Oprogramowanie

1. Informacje ogólne

Kluczowymi elementami pracy było stworzenie oprogramowania umożliwiającego odtwarzanie nieskompresowanych sekwencji wideo oraz przeprowadzenie testów subiektywnych z użyciem autorskiego odtwarzacza.

1. Wybór narzędzi

Wybór narzędzi wiązał się z koniecznością przeglądu dostępnych zasobów ludzkich pod kątem umiejętności tworzenia oprogramowania w danym języku. Kolejną niezbędną kwestią była analiza istniejących rozwiązań w celu znalezienia jak najlepszego narzędzia do rozwiązania problemu.

Odtwarzacz nieskompresowanych sekwencji wideo musi charakteryzować się jak największą wydajnością. Poprzez wydajność rozumiane jest jak najlepsze zarządzanie zasobami w taki sposób, aby na dostępnym sprzęcie komputerowym uzyskać jak najlepsze parametry wyświetlania kolejnych klatek. Zdecydowano się zrezygnować z języków korzystających z maszyn wirtualnych na korzyść takich, które pozwalają na dużą swobodę w zarządzaniu pamięcią. Wybór padł na język C++ ze względu na doświadczenie autorów w pracy z nim, zarówno zawodowe jak i nabyte podczas studiów.

Wybranym systemem operacyjnym został Linux, dystrybucja Xubuntu 16.04.2 posiadająca jądro w wersji 4.4.0-72-generic. Decydującymi czynnikami były łatwość instalowania kolejnych pakietów bibliotek, dostęp do narzędzi konwertujących parametry filmów oraz niskie zużycie zasobów sprzętowych przez biernie działający system.

[https://xubuntu.org/]

Zdecydowano się użyć środowiska programistycznego CLion dostarczanym przez firmę JetBrains, korzystając z licencji studenckiej, która pozwala na użycie IDE (ang. *Integrated Development Environment*) w celach edukacyjnych.

[https://www.jetbrains.com/]

Analiza istniejących odtwarzaczy wideo pozwoliła wyselekcjonować bibliotekę, która dała możliwość darmowego użycia i jak największej możliwości modyfikacji, czyli posiadała licencję *open source*. Ze względu na popularność odtwarzacza VLC w systemach operacyjnych Linux, podjęto decyzję o wykorzystaniu biblioteki z użyciem której powyższy odtwarzacz został stworzony – libVLC.

[http://www.videolan.org/vlc/libvlc.html]

FFmpeg, czyli narzędzie pozwalające na edycje parametrów wzorcowego wideo zostało wybrane ze względu na łatwość użycia, swobodę w wyborze zmienianych aspektów filmu oraz licencję *open source.* Wszystkie użyte w pracy sekwencje wideo zostały wygenerowane przy pomocy FFmpeg z filmów wzorcowych.

[https://ffmpeg.org/]

1. Proces tworzenia

Organizacja pracy w grupie jest zależna od ilości jej członków, natury rozwiązywanego problemu oraz wielu innych czynników np. ograniczeń czasowych lub sposobu prezentacji postępów.

Obecnie w informatyce dąży się do wdrożenia tak zwanych metod zwinnych programowania. Pozwalają one zminimalizować straty w przypadku, gdy część wymogów klienta ulegnie zmianie. Wynika to z faktu, że kod źródłowy programu dostarczany jest iteracyjnie w jak najmniejszych częściach. Podejście to pozwala również zmaksymalizować czas faktycznej pracy każdego z pracowników, ponieważ eliminowane są sytuacje, w których pojawia się konieczność oczekiwania na wartość dostarczaną przez innych pracowników.

Oprogramowanie przedstawione w pracy tworzone było w zespole dwuosobowym. Jednym z pierwszych kroków podczas realizacji pracy magisterskiej było zdefiniowanie zasad współpracy między opiekunem, uczelnią a osobami odpowiedzialnymi za nią samą. W związku z koniecznością dotrzymania terminów odgórnie narzuconych przez uczelnie zdecydowano się wyznaczyć kilka terminów wraz z zakresem funkcjonalności, które na dany termin miały by być dostarczone. Pozwoliło to zsynchronizować wiedzę o projekcie między studentami, a prowadzącym oraz na skupienie się na wyznaczonych celach. Dopiero po akceptacji postępów przez opiekuna następowało przejście do kolejnego etapu. Na każdym ze spotkań projektowych przeprowadzano pokaz aktualnej wersji, na którym omawiano aktualny sposób działania, najnowsze zmiany w funkcjonowaniu programu oraz wyszukiwano ewentualne niedociągnięcia.

W czasie realizacji pracy magisterskiej obaj członkowie zespołu pracowali zawodowo oraz uczęszczali na zajęcia prowadzone na uczelni. W związku z tym proces tworzenia pracy magisterskiej musiał być dostosowany do ich obłożenia czasowego. Naturalnym wyborem było więc działanie w metodyce *Kanban* – skupiającej się na produkcie, bez dodatkowych, niepotrzebnych nakładów pracy oraz minimalizacji bezczynności. Stworzono prostą *tablicę kanbanową* czyli złożenie danych o postępie prac nad projektem przy podziale na poszczególne jego zadania. Zaletą takiego rozwiązania jest wiedza o przewidywanym terminie zakończenia prac oraz dowolność w wyborze czasu, w którym są one realizowane (oprócz terminów końcowych). Takie podejście pozwoliło powiązać pracę zawodową z rozwojem naukowym.

System kontroli wersji zastosowany w projekcie pozwolił na równoczesną pracę nad wieloma funkcjonalnościami programu, wprowadził historię zmian oraz pozwolił na bezproblemową ich synchronizację między członkami zespołu. Wykorzystano popularną platformę GitHub, która dostarcza gotowego rozwiązania wraz z serwerem do przechowywania danych.

[https://github.com/Kondix/UHDPlayer]

1. Analiza systemu

Analiza systemu została przeprowadzona przed implementacją rozwiązania. Podczas tej fazy stworzono wymagania, które będą stanowiły o tym kiedy projekt można uznać za zakończony. Ich definicja była konieczna, aby ukierunkować tok prac.

Lista wymagań projektowych:

* Odtwarzanie nieskompresowanych sekwencji wideo o wybranych parametrach
* Utrzymanie jakości filmu przez cały jego czas trwania
* Możliwość przeprowadzenia różnych scenariuszy testowych
* Możliwość wygenerowania filmów o zadanych parametrach z filmu źródłowego
* Automatyczna prezentacja filmów osobie testowanej
* Zbieranie danych o ocenie filmu osoby testowanej

Wszystkie powyższe wymagania są przedstawione z punktu widzenia nietechnicznego, co pozwala na ich zrozumienie również osobie, która nie posiada wiedzy specjalistycznej.

Testy subiektywne przeprowadzane są z użyciem nieskompresowanych sekwencji wideo, w celu wyeliminowania wpływu procesu dekodowania na otrzymany obraz, a co za tym idzie otrzymania lepszej jakości wyników samego testu. Różne implementacje tego samego sposobu dekodowania mogły by spowodować, że przeprowadzenie teoretycznie tego samego testu na dwóch różnych, niezależnych maszynach, otrzyma wyniki dla rożnych obrazów widzianych przez osoby testujące.

Odtwarzacz takich sekwencji oraz maszyna, na której jest uruchomiony powinny być skonfigurowane tak, aby film w zadanej jakości odtwarzał się płynnie, wyświetlane były wszystkie klatki oraz ich kolejność i jakość została zachowana. Bardzo ważnym elementem analizy było ustalenie zbioru jakości, które odtwarzacz będzie mógł odtworzyć. Ograniczeniem w tej kwestii pozostaje dostępny sprzęt elektroniczny, budżet na zakup nowych części i rozwiązań oraz sama implementacja odtwarzacza. O tym czy zapewniona została odpowiednia jakość filmów decydował zestaw testów wydajnościowych, o których więcej pojawi się w dalszej części pracy.

Przeprowadzenie testów powinno odbywać się z jak najmniejszym udziałem człowieka, tj. po uprzednim przygotowaniu scenariuszy kolejne uruchomienia testu powinny być zautomatyzowane, a sama osoba testująca powinna być „prowadzona za rękę” przez cała długość jego trwania. Tester musi mieć dodatkowo możliwość interakcji z systemem w celu podawana kolejnych odpowiedzi na zadane pytania.

Kluczową dla pracy magisterskiej jest kwestia porównania między sobą wybranych scenariuszy testowych. Dlatego wymagana jest również możliwość ich przeprowadzenia, a co za tym idzie skonfigurowania. Konfiguracja i struktura plików nie powinna być dostępna dla testera ze względu na przykład na ich nazwy, które podczas generacji przez skrypt zawierają sugestie, w jakiej jakości zapisany jest dany film.

Jakości poszczególnych filmów testowych powinny być definiowane przez administratora lub osobę za to odpowiedzialną, a nie być narzucone odgórnie przez skrypt, ze względu na różnice pomiędzy koncepcjami testowymi. Niektóre scenariusze mogą zakładać użycie materiałów o nieznaczne zmienionych parametrach, inne korzystać z takich, których parametry różnią się znacznie. Generacja filmów powinna odbywać się automatycznie, korzystając z pliku źródłowego oraz informacji o pożądanej jakości wynikowego filmu. Należy zaznaczyć, że skrypt generujący został stworzony tylko na potrzeby wewnętrzne projektu w celu zaspokojenia zapotrzebowania na dane wejściowe systemu. Użytkownik końcowy powinien zapewniać jakość sekwencji wideo we własnym zakresie.

1. Wybrane jakości wideo
2. Szczegóły implementacyjne

Kolejnym krokiem po szczegółowej analizie systemu było zaplanowanie architektury kodu. Naturalnym początkiem było wydzielenie części odpowiedzialnej za przetwarzanie i wyświetlanie nieskompresowanych sekwencji wideo. Po ukończeniu odtwarzacza ruszyły prace, mające na celu stworzenie warstwy widocznej przez użytkownika, wraz z funkcją przeprowadzenia wybranych scenariuszy testowych.

Biblioteka *libVLC* zapewnia dostęp do wielu gotowych metod obsługi materiału wideo, nie posiada jednak bezpośrednich metod do odtwarzania nieskompresowanych sekwencji wideo. W przypadku sekwencji skompresowanych jednym ze standardowych sposobów odtwarzania jest wczytywanie nie całych klatek, a tylko zmian zachodzących między kolejnymi dwoma poprzez dekompresje. Takie podejście pozwala znacząco ograniczyć rozmiar wczytywanych do pamięci danych. Mniejsza ilość danych pozwala na zwiększenie maksymalnej jakości filmu odtwarzanego na tym samym komputerze względem programu ładującego każdą klatkę od nowa.

W przypadku inkrementacyjnego ładowania zmian w jednym z ostatnich kroków procesu otrzymywana jest klatka reprezentowana w ten sam sposób, co nieskompresowana. Dzięki tej obserwacