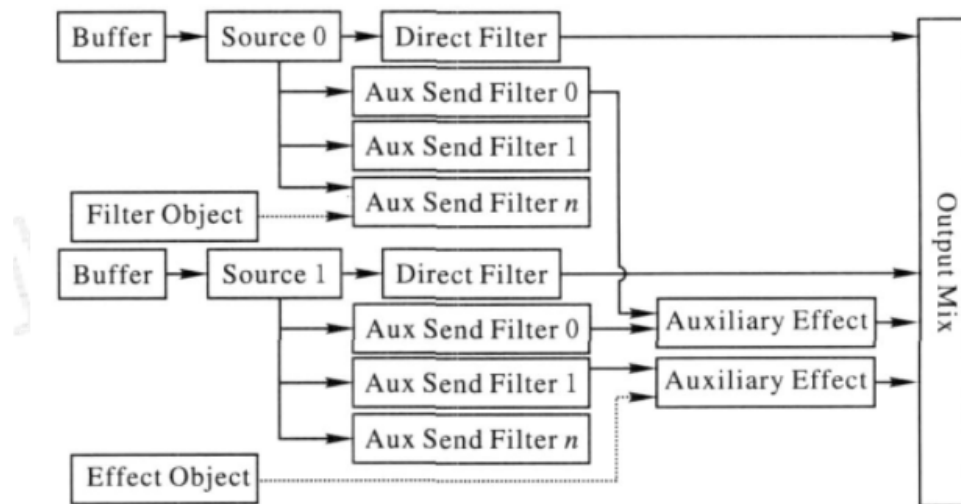
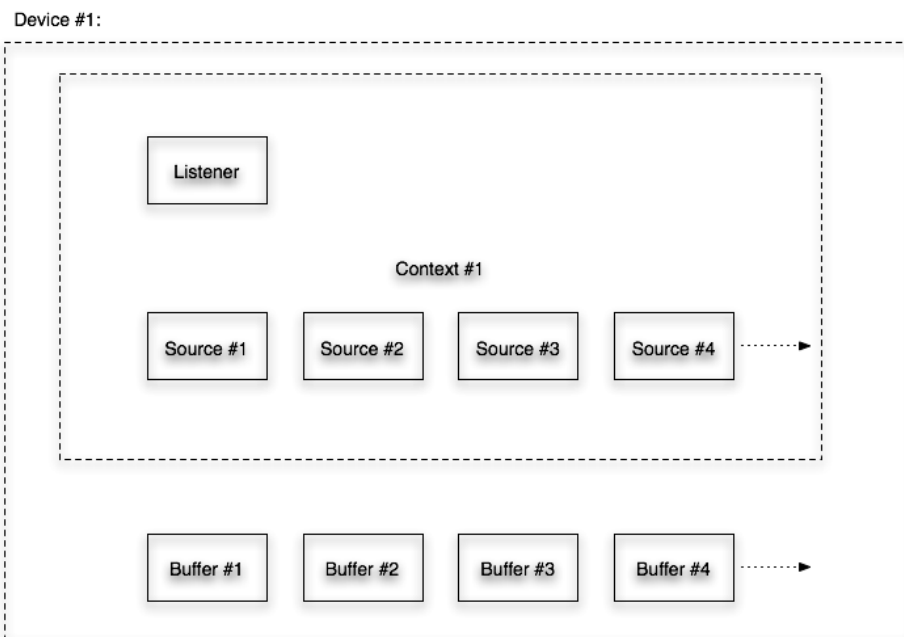


图 1 带环境音效扩展的 OpenAL 结构图

- OpenAL抽象出三种基本对象：buffers、sources、listener。
Buffer用来填充声音数据，然后附加到一个Source上，Source可以被定位并播放。声音播放的效果取决于source相对于listener的位置和方向。通过创建数个sources、buffers和一个唯一的listener，并动态更新sources的位置和方向，就可以产生逼真的3D音效。
- OpenAL基本对象及其与context和device之间的关系：



7、OpenGL/OpenAL/OpenCV/OpenCL



■ OpenCV (Open Source Computer Vision Library)

