Гамма Коррекция

Выпуск 0

Gorichenko Ivan

окт. 17, 2022

Оглавление

– Гамма-коррекция — предыскажения яркости чёрно-белого или цветоделённых составляющих цветного изображения при его записи в телевидении и цифровой фотографии. В качестве передаточной функции при гамма-коррекции чаще всего используется степенная в виде

Vout = A * Vin^gamma

Где A служит коэффициентом, а входные Vin и выходные Vout значения — неотрицательные вещественные числа. В общем случае, если A = 1, то входные и выходные значения находятся в пределах от 0 до 1. При равенстве gamma единице характеристика передачи полутонов линейна и перепады освещённости объекта в светах и тенях отображаются одинаково.

Содержание

Глава 1

Класс DisplayCustomGamma

Класс отвечает за выбор и сброс настроек гаммы.

Список 1: core/displaycustomgamma.h

```
class DisplayCustomGamma : public QWidget
{
    Q_OBJECT
public:
    explicit DisplayCustomGamma(const QDomNode &node, QWidget *parent = 0);
    ~DisplayCustomGamma();
    void paintEvent(QPaintEvent *event);
    static void resetGamma();

private:
    void showEvent(QShowEvent *);
    void hideEvent(QHideEvent *);

    bool eventFilter(QObject *object, QEvent *event);
    QPointer<QuickMenu> quickMenu;
    QPointer<TextBar> textOk;
    QPointer<QTimer> updateTimer;
    int prevGamma;//!< Гамма до входа в режим настройки. Используется для востановления гаммы
↪после выхода
signals:
    void exit();
};
```

1.1 Конструктор класса

Конструктор класса создает объект QuickMenu (параметры меню берутся из файла xml/menu_display_custom_gamma.xml) и выставляет для него параметры (геометрические, таймер), также изменяет параметр прозрачности дисплея и выключает ключевой цвет через объект класса QDisplay. Вызывает метод setValueWrapper, через который создает сигнал и отправляет сообщение по DBus.

Список 2: core/widget.cpp.1426

```
QDBusMessage mess =  QDBusMessage::createSignal("/teplovisor/displaydemon","displaydemon.
↪teplovisor.iwt","set_gamma");       // не место аргуменов подставлены значения
...
if (signalArgType == 0)
    mess << value.toInt();                          // в переменной value лежит значение prevGamma␣
↪класса DisplayCustomGamma
...
if(QDBusConnection::sessionBus().send(mess) == 0)
...
```

Список 3: core/displaycustomgamma.cpp.14

```
DisplayCustomGamma::DisplayCustomGamma(const QDomNode &node, QWidget *parent);
```

1.2 Метод paintEvent

Выводит окно с черно-белым градиентом, отображая на нем сетку со значениями настроек, которые возвращает статический метод **GenericConfig::getGenericIntValue**.

Список 4: core/displaycustomgamma.cpp.64

```
void DisplayCustomGamma::paintEvent(QPaintEvent *event);
```

1.3 Статический метод resetGamma

Статический метод сбрасывает текущие настройки гаммы дисплея до стандартных, используя функцию **GenericConfig::setGenericIntValue**. В конце выполнения выводит сообщение на дисплей.

Список 5: core/displaycustomgamma.cpp.113

```
void DisplayCustomGamma::resetGamma();
```

1.4 Метод setGenericIntValue

Функция определяет тип настройки: в глобальной переменной “map_target” по ключу (group / groupe+key). В зависимости от типа добавляет настройку.

Список 6: part_genericconfigs/genericconfig.cpp.242

```
void GenericConfig::setGenericIntValue(const QString &group, const QString &key, const int &value);
```

В случае с методом resetGamma все настройки отностся к типу Global, что значит он создает 17 настроек QSettings следующего образца:

Список 7: part_genericconfigs/genericconfig.cpp.248

```
settings.beginGroup("displaygamma");
setValue("displaygamma custom_user_X", Y);
```

Где, X - это число от 0 до 16. Значения Y:

0xD0	0xE3	0xF6	0x109
0x11C	0x12F	0x142	0x155
0x17B	0x18E	0x1A1	0x1B4
0x1C7	0x1DA	0x1ED	0x200

1.5 Приватные методы класса DisplayCustomGamma

Расположены в файле core/displaycustomgamma.cpp.

Список 8: core/displaycustomgamma.cpp.146

```
void DisplayCustomGamma::showEvent(QShowEvent *);
void DisplayCustomGamma::hideEvent(QHideEvent *);
bool DisplayCustomGamma::eventFilter(QObject *object, QEvent *event);
```

ShowEvent регистрирует фильтр событий в родительском классе на объект, hideEvent удаляет. EventFilter - метод, через который проходят прерывания, если фильтр установлен.

Глава 2

Виджеты гаммы

Конструктор класса QWidget соединяет сигнал объекта ServiceMenu clickGammaCustom со слотом toMenuGammaDisplay, который создает и выводит на дисплей объект класса Menu, на основе файла *sensors_configs.xml*. **(файла нет)**

Список 1: core/widget.cpp.212

```
QObject::connect(serviceMenu, SIGNAL(clickGammaCustom()), this, SLOT(toMenuGammaDisplay()));
```

2.1 Публичный слот класса Widget setValueWrapper

Список 2: core/widget.cpp.650

```
void setValueWrapper(QString name, QVariant value, bool isNeedWriteSetting=true);


else if(target == DISPLAYGAMMA_CONFIG_GROUP){
    isNeedSendSignal =true;
    objectPath = DISPLAYDEMON_OBJECT_PATH;
    interface = DISPLAYDEMON_INTERFACE_NAME;
    if(property == "reset_gamma"){
        DisplayCustomGamma::resetGamma();
        isNeedWriteSetting = false;
   }
}
...
if(isNeedSendSignal) // если требуется послать сигнал
{
    if(namesignal.isEmpty()) // имя сигнала может быть задано отдельно. Если оно не задано, то мы␣
↪его генерируем сами.
        namesignal = "set_"+property;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    QDBusMessage mess =  QDBusMessage::createSignal(objectPath,interface,namesignal);
    if (signalArgType == 0)
                    mess << value.toInt();
            else if (signalArgType == 1)
                    mess << value.toString();
    else if (signalArgType == 2){
        mess << value.toDouble();
    }

    if(QDBusConnection::sessionBus().send(mess) == 0){
        CO_WARN("Cannot send signal ");
    }else{
        DBG("signal sended");
    }
}
```

Глава 3

Класс Menu

Объект класса Menu для главного меню приложения создается на основе файла menu.xml. В этом файле заданы настройки всех подменю, в том числе и подменю гамма-коррекции дисплея.

Список 1: core/menu.cpp.2432

```
void Menu::set();
```

Метод класса set в случае выбора пользователем определенной конфигурации гаммы вызывает метод Widget::setValueWrapper, в качестве аргументов он передает *valuename* и *valueid* из menu.xml.

Список 2: core/menu.cpp.2663

```
QString valuename =  items->at(tmp)->getValuename();
QString valueid =  items->at(tmp)->getValueid();
if( (!valueid.isEmpty()) && (!valuename.isEmpty()))
    ((Widget*)parentwidget)->setValueWrapper(valuename, valueid);
if(dynamic_cast<RadioButton*>(items->at(tmp)))
    emit itemValueChanged();
```

Список 3: xml/menu.xml.5805

```
<menu name="displaymodegamma">
    <item type="radiobutton">
        <name>custom_1</name>
        <checked>yes</checked>
        <tickicon>:/res/res/icons/tick.png</tickicon>
        <valuename>display_gamma</valuename>
        <valueid>0</valueid>
    </item>
```

Получается аргументами функции setValueWrapper(valuename, valueid) будут

valuename:

«display_gamma»

valueid:

«0» - «7»

Результатом выполнения *функции* будет создание сигнала и отправка сообщения (0 - 7) по DBus.