# Graficzny interfejs użytkownika --- szkic do wklejenia i pocięcia

Wstęp

Przeprowadzanie jakichkolwiek testów z udziałem losowych osób wymaga, aby testerzy byli prowadzeni przez cały test niejako za rękę. Wiążę się to z przemyśleniem i zaprojektowaniem interfejsu użytkownika w taki sposób, aby zapewnić możliwie jak największą czytelność i ergonomie procesu testowania. Interfejs użytkownika musi pozwalać na oglądanie filmów i wystawianie ocen, do interfejsu można wprowadzić dodatkowe panele pozwalające na informowanie testera, czy też pobieranie od niego dodatkowych informacji.

Narzędzia

*QT* to zestaw bibliotek dedykowanych dla m.in. języka C++ pozwalający na tworzenie zaawansowanych interfejsów użytkownika. *QT* zawiera również elementy pozwalające na obsługę procesów, sieci i grafiki trójwymiarowej a także integrację z bazami danych, posiada także narzędzia pozwalające na przeprowadzenie lokalizacji dla innych wersji językowych programu. Wykorzystane narzędzia zostały szczegółowo omówione w części pracy poświęconej graficznemu interfejsowi użytkownika. Wybór bibliotek *QT* jako narzędzia do tworzenia *GUI* (*graphic user interface*) był niejako oczywisty ze względu na fakt, iż VLC wykorzystuje jako QT jako jeden z podstawowych interfejsów użytkownika w odtwarzaczu podstawowym. Kluczowa była także opinia Jean-Baptiste Kempfa jednego z autorów i moderatorów biblioteki VLC, a także prezydenta i administratora forum jej poświęconego. Po przeanalizowaniu innych dostępnych możliwości uznano, iż jest to jedyny dostępny produkt dający możliwość swobodnego tworzenia paneli np. do oceny obejrzanego filmu.

QT Creator jedno z narzędzi udostępnianych w ramach zestawu bibliotek QT. Jest to kolejne użyte w pracy środowisko programistyczne pozwalające nie tylko na edycje kodu źródłowego, ale także (w narzędziu QT Designer) na projektowanie graficznego interfejsu użytkowania w okienkowym edytorze za pomocą widżetów z biblioteki QT. Za pomocą dodatkowych narzędzi uic i moc każda klasa korzystająca z sygnałów i slotów QT (reprezentująca element GUI) otrzymuje dodatkowe pliki \*.cpp i \*.ui reprezentujące rozmieszczenia okien, ustawienia grafiki, standardowe zdarzenia i przypisane do nich metody.

VLC-QT darmowa biblioteka autorstwa Tadej Novaka służąca do połączenia bibliotek QT z biblioteką libvlc. Pozwala ona na stworzenie prostego odtwarzacza wraz z dowolnym interfejsem pozwalającym na kontrolowanie odtwarzania. VLC-QT nakrywa metody i klasy upraszczając użycie libvlc w oknach QT. Ze względu na ograniczenia spowodowane przez użycie interfejsu imem do wczytywanie klatek nieskompresowanych sekwencji wideo, użycie VLC-QT zostało ograniczone do użycia widżetu wideo dającego większe możliwości niż standardowe QFrame.

Konfiguracja środowiska

Przed przystąpieniem do pracy z bibliotekami QT należy skonfigurować środowisko programistyczne. Ponieważ projekt został stworzony w środowisku CLion opartym na cmake należało zmodyfikować plik skrypt konfiguracyjny, dodają do linkera ścieżki do uprzednio zainstalowanej biblioteki QT. Dla poprawnego działania narzędzi biblioteki, a także do automatycznej generacji koniecznych do kompilacji plików \*.ui oraz ui\_\*.h należy w skrypcie CMakeLists uruchomić następujące parametry:



