Zuazo

Librería para manipular video en tiempo real

Oier Lauzirika Zarrabeitia

Estudiante de Ingeniería de Sonido e Imagen en ETSIST-UPM

Fase final del CUSL 13, 10/05/2019

ZUAZO

Real time video manipulation library

Concurso Universitario
De Software Libre

Contenidos

¿Qué es Zuazo?

- Una librería para manipular video en tiempo real
- Escrita en C++17, incluyendo la API
- Soporta distribuciones GNU/Linux basadas en Debian
- Emplea aceleración por GPU en la medida de lo posible
- Es seguro en hilos (thread safe)
- Relativamente fácil de usar

¿Por que decidí empezar Zuazo?

Ya existían alternativas

- MLT (Media Lovin' Toolkit)
- GStreamer
- ...

Entonces, ¿Por qué?

- Falta de documentación (MLT)
- Diseñada para C, API compleja (GStreamer)
- Sin énfasis en manipular video por hardware (Ambas)
- Sin énfasis en tiempo real (MLT)
- Ganas de hacer un proyecto relacionado con manipulación de video

Librerías de terceros

- Librería estándar de C++
- OpenGL ES 3.0
- GLFW3 (a reemplazar por SDL2)
- FFmpeg
- Magick++
- V4L2

Contenidos

Dináminca general I

- Todo se encuentra en el espacio de nombres (o en un subespacio de nombres de) `Zuazo`, o abreviado `zz`
- Para poder empezar a usar Zuazo, hay que inicializarlo con `Zuazo::init()`
- Al terminar, conviene liberar todo con `Zuazo::terminate()` (No hacerlo es de malas personas)
- Antes de llamar a `Zuazo::terminate()` **TODOS** los objetos de Zuazo deben haberse destruido o cerrado
- **TODOS** los comandos (excepto `init()` y `terminate()`) se deben de ordenar entre un `begin()` y un `end()` o equivalente, que llamaremos "contexto"
- Declarar un objeto de la clase `zz::Context`, es equivalente a hacer `begin()` y `end()` (Cuando `zz::Context` se construye se ejecuta `begin()` y cuando se destruye se ejecuta `end()`)

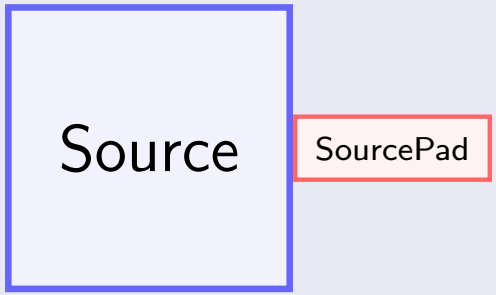
Dináminca general II

- Sólo puede haber un contexto activo en todo momento (está protegido con un mutex)
- Mientras que haya un contexto activo, el renderizado de Zuazo estará pausado (minimizar tiempo con contexto activo)
- Todos los cambios realizados en un mismo contexto ocurrirán en un mismo fotograma

```
zz::init(); //Inicializa Zuazo

zz::begin(); //Comenzamos a ordernar cosas a Zuazo
//[...]
zz::end(); //Hemos terminado de momento

if(){
 zz::Context ctx; //La vida de este objeto representa un begin() y end()
 //[...]
}

zz:terminate();//Libera todos los recursos adquiridos por Zuazo
```

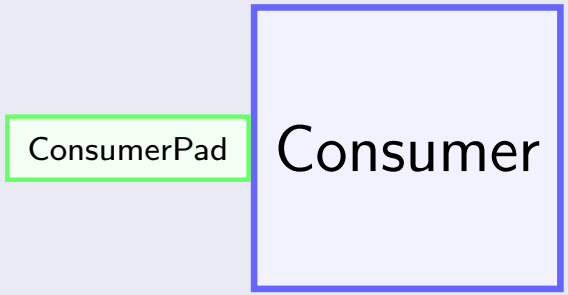
Source / Consumer / Processor

Source



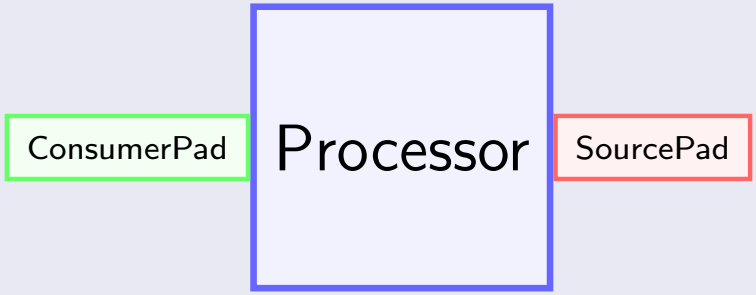
- Al menos un SourcePad
- Ningún ConsumerPad

Consumer



- Al menos un ConsumerPad
- Ningún SourcePad

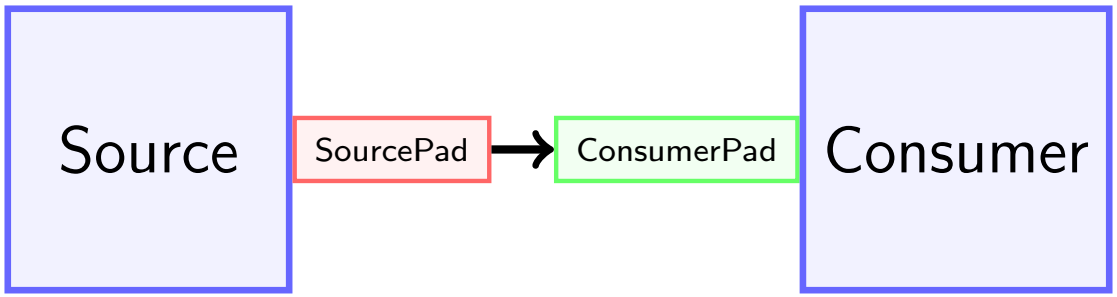
Processor



- El resto

Dináminca de flujo I

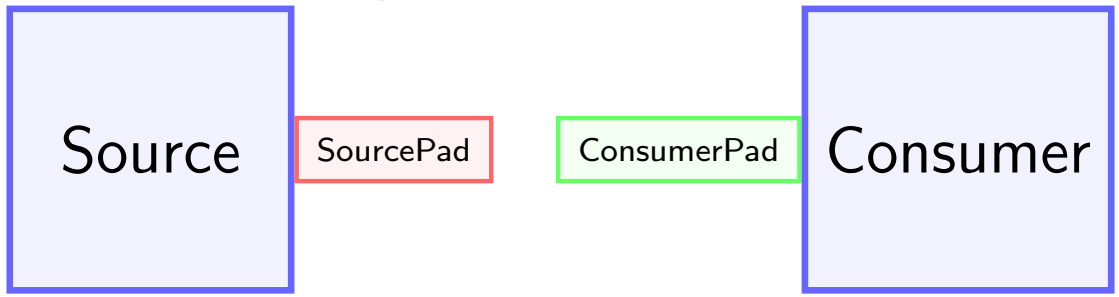
Un "Consumer" puede ser alimentado de un "Source"




```
FooSource source;
FooConsumer consumer;

//Los 3 son equivalentes
consumer.consumerPad << source.sourcePad;
source.sourcePad >> consumer.consumerPad;
consumer.setSource(&source.sourcePad);
```

Dináminca de flujo II

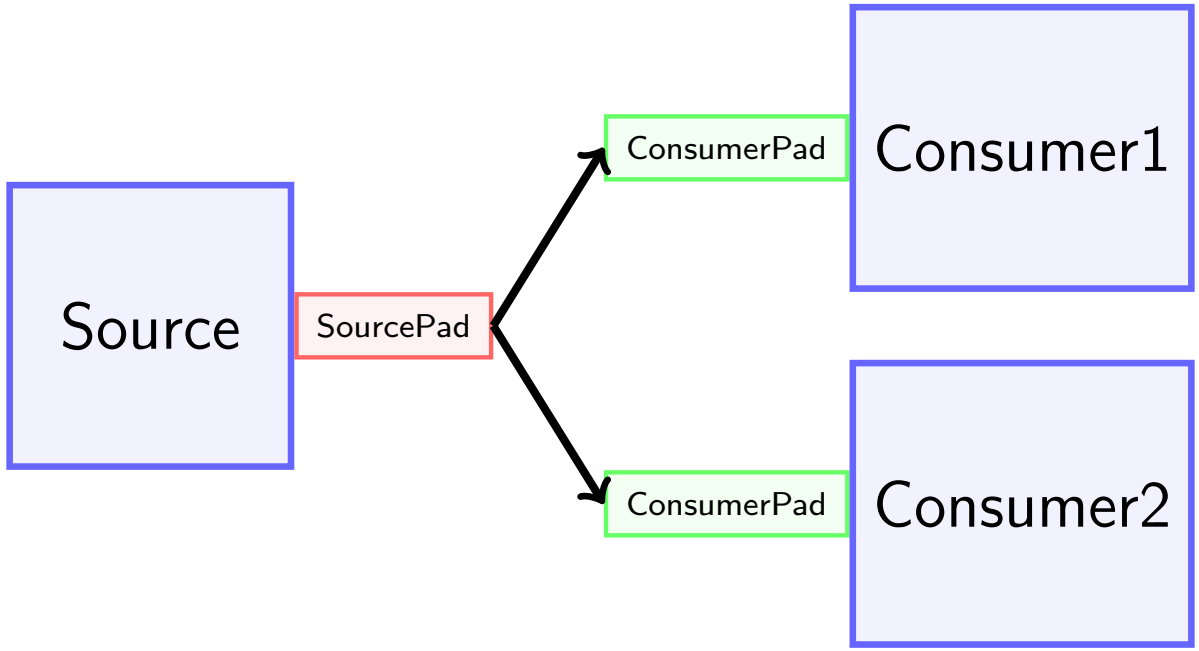
Los elementos pueden estar sin "conectarse"




```
consumer.consumerPad << nullptr; //Para dejar de alimentar un Consumer
```

Dináminca de flujo III

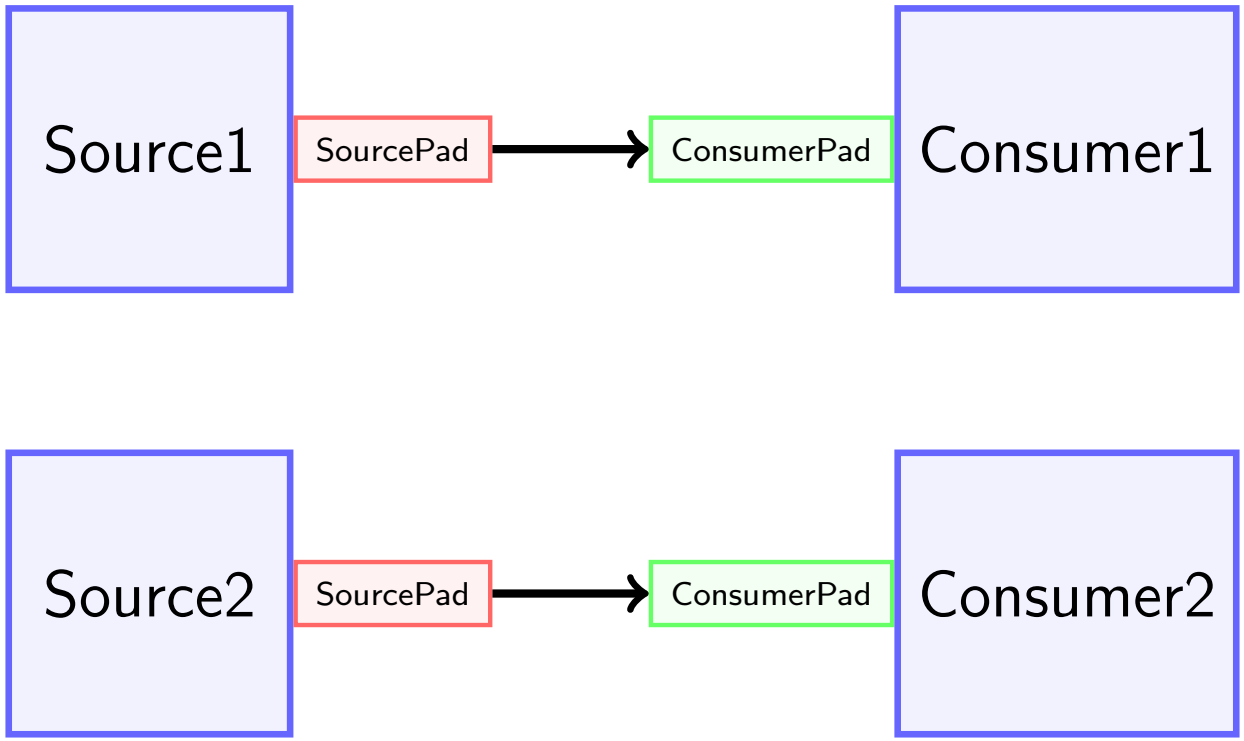
Un "Source" puede alimentar varios "Consumer"




```
consumer1.consumerPad << source.sourcePad;
consumer2.consumerPad << source.sourcePad;
```

Dináminca de flujo IV

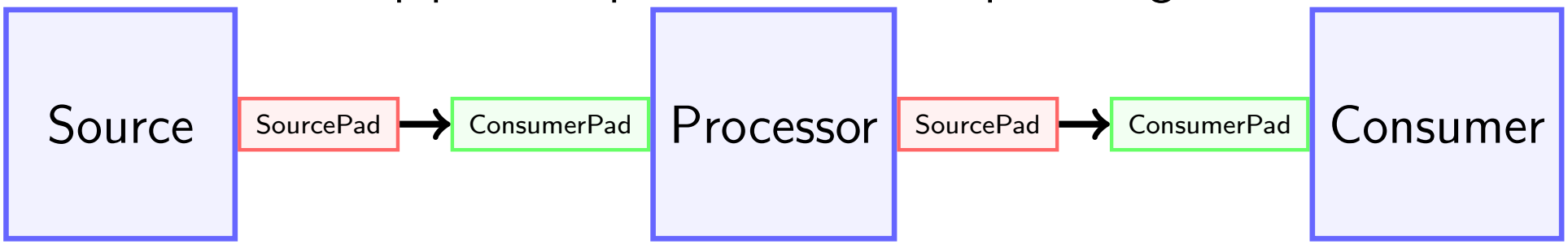
Puede haber varios "pipelines" independientes




```
consumer1.consumerPad << source1.sourcePad;
consumer2.consumerPad << source2.sourcePad;
```

Dináminca de flujo V

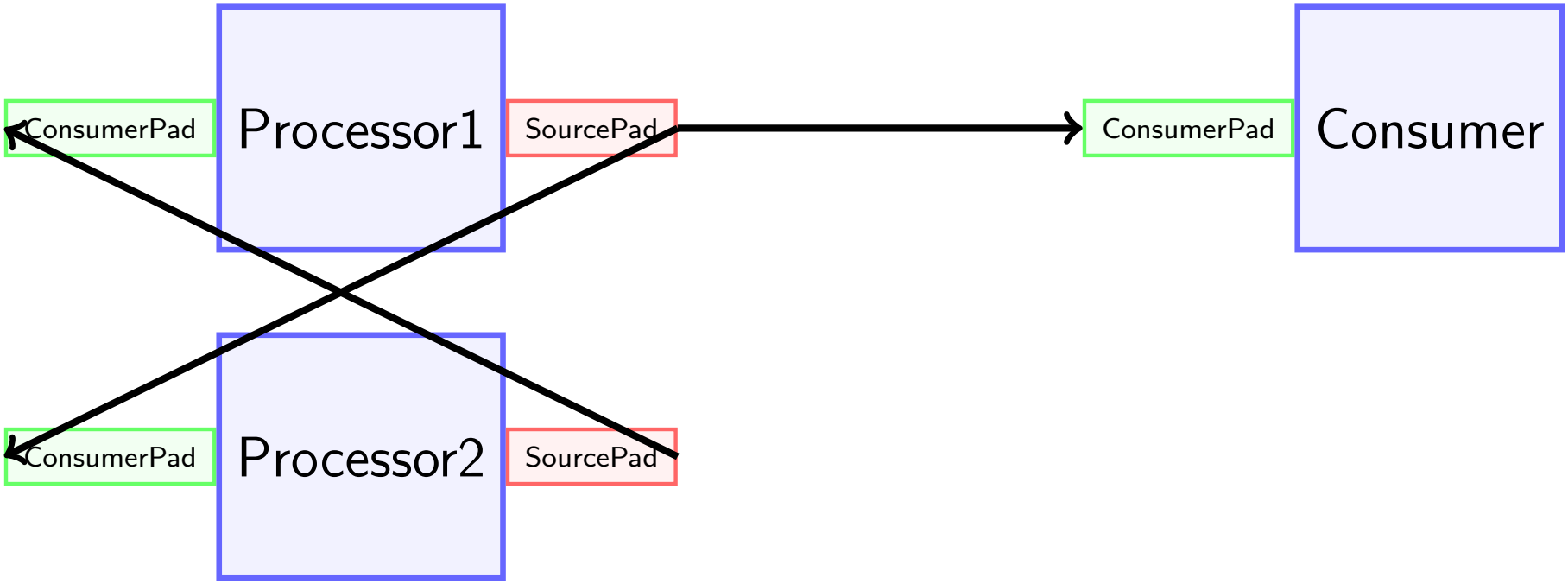
Los "pipelines" pueden tener cualquier longitud




```
consumer.consumerPad << processor.sourcePad;
processor.consumerPad << source.sourcePad;
```

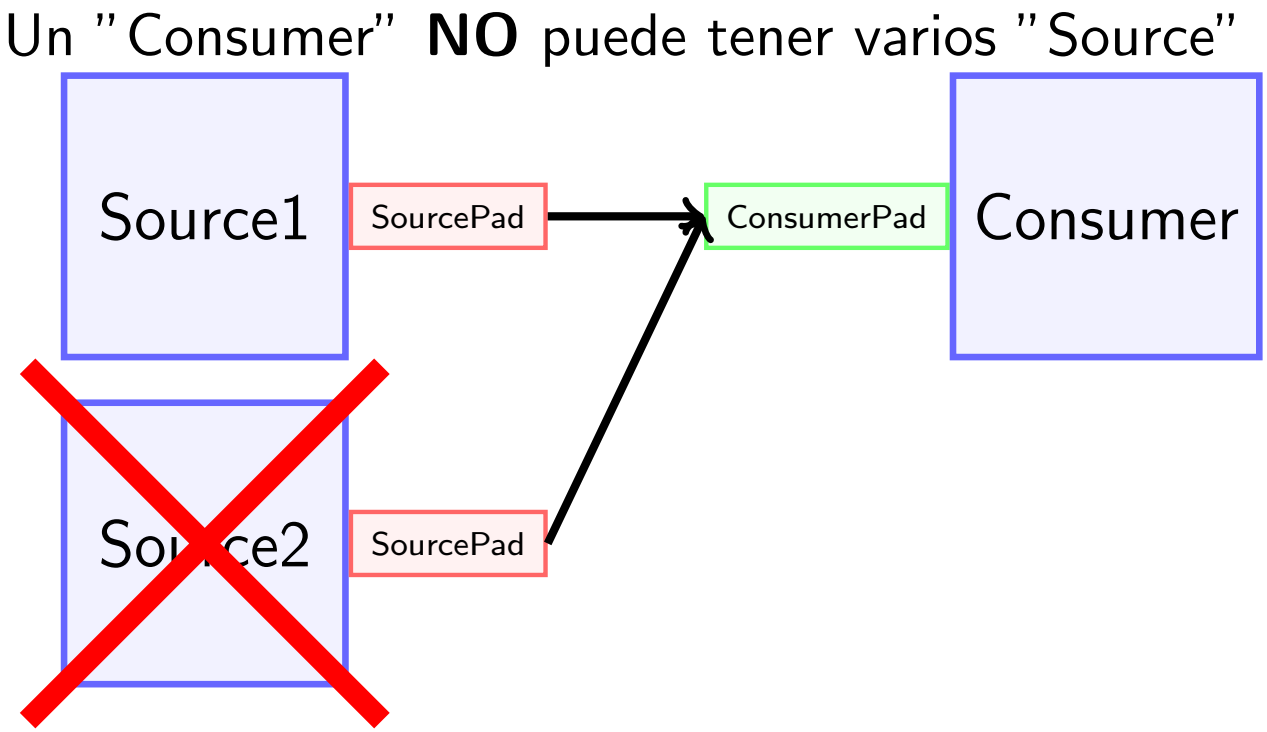
Dináminca de flujo VI

Los "pipelines" pueden tener recursividad (hasta un nivel finito)



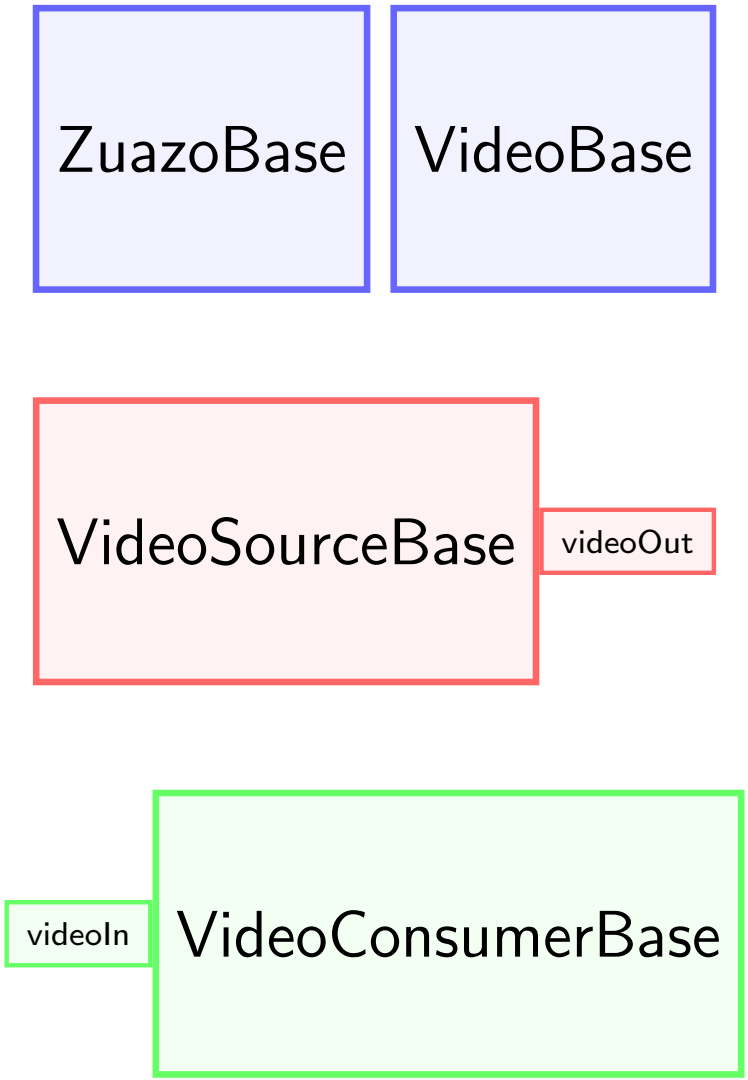

```
consumer.consumerPad << processor1.sourcePad;
processor1.consumerPad << processor2.sourcePad;
processor2.consumerPad << processor1.sourcePad;
processor1.setMaxRecursion(10);
processor2.setMaxRecursion(10); //En caso de ser distinto, se escojera el minimo
```

Dináminca de flujo VII



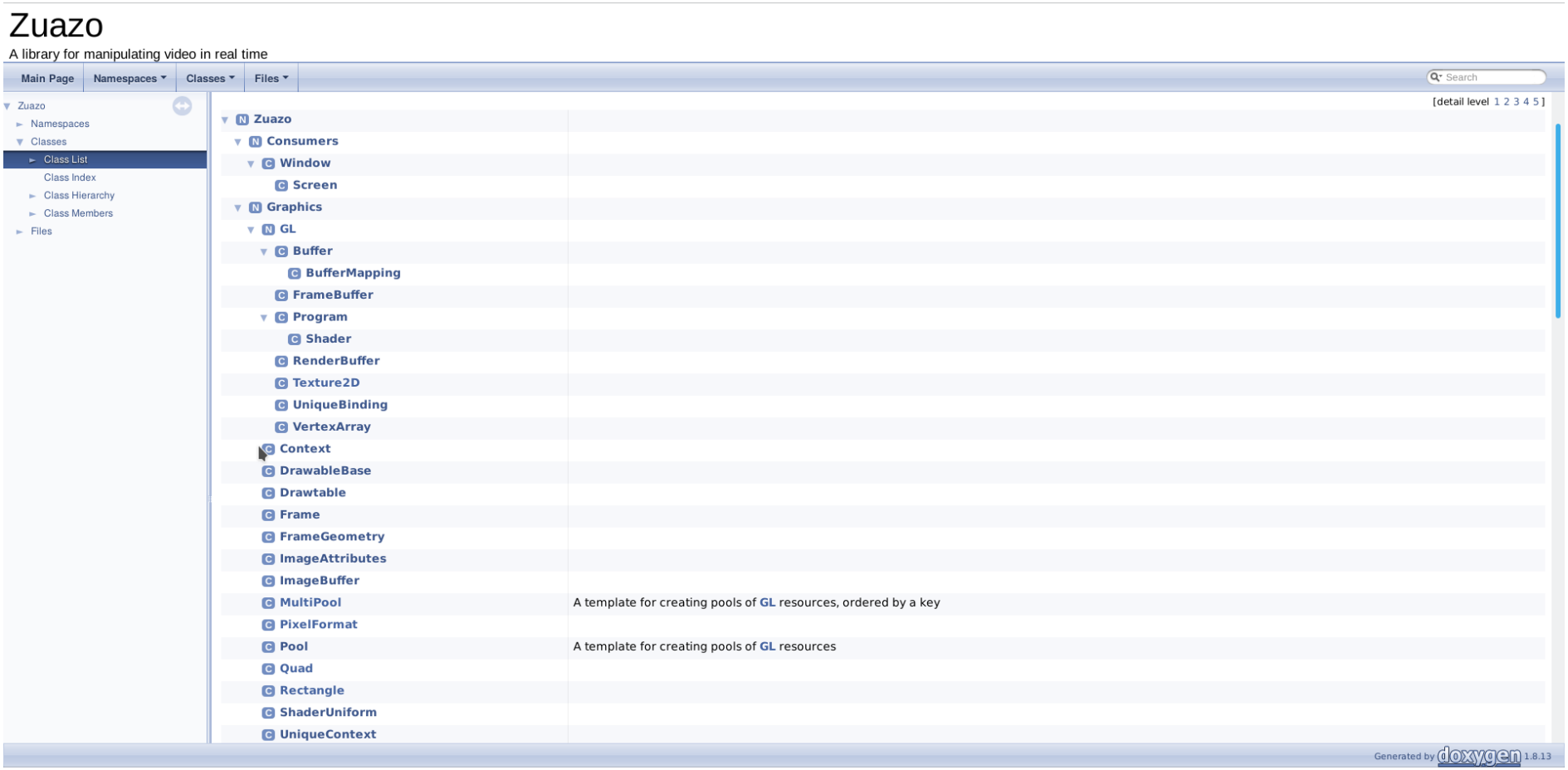

```
consumer.consumerPad << source1.sourcePad; //No sirve para nada, ya que acto seguido
     se "machacara"
consumer.consumerPad << source2.sourcePad; //Aqui se "machaca" la fuente anterior
```

Clases base de Zuazo



Contenidos

Documentación



ZuazoBase I

Public Member Functions

	ZuazoBase ()=default
	ZuazoBase (const **ZuazoBase** &other)=default
virtual	**~ZuazoBase** ()=default
virtual void	**open** ()
virtual void	**close** ()
bool	**isOpen** () const

API general del video I

VideoBase

Public Member Functions

	VideoBase ()=default
	VideoBase (const **VideoBase** &other)=default
	VideoBase (const **Utils::VideoMode** &vidMode)
virtual	**~VideoBase** ()=default
virtual bool	**supportsGettingPixelFormat** () const
virtual bool	**supportsSettingPixelFormat** () const
virtual bool	**supportsAnyPixelFormat** () const
virtual bool	**supportsListingPixelFormats** () const
virtual std::set< **Utils::PixelFormat** >	**getSupportedPixelFormats** () const
const **Utils::PixelFormat** &	**getPixelFormat** () const
virtual void	**setPixelFormat** (const **Utils::PixelFormat** &pixFmt)
virtual bool	**supportsGettingResolution** () const
virtual bool	**supportsSettingResolution** () const
virtual bool	**supportsAnyResolution** () const
virtual bool	**supportsListingResolutions** () const
virtual std::set< **Utils::Resolution** >	**getSupportedResolutions** () const
const **Utils::Resolution** &	**getResolution** () const
virtual void	**setResolution** (const **Utils::Resolution** &res)
virtual bool	**supportsGettingCodec** () const
virtual bool	**supportsSettingCodec** () const
virtual bool	**supportsAnyCodec** () const
virtual bool	**supportsListingCodecs** () const
virtual std::set< **Utils::Codec** >	**getSupportedCodecs** () const
const **Utils::Codec** &	**getCodec** () const
virtual void	**setCodec** (const **Utils::Codec** &codec)

API general del video II

virtual bool	**supportsGettingFramerate** () const
virtual bool	**supportsSettingFramerate** () const
virtual bool	**supportsAnyFramerate** () const
virtual bool	**supportsListingFramerates** () const
virtual std::set< **Utils::Rational** >	**getSupportedFramerates** () const
const **Utils::Rational** &	**getFramerate** () const
virtual void	**setFramerate** (const **Utils::Rational** &rat)
virtual bool	**supportsGettingProgressive** () const
virtual bool	**supportsSettingProgressive** () const
bool	**isProgressive** () const
virtual void	**setProgressive** (bool progressive)
virtual bool	**supportsGettingVideoMode** () const
virtual bool	**supportsSettingVideoMode** () const
virtual bool	**supportsAnyVideoMode** () const
virtual bool	**supportsListingVideoModes** () const
virtual std::set< **Utils::VideoMode** >	**getSupportedVideoModes** () const
const **Utils::VideoMode** &	**getVideoMode** () const
virtual void	**setVideoMode** (const **Utils::VideoMode** &videoMode)

API general del video III

Public Member Functions

	VideoMode ()=default
	VideoMode (const **VideoMode** &other)=default
	~VideoMode ()=default
constexpr int	**operator==** (const **VideoMode** &other) const
constexpr int	**operator!=** (const **VideoMode** &other) const
constexpr int	**operator<** (const **VideoMode** &other) const
constexpr int	**operator>** (const **VideoMode** &other) const
constexpr int	**operator<=** (const **VideoMode** &other) const
constexpr int	**operator>=** (const **VideoMode** &other) const
constexpr **ImageAttributes**	**toImageAttributes** () const

Public Attributes

PixelFormat	**pixFmt** =**PixelFormats::PIX_FMT_NONE**
Resolution	**res** =**Resolution**(0, 0)
Codec	**codec** =**Codecs::CODEC_NONE**
Rational	**frameRate** =**Rational**(0, 0)
bool	**progressive** =true

Detailed Description

Definition at line **11** of file **VideoMode.h**.

API general del video IV

VideoConsumer

Public Member Functions

	VideoConsumerBase (**VideoConsumer** &consumer)
	VideoConsumerBase (**VideoConsumer** &consumer, const **Utils::VideoMode** &videoMode)
	VideoConsumerBase (const **VideoConsumerBase** &other)=default
virtual	**~VideoConsumerBase** ()=default
virtual bool	**supportsGettingScalingMode** () const
virtual bool	**supportsSettingScalingMode** () const
Utils::ScalingMode	**getScalingMode** () const
virtual void	**setScalingMode** (**Utils::ScalingMode** scaling)
virtual bool	**supportsGettingScalingFilter** () const
virtual bool	**supportsSettingScalingFilter** () const
Utils::ScalingFilter	**getScalingFilter** () const
virtual void	**setScalingFilter** (**Utils::ScalingFilter** scaling)

API general del video V

ScalingFilter

enum **Zuazo::Utils::ScalingFilter**

Enumerator
Nearest
Bilinear
Bicubic

Definition at line **5** of file **ScalingFilter.h**.

ScalingMode

enum **Zuazo::Utils::ScalingMode**

Enumerator
Stretched
Boxed
Cropped
ClampVert
ClampHor

Definition at line **5** of file **ScalingMode.h**.

Contenidos

Entradas de video disponibles

- Archivos de video

Entradas de video disponibles

- Archivos de video
- Archivos de imágenes

Entradas de video disponibles

- Archivos de video
- Archivos de imágenes
- Archivos SVG

Entradas de video disponibles

- Archivos de video
- Archivos de imágenes
- Archivos SVG
- Dispositivos compatible con V4L2

Salidas de video disponibles

- Ventana
- Monitor

Efectos basados en shaders de fragmento

- Corrección de tono, saturación y luminosidad (HSL)

Efectos basados en shaders de fragmento

- Corrección de tono, saturación y luminosidad (HSL)
- Corrección de brillo y contraste

Efectos basados en shaders de fragmento

- Corrección de tono, saturación y luminosidad (HSL)
- Corrección de brillo y contraste
- Color a escala de grises

Efectos basados en shaders de fragmento

- Corrección de tono, saturación y luminosidad (HSL)
- Corrección de brillo y contraste
- Color a escala de grises
- Chroma key

Efectos basados en shaders de fragmento

- Corrección de tono, saturación y luminosidad (HSL)
- Corrección de brillo y contraste
- Color a escala de grises
- Chroma key
- Representación en luminancia del canal alpha

Efectos basados en shaders de fragmento

- Corrección de tono, saturación y luminosidad (HSL)
- Corrección de brillo y contraste
- Color a escala de grises
- Chroma key
- Representación en luminancia del canal alpha
- Invertir color

Efectos basados en shaders de fragmento

- Corrección de tono, saturación y luminosidad (HSL)
- Corrección de brillo y contraste
- Color a escala de grises
- Chroma key
- Representación en luminancia del canal alpha
- Invertir color
- A definir por el usuario

Compositor

- Combina varias fuentes en una sola salida

Compositor

- Combina varias fuentes en una sola salida
- Se organiza en capas

Compositor

- Combina varias fuentes en una sola salida
- Se organiza en capas
- Entorno 3D: Cámara y capas transformables en el espacio

Contenidos

Programa "¡Hola Mundo!" I

../code/HolaMundo.cpp

```
/* COMO COMPILAR:
 * g++ HolaMundo.cpp -o HolaMundo -std=c++17 -lzuazo -lavutil -lavformat -lavcodec -
      lswscale -lglfw -lMagick++-6.Q16 -lMagickWand-6.Q16 -lMagickCore-6.Q16
 */

#include<zuazo/Includes.h>
#include<cstdio>

int main(){
    zz::init(); //Inizializa Zuazo
    zz::begin(); //Comenzamos a configurar

    //Abre el primer dispositivo V4L2. Devuelve un std::unique_ptr
    auto fuente=zz::videoSourceFromFile("/dev/video0");

    //Crea una ventana con los parametros indicados
    zz::Utils::VideoMode modoDeVideo;
    modoDeVideo.res=zz::Utils::Resolution(1280, 720);
    modoDeVideo.frameRate=zz::Utils::Rational(30.0);
    zz::Consumers::Window ventana(modoDeVideo, "Hola␣Mundo");

    ventana.videoIn << fuente->videoOut; //Establece la entrada

    zz::end(); //Indicar que hemos terminado de configurar
    getchar(); //Esperar
    zz::begin(); //Comenzamos a configurar
```

Programa "¡Hola Mundo!" II

```
    //Eliminar objetos antes de llamar a terminate()
    fuente.reset(); //Destruye "fuente"
    ventana.close();

    zz::end(); //Indicar que hemos terminado de configurar
    zz::terminate();
}
```

Creación de shaders de fragmento I

../code/quantize.glsl

```
R""(

uniform shaderFxDataBlock{
  int nivelesR;
  int nivelesG;
  int nivelesB;
  //[...] otras variables
};

vec4 shaderFx(sampler2D tex, vec2 texCoord){
  vec4 texColor=texture(tex, texCoord); //Obtiene el color del pixel deseado
  vec3 niveles=vec3(nivelesR, nivelesG, nivelesB); //Crea un vector de niveles

  vec3 cuantizado=round(texColor.rgb * niveles) / niveles;

  //Devuelve el vector de color con el resultado
  return vec4(cuantizado, texColor.a);
}

)""
```

Creación de shaders de fragmento II

../code/Cuantizar.h

```
#pragma once

#include "zuazo/Includes.h"

class Cuantizar : public Zuazo::Processors::ShaderEffect{
public:
    Cuantizar();
    Cuantizar(const Zuazo::Utils::VideoMode& vidMode);
    Cuantizar(const Cuantizar& other)=delete;
    Cuantizar(Cuantizar&& other)=default;
    virtual ~Cuantizar()=default;

    void  setNivelesRojo(int niveles);
    int   getNivelesRojo() const;

    void  setNivelesVerde(int niveles);
    int   getNivelesVerde() const;

    void  setNivelesAzul(int niveles);
    int   getNivelesAzul() const;
private:
    static const std::string s_shaderSrc;
};

inline void Cuantizar::setNivelesRojo(int niveles){
    ShaderEffect::setParam("nivelesR", niveles);
}
```

Creación de shaders de fragmento III

```
inline int Cuantizar::getNivelesRojo() const{
    const int* niveles=ShaderEffect::getParam<int>("nivelesR");
    return niveles ? *niveles : 0;
}

inline void Cuantizar::setNivelesVerde(int niveles){
    ShaderEffect::setParam("nivelesG", niveles);
}

inline int Cuantizar::getNivelesVerde() const{
    const int* niveles=ShaderEffect::getParam<int>("nivelesG");
    return niveles ? *niveles : 0;
}

inline void Cuantizar::setNivelesAzul(int niveles){
    ShaderEffect::setParam("nivelesB", niveles);
}

inline int Cuantizar::getNivelesAzul() const{
    const int* niveles=ShaderEffect::getParam<int>("nivelesB");
    return niveles ? *niveles : 0;
}
```

Creación de shaders de fragmento IV

../code/Cuantizar.cpp

```
#include "Cuantizar.h"

const std::string Cuantizar::s_shaderSrc(
    #include "quantize.glsl"
);

Cuantizar::Cuantizar() :
ShaderEffect(s_shaderSrc)
{
    setNivelesRojo(2);
    setNivelesVerde(2);
    setNivelesAzul(2);
}

Cuantizar::Cuantizar(const Zuazo::Utils::VideoMode& vidMode) :
ShaderEffect(vidMode, s_shaderSrc)
{
    setNivelesRojo(2);
    setNivelesVerde(2);
    setNivelesAzul(2);
}
```

Creación de shaders de fragmento V

../code/CuantizarApp.cpp

```
/* COMO COMPILAR:
 * g++ CuantizarApp.cpp Cuantizar.cpp -o CuantizarApp -std=c++17 -lzuazo -lavutil -
      lavformat -lavcodec -lswscale -lglfw -lMagick++-6.Q16 -lMagickWand-6.Q16 -
      lMagickCore-6.Q16
 */

#include<zuazo/Includes.h>
#include<cstdio>
#include <iostream>
#include "Cuantizar.h"

int main(){
    zz::init(); //Inizializa Zuazo
    zz::begin(); //Comenzamos a configurar

    //Abre el primer dispositivo V4L2. Devuelve un std::unique_ptr
    auto fuente=zz::videoSourceFromFile("/dev/video0");

    //Crea una ventana con los parametros indicados
    zz::Utils::VideoMode modoDeVideo;
    modoDeVideo.res=zz::Utils::Resolution(1280, 720);
    modoDeVideo.frameRate=zz::Utils::Rational(30.0);
    modoDeVideo.pixFmt=zz::Utils::PixelFormats::PIX_FMT_RGB32;

    zz::Consumers::Window ventana(modoDeVideo, "Cuantizar");
    Cuantizar cuant(modoDeVideo);
```

Creación de shaders de fragmento VI

```
    cuant.videoIn << fuente->videoOut;
    ventana.videoIn  << cuant.videoOut;

    zz::end(); //Indicar que hemos terminado de configurar
    std::cout << cuant.getShaderLog(); //Muestra el log de la compilacion del shader
    getchar(); //Esperar
    zz::begin(); //Comenzamos a configurar

    //Eliminar objetos antes de llamar a terminate()
    fuente.reset(); //Destruye "fuente"
    ventana.close();
    cuant.close();

    zz::end(); //Indicar que hemos terminado de configurar
    zz::terminate();
}
```

Contenidos

Cosas por hacer a corto plazo

- Cambiar librería de ventanas (GLFW3 por SDL2)
  - Permite integración con QT
  - Reescribir la clase "Window", con el debido soporte de pantalla completa y gestión de eventos
- Completar efectos ya existentes
  - Mejor gestión de parámetros
  - Mayor número de parámetros
- Más tipos de capas en el compositor
  - Capa de video deformable
  - Formas geométricas
- Terminar de codificar los métodos `init()` y `terminate()`
- Añadir las extensiones que faltan a `videoSourceFromFile([...])`
- Añadir los comentarios de Doxygen

Cosas por hacer a largo plazo

- Añadir más entradas y salidas
  - Tarjetas Blackmagick Decklink
  - Salida a fichero de video por FFmpeg
  - E/S en streaming por FFmpeg
  - Superficie de Cairo
  - Generador de texto
  - Script personalizado de OpenGL
  - ...
- Añadir más tipos de procesadores y efectos
  - Un *keyer* completo, estilo compositor
  - *Buffer* de fotogramas (para *replays*)
  - Efectos de difusión (Gaussiana, desenfoque...)
- Añadir soporte para audio
- ...

Contenidos

Posibles aplicaciones

- Producción **lineal** de video (retransmisiones en directo)
  - Insertar gráficos
  - Conmutar entre entradas
  - Composiciones de múltiples entradas
  - Aplicar efectos
- Realidad aumentada
  - Estudios virtuales
- ...

Pero, ¿Se pueden hacer memes?

Contenidos

Bugs

- Transparencia independiente del orden (OIT) en el compositor
- Renderizado de color a formato de pixeles del tipo "Escala de grises" hace incorrectamente la conversión de luminancia

Agradecimientos

- Blog "Software libre en UPM" mantenida por Laura Arjona Reina
- Hackelarre
- Organización del CUSL
- Mis padres, familia y amigos

ZUAZO
Real time video manipulation library

Concurso Universitario
De Software Libre

Eskerrik asko

(Muchas gracias)

Diapositivas y el código mostrado

`https://github.com/oierlauzi/zuazo-cusl13`

Forja de Zuazo

`https://github.com/oierlauzi/zuazo`