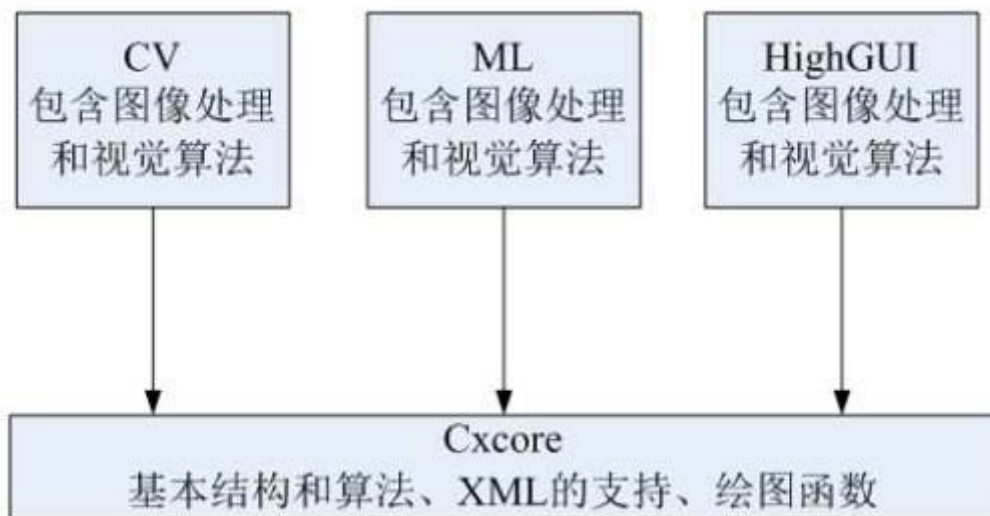
- OpenCV是一个基于BSD许可（开源）发行的跨平台计算机视觉库，可以运行在Linux、Windows、Android和Mac OS操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了Python、Ruby、MATLAB等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。
- OpenCV用C++语言编写，它的主要接口也是C++语言，但是依然保留了大量的C语言接口。该库也有大量的Python, Java and MATLAB/OCTAVE（版本2.5）的接口。这些语言的API接口函数可以通过在线文档获得。如今也提供对于C#, Ruby的支持。
- 应用领域：人机互动、物体识别、图像分割、人脸识别、动作识别、运动跟踪、机器人、运动分析、机器视觉、结构分析、汽车安全驾驶

■ OpenCV功能

- 图像数据操作（内存分配与释放，图像复制、设定和转换）
- 图像/视频的输入输出（支持文件或摄像头的输入，图像/视频文件的输出）
- 矩阵/向量数据操作及线性代数运算（矩阵乘积、矩阵方程求解、特征值、奇异值分解）
- 支持多种动态数据结构（链表、队列、数据集、树、图）
- 基本图像处理（去噪、边缘检测、角点检测、采样与插值、色彩变换、形态学处理、直方图、图像金字塔结构）
- 结构分析（连通域/分支、轮廓处理、距离转换、图像矩、模板匹配、霍夫变换、多项式逼近、曲线拟合、椭圆拟合、狄劳尼三角化）
- 摄像头定标（寻找和跟踪定标模式、参数定标、基本矩阵估计、单应矩阵估计、立体视觉匹配）
- 运动分析（光流、动作分割、目标跟踪）
- 目标识别（特征方法、HMM模型）
- 基本的GUI（显示图像/视频、键盘/鼠标操作、滑动条）
- 图像标注（直线、曲线、多边形、文本标注）

■ OpenCV模块

- cv - 核心函数库
- cvaux - 辅助函数库
- cxcore - 数据结构与线性代数库
- highgui - GUI函数库
- ml - 机器学习函数库



7、OpenGL/OpenAL/OpenCV/**OpenCL**

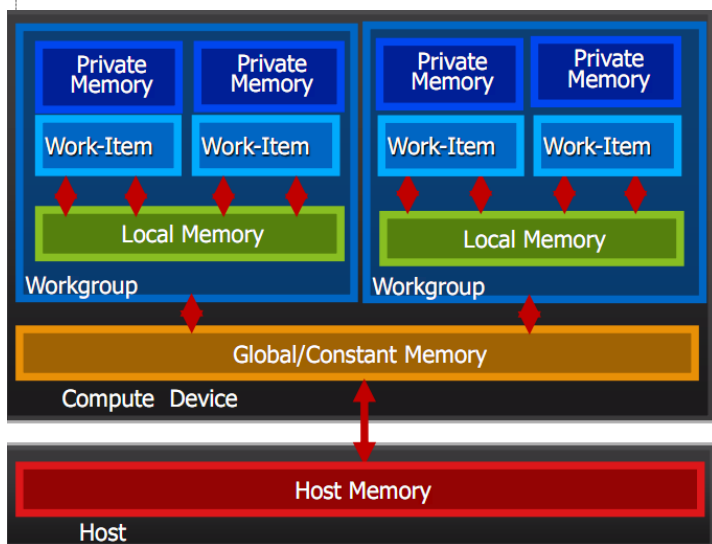


■ OpenCL (Open Computing Language)

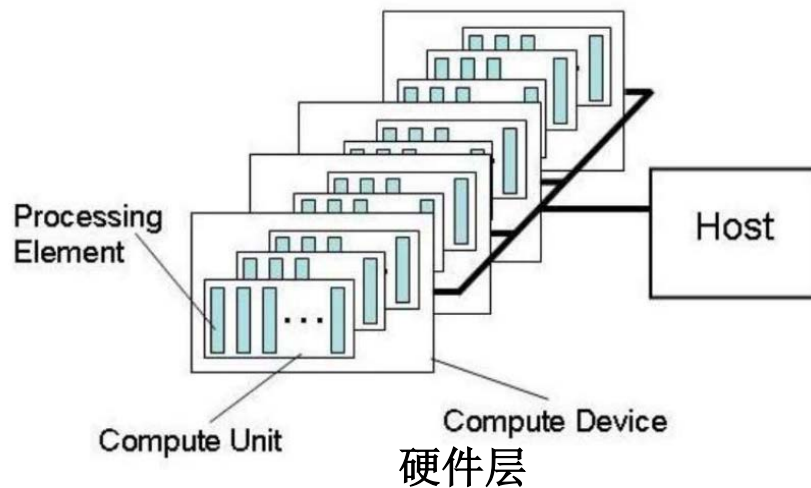
OpenCL

- OpenCL是一个为异构平台编写程序的框架，此异构平台可由CPU，GPU或其他类型的处理器组成。OpenCL由一门用于编写kernels（在OpenCL设备上运行的函数）的语言（基于C99）和一组用于定义并控制平台的API组成。OpenCL提供了基于任务分割和数据分割的并行计算机制。
- OpenCL类似于另外两个开放的工业标准OpenGL和OpenAL，这两个标准分别用于三维图形和计算机音频方面。OpenCL扩展了GPU用于图形生成之外的能力。OpenCL由非盈利性技术组织Khronos Group掌管。

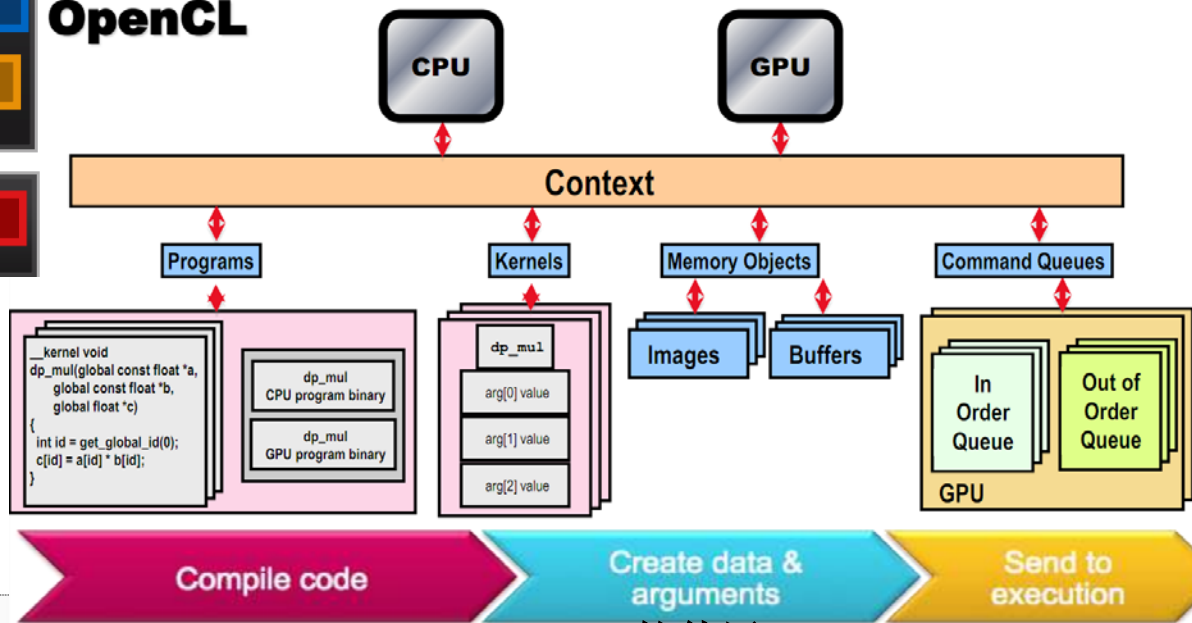
■ OpenCL架构



内存架构



OpenCL



软件层

8、 JOGL

■ JOGL (Java Bindings for OpenGL)

- net.java.games.jogl包, 类有:

- *GLDrawable

- *GLCanvas

- *GLJPanel

- *GLCapabilities

- *GLDrawableFactory

■ WorldWind (NASA)

- JAVA

- C#

9、JMF

- JMF (Java Media Framework) 是Java的一个类包。JMF技术提供媒体的处理能力，从而扩展了Java平台的功能。
 - 包括：媒体捕获、压缩、流转、回放，以及对各种主要媒体形式和编码的支持
- Java Media系列软件包括Java 3D、Java 2D、Java Sound和Java Advanced Imaging等API。JMF的组件结构非常的灵活，它的组件一般可以分成三个部分：
 - · Input
 - · process
 - · Output
- *Download Address:*

10、MM Cloud Services

■ 微软认知服务 (<https://azure.microsoft.com/zh-cn/services/cognitive-services/>)

Microsoft Azure

销售热线 1-800-867-1389

我的帐户 门户 搜索

为什么选择 Azure? 解决方案 产品 文档 价格 培训 Marketplace 合作伙伴 博客 资源 支持

免费帐户

1 选择 API


2 获取 API 密钥

3 开始使用 API

影像 语音 语言 知识 搜索

登录


The Microsoft Cognitive Toolkit



计算机影像 API

从图像中提取可操作信息
5,000 个事务，每分钟 20 个。


获取 API 密钥 >



情感 API 预览版

通过情绪识别实现个性化用户体验
30,000 个事务，每分钟 20 个。

获取 API 密钥 >



人脸 API

检测、识别、分析、组织和标记照片中的人脸
30,000 个事务，每分钟 20 个。

获取 API 密钥 >

■ IBM Watson认知计算平台

IBM

outthink

市场

Search



概览

Watson

IBM 云

行业应用

Watson，不止于人工智能

Watson 正与企业组织、政府机构、及各行各业合作，共同智胜我们所面临的挑战。

探索 Watson 更多应用



Watson Explorer



Watson 健康



Watson 物联网

Watson 是认知技术,它可以像人类一样思考



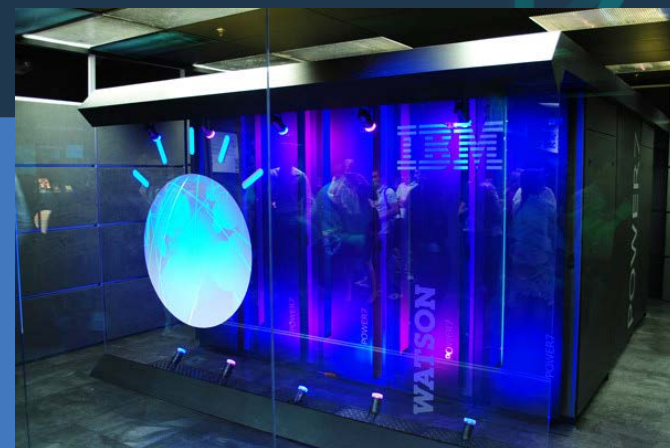
理解

通过自然语言理解 (Natural language understanding) 技术，分析所有类型的数据，包括文本、音频、视频和图像等非结构化数据。



推理

通过假设生成 (Hypothesis generation)，透过数据揭示洞察、模式和关系。将散落在各处的知识片段连接起来，进行推理、分析、对比、归纳、总结和论证，获取深入的洞察以及决策的证据。

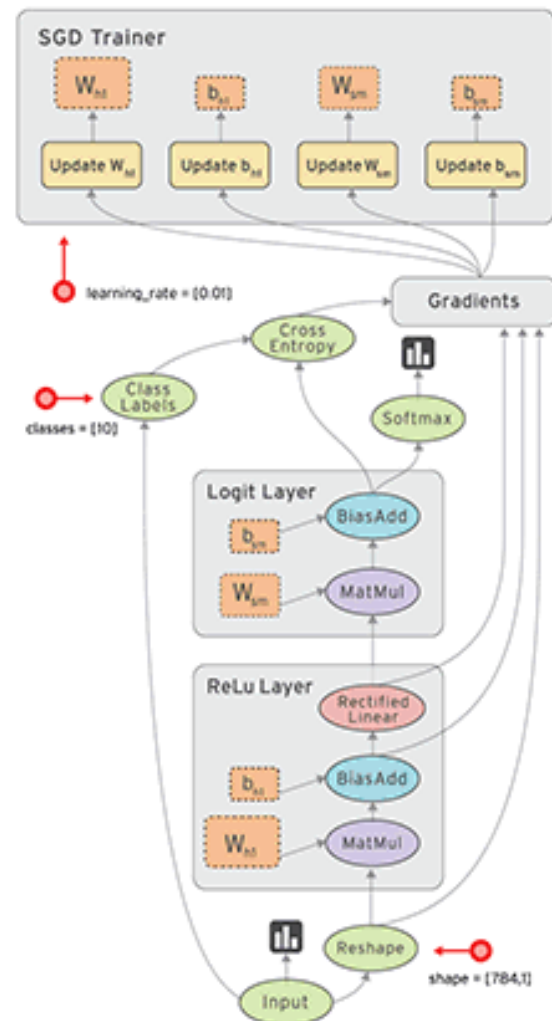
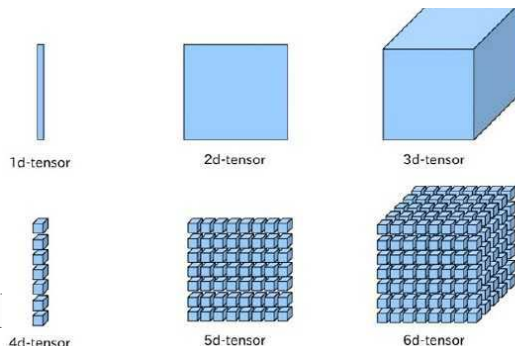
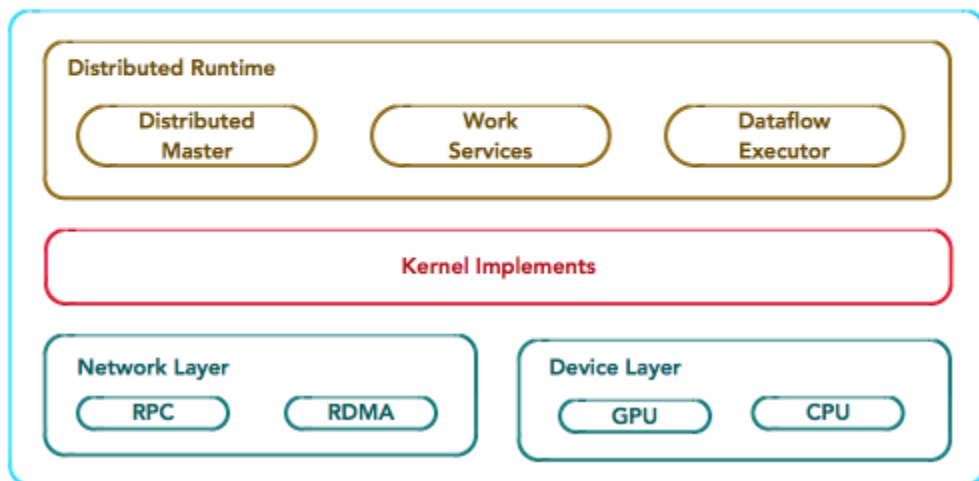


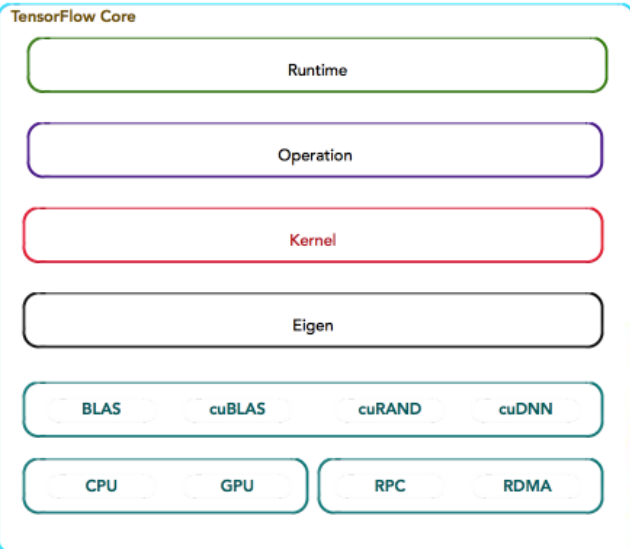
■ Google Tensorflow(<http://www.tensorfly.cn/>)

Front End



Exec System





TensorFlow Core

层	功能	组件
视图层	计算图可视化	TensorBoard
工作流层	数据集准备, 存储, 加载	Keras/TF Slim
计算图层	计算图构造与优化 前向计算/后向传播	TensorFlow Core
高维计算层	高维数组处理	Eigen
数值计算层	矩阵计算 卷积计算	BLAS/cuBLAS/cuRAND/cuDNN
网络层	通信	gRPC/RDMA
设备层	硬件	CPU/GPU

TensorFlow技术栈

■ 百度AI开放平台(百度大脑)<http://ai.baidu.com/>

Baidu 百度 | AI开放平台

首页

产品服务

行业生态

解决方案

新闻中心

SDK下载

文档与支持

合作伙伴

控制台

快速接入全球领先的AI服务



图像技术



自然语言



语音技术



视频技术



知识图谱



数据智能



增强现实



深度学习

apollo

百度Apollo计划

开放、完整、安全的无人驾驶软件平台

DUER OS

小度对话式人工智能系统

简单、便捷、低门槛的开放平台，让设备拥有与人类对话的能力，用声音链接世界

■ 讯飞开放平台(讯飞超脑)AIUI <http://www.xfyun.cn>



语音合成

精准、灵活、可靠



在线语音合成



离线语音合成



语音识别

准确率98%以上



语音听写



语音转写



语音唤醒



语音扩展

随时、随地、随心



语音评测



机器翻译



语音硬件

小身材，大智慧



麦克风阵列



语音合成芯片



离线识别模块



模式识别

闻声识人，一眼万年



人脸识别



声纹识别



基础服务

安全、便捷、强大



云储存



讯推



应用加密

本章重点

- MM API
- MCI
- GDI+
- DirectX
- XNA
- OpenGL
- JOGL
- JMF