Podstawowe połączenie vlc z qt

Odtwarzacz wideo libvlc\_media\_player posiada domyślny interfejs (w zależności od ustawień może być oparty o QT) jednakże interfejsu tego nie można w żaden sposób modyfikować. Dlatego też do modyfikowania interfejsu odtwarzacza należy stworzyć własny widżet wideo. Aby to uczynić w najbardziej podstawowy sposób należy utworzyć klasę dziedziczącą z QWidget, a następnie utworzyć w niej obiekt typu QFrame, który następnie należy połączyć z odtwarzaczem za pomocą metody libvlc\_media\_player\_set\_xwindow(), która za argumenty przyjmuje utworzoną wcześniej instancje libvlc\_media\_player, a także identyfikator okna, który uzyskujemy wołając metodę winId() na obiekcie okna. Następnie można uruchamiać i kończyć wideo tak samo jak robiono to dotychczas bez dodatkowego interfejsu, pamiętając o tym, że nie wolno zniszczyć obiektu okna, do którego przypisany jest odtwarzacz wideo. Standardowy obiekt QFrame możemy modyfikować nadając mu zależne od nas wymiary, czy też umieszczając go w layoucie w wybranym przez siebie miejscu otaczając dowolnymi obiektami dodając np. paski do sterowania głośnością. Kolejnym krokiem powinno być wywołanie w głównej funkcji main() programu utworzenie obiektu klasy QApplication, którego konstruktor przyjmuje standardowe argumenty programu. QApplication to klasa służąca do zarządzania aplikacją okienkową, a także podstawowymi opcjami takimi jak np. rozmiary okna podstawowego czy ustawienie trybu pełnoekranowego. Ważną rolą jest także możliwość przenoszenia argumentów i sygnałów sterujących do dalszych paneli programu. Następnym krokiem jest powołanie do życia obiektu klasy macierzystej za pomocą jej konstruktora. Standardowy konstruktor klasy dziedziczącej z QWidget nie posiada dodatkowych parametrów, jednakże warto go przeciążyć i przekazać mu argumenty programu, aby następnie przekazać je do konstruktora instancji VLC, jeżeli to konieczne. Stworzony widżet należy wyświetlić metodą show(). Ze względu na konieczność otwierania i zamykanie widżetu wideo, a także stworzenia paneli do oceniania i prowadzenia testera zastosowano bardziej zaawansowane podejście.

Zaproponowany Interfejs

Wstęp

Ze względu na silny związek interfejsu użytkownika ze scenariuszami testowymi (końcowa wersja powinna korespondować z opracowanymi scenariuszami pozwalając na ich przeprowadzenie) postanowiono zaprojektować interfejsy dedykowane do konkretnych scenariuszy dopiero po ich określeniu. Ponieważ zdecydowano się na standardowe metody ACR, PC oraz menu z wyborem filmów postanowiono opracować GUI oparte o modułowe panele podmieniane automatycznie w zależności od wybranej konfiguracji testów.

Stworzony Interfejs

Interfejs użytkownika wykonano za pomocą środowiska QT Creator w narzędziu QT Designer pozwalającym na wstawianie komponentów GUI w trybie okienkowym. Ułatwiło to edycje i umiejscowienie elementów dokładnie w tych miejscach w których zamierzano. Całość została ponownie zaimportowana do środowiska CLion głównie ze względu na wygodę i przyzwyczajenie do tego środowiska, dającego większe możliwości debugowania, zwłaszcza przy zaimportowanej bibliotece libvlc.

Projektując interfejs kierowano się jego funkcjonalnością. Ze względu na konieczność użytkowania aplikacji zarówno przez administratora testów jak i przez testera należało stworzyć rozdzielne panele testerki i administracyjny. Użytkownik staje przed wyborem czy zamierza konfigurować testy czy też jest testerem i chce je uruchomić.

[Obrazek z main]

Panel, który widzimy na powyższym zrzucie ekranu jest obiektem typu MainWindow. MainWindow to klasa stworzona jako główne okno aplikacji, klasa dziedziczy z QMainWindow, klasy QT wersji 5 udostępniającej szereg metod to sterowania aplikacją, dziedziczącą z klasy QWidget. Ponieważ obiekt mainWindow jest tworzony w głównej funkcji programu w konstruktorze dodano parametry wysokości i szerokości okna pobrane uprzednio z ustawień ekranu systemowego. Pobieranie rozdzielczości ekranu odbywa się za pomocą dostarczanych w bibliotece QT klas QScreen i QRect. Pierwsza z nich to typ wskaźnika, który za pomocą metody primaryScreen() z klasy QApplication, zostaje ustawiony na obiekt reprezentujący podstawowy ekran w systemie. Następnie za pomocą metody availableGeometry() zostają pobrane jego wysokość i szerokość, następnie przekazane do mainWindow. Konstruktor MainWindow wymusza maksymalizacje okna na ekranie za pomocą metody showFullScreen(), a także tworzy przy pomocy słowa kluczowego new pusty obiekt typu PlayerConfigurationsHandler(), który został omówiony przy okazji obsługi konfiguracji. Na zrzucie ekranu widoczne są dwa przyciski czyli obiekty typu QPushButton. Zostały one umieszczone na środku ekranu przy pomocy linii zakotwiczających przyciski w szablonie, co pozwoliło utrzymać je na swoim miejscu dla różnych rozdzielczości.

Przyciski typu QPushButton to standardowe przyciski z biblioteki QT, poza szerokimi możliwościami związanymi z geometrią, czyli położeniem i wymiarami dostarczają szereg metod pozwalających na interakcje z użytkownikiem. Jedną z tych metod jest metoda on\_pushButton\_clicked(), która jest wyzwalana w momencie naciśnięcia na aktywny obszar przycisku. W przypadku MainWindow zaimplementowano dwie takie metody (do każdego z przycisków osobną). Pierwsza z nich uruchamia panel administracyjny odbywa się to w następujący sposób. Tworzony jest nowy obiekt typu AdminPanel, następnie na ty obiekcie wołane są kolejno metody show(), activateWindow() oraz topLevelWidget(), pozwala to na wyciągnięcie powstałego panelu administracyjnego na wierzch oraz nadanie mu aktywności. Konstruktor AdminPanel przyjmuje jako parametr wskaźnik na wspomniany obiekt typu PlayerConfigurationsHandler, celem nadpisania pustego obiektu, obiektem przechowującym aktualnie wczytaną konfigurację.

[obrazek z admin]

Panel administracyjny widoczny na powyższym zrzucie ekranu posiada dwa przyciski z których jeden służy do zamykania go (poprzez zniszczenie obiektu za pomocą delete), a drugi uruchamia okno systemowe okno dialogowe za pomocą którego można wybrać plik konfiguracyjny. Następuje to za pomocą metody getOpenFileName() która zwraca ścieżkę do wybranego filmu w formacie QString. QString ze względu na brak koniczności użycia jest rzutowany do std::string metodą toStdString(). Następnie ścieżka do pliku konfiguracyjnego jest przekazywana jako argument w konstruktorze tworzonego obiektu PlayerConfigurationsHandler, konfiguracja zostaje wczytana, przekazany z głównego okna wskaźnik zostaje ustawiony na nowo powstały obiekt, a panel administracyjny zostaje zamknięty.

Drugi z przycisków MainWindow jest aktywny tylko w momencie, gdy wczytana jest konfiguracja. Sprawdzanie odbywa się za pomocą wywołania metody CheckConfiguration() z obiektu na który wskazuje playerConfigurationsHandler. Metoda została opisana w podrozdziale dotyczącym konfiguracji. Naciśnięcie na aktywny przycisk tworzy i uruchamia kolejny panel, ustawiając jego geometrie na taką samą jaka została wpisana w MainWindow, skalując go do całości ekranu. Za pomocą szeregu metod okno staje się aktywnym i pełnoekranowym. Nowo powstałe okno jest obiektem typu UserPanel.

UserPanel przyjmuje w konstruktorze parametry dotyczące wielkości ekranu (przekazywane z MainWindow) oraz wskaźnik na obiekt typu PlayerConfigurationsHandler przechowujący aktualną konfigurację. Klasa UserPanel poza przekazanymi parametrami posiada również kilka istotnych pól. Jednym z nich jest wskaźnik mTimer na obiekt typu QTimer, który jest używany w konfiguracji numer 3. Klasa QTimer dostracza liczniki pozwalające na odmierzanie chwil czasowych do odświeżania interfejsów bądź uruchamiania zdarzeń w czasie. Wykorzystanie w praktyce zostanie omówione przy konkretnej konfiguracji. Należy jednak nadmienić, że w konstruktorze klasy UserPanel zostają wybrane konkretne ustawienia obiektu typu QTimer. Typ licznika zostaje ustawiony na singleShot, oznacza to, że przypisane zdarzenie jest uruchamiane tylko raz po upłynięciu określonej chwili. W konfiguracji, w której licznik jest wykorzystywany przypisana zostaje metoda StartPlayback, omówiona w dalszej części tego podrozdziału. Kolejnymi polami są dwie tablice typu bool reprezentujące stany pól wyboru służących do oceny filmów. Ich wykorzystanie zostanie omówione przy opisie systemu oceniania filmów. Klasa UserPanel zawiera również numer aktualnie odtwarzanego filmu, oraz pole typu string przechowujące wyniki testów gotowe do zapisania w pliku tekstowym z wynikami. Klasa zawiera także cztery widżety: startWidget, ratingWidget, ratingWidget\_2 oraz chooseMovieWidget.

[obrazek z start]

Widżet startowy służy podaniu imienia testera mające na celu identyfikacje, czy też odróżnienie od siebie kolejnych osób. Prośba o podanie imienia ma również na celu przywiązanie testera do administratorów, zbudowanie swoistej relacji, daje poczucie większej odpowiedzialności za swoje wyniki niż w przypadku zupełniej aminowości. Po naciśnięciu na przycisk z napisem „Rozpocznij test” wywołana zostaje metoda on\_pushButton\_clicked() rozpoczynająca test. Po wywołaniu metody zostaje sczytana zawartość pola textedit, będący miejscem na wpisanie imienia przez użytkownika. Zawartość zostaje wpisana do pola testsOutputToFileString klasy UserPanel przechowującego zawartość do wpisania w plik wynikowy. Następnie w zależności od numer konfiguracji zostaje wywołana metoda StartPlayback uruchamiająca film, bądź zostaje otwarty kolejny widżet. Pozostałe widżety zostaną omówione w dalszej części pracy.

StartPlayback() to metoda odpowiedzialna za główną funkcjonalność programu czyli odtwarzanie nieskompresowanych sekwencji wideo. W metodzie tworzone są obiekty typu RawDataHandler, FramesHandler oraz ThreadsHandler, oraz OptionsHandler. Funkcjonalność poszczególnych obiektów została opisana w podrozdziale dotyczącym wstrzykiwania klatek wideo. Należy zwrócić uwagę na ustawienia opcji instancji LibVLC. Opcje zostają ustawione tak samo jak w przypadku odtwarzania wideo bez własnego interfejsu jednak parametry takie jak rozdzielczość czy ilość klatek na sekundę zostają wczytane z obiektu typu PlayerConfigurationsHandler do którego UserPanel posiada referencje. W metodzie tworzony jest również obiekt typu Controler, którego zastosowanie również zostało omówione w poprzednich podrozdziałach. Na obiekt ten zostaje ustawiony wskaźnik współdzielony klasy shared\_ptr. Jest to konieczne, aby zachować dostęp do obiektu dla innego wątku. Kolejnym krokiem jest stworzenie kolejnego panelu tym razem typu VideoPanel, który zostanie opisany w kolejnym akapicie. Metoda StartPlayback() posiada również instrukcje warunkowe wynikające z różnych konfiguracji zgodne z przepływem związanego z nią scenariusza testowego. Wywołana zostaje również metoda StopPlayBackThread() na obiekcie threadsHandler. Jest ona odpowiedzialna za zniszczenie okna VideoPanel w momencie zakończenia odtwarzania filmu.

VideoPanel to klasa odpowiedzialna za powiązanie libvlc\_media\_player z oknem odtwarzacza, odbywa się to w konstruktorze tej klasy zgodnie z opisem przedstawionym w przypadku podstawowego połączenia odtwarzacza z QFrame, jednakże zamiast QFrame użyto obiektu klasy VlcWidgetVideo, aby uzależnić go od opcji instancji libvlc\_instance. Konstruktor wraz z metodą przypisania odtwarzacza do widżetu został pokazany na poniższym rysunku.

