Zuazo

Librería para manipular video en tiempo real

Oier Lauzirika Zarrabeitia
oier.lauzirika.zarrabeitia@alumnos.upm.es

Estudiante de Ingeniería de Sonido e Imagen en la ETSI de Sistemas de Telecomunicación,
Universidad Politécnica de Madrid

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

Librerías de terceros

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

Dináminca

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
- Representación en luminancia del canal alpha

Efectos basados en shaders de fragmento

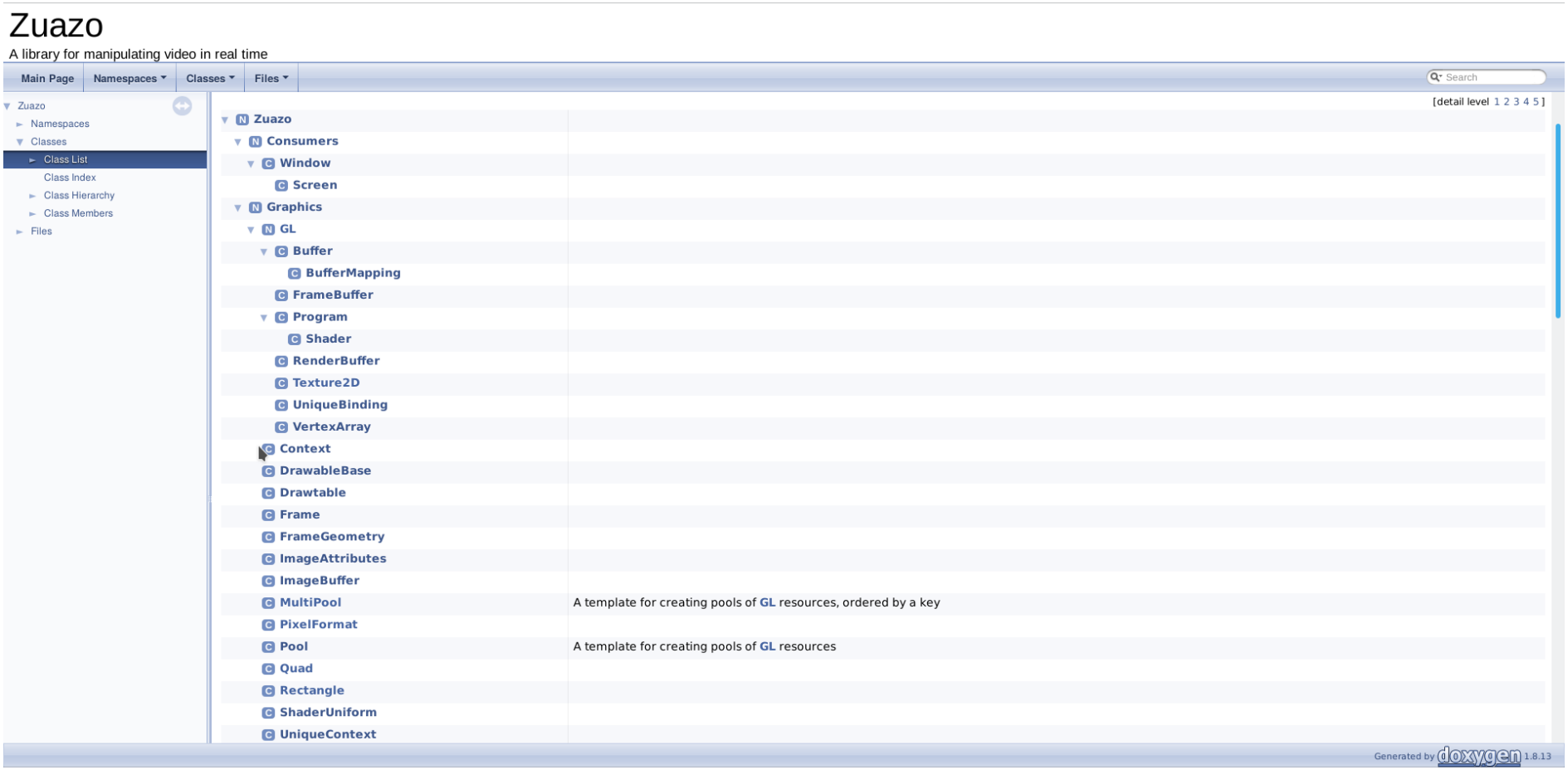
- Corrección de tono, saturación y luminosidad (HSL)
- Corrección de brillo y contraste
- Color a escala de grises
- Representación en luminancia del canal alpha
- Invertir color

Efectos basados en shaders de fragmento

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

Contenidos

Documentación I



Documentación II

Public Member Functions

	ShaderEffect (const **Utils::VideoMode** &vidMode, const std::string &fragShader)
	ShaderEffect (const std::string &fragShader)
	ShaderEffect (const **ShaderEffect** &other)=delete
	ShaderEffect (**ShaderEffect** &&other)=default
virtual	**~ShaderEffect** ()
template<typename T > void	**setParam** (const std::string &pname, const T &data)
template<typename T > const T *	**getParam** (const std::string &pname) const
std::string	**getShaderLog** () const
virtual std::set< **Utils::PixelFormat** >	**getSupportedPixelFormats** () const override
virtual void	**setPixelFormat** (const **Utils::PixelFormat** &pixFmt) override
virtual void	**setResolution** (const **Utils::Resolution** &res) override
void	**open** () override
void	**close** () override
void	**update** () const override

- **Public Member Functions inherited from Zuazo::Video::TVideoSourceBase< Video::LazyVideoSourcePad >**
- **Public Member Functions inherited from Zuazo::Video::VideoSourceBase**
- **Public Member Functions inherited from Zuazo::Video::VideoBase**
- **Public Member Functions inherited from Zuazo::Video::TVideoConsumerBase< Video::VideoConsumerPad >**
- **Public Member Functions inherited from Zuazo::Video::VideoConsumerBase**
- **Public Member Functions inherited from Zuazo::ZuazoBase**

Additional Inherited Members

- **Public Attributes inherited from Zuazo::Video::VideoSourceBase**
- **Public Attributes inherited from Zuazo::Video::VideoConsumerBase**
- **Protected Attributes inherited from Zuazo::Video::TVideoSourceBase< Video::LazyVideoSourcePad >**
- **Protected Attributes inherited from Zuazo::Video::VideoBase**

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

Estado del arte

Cosas por hacer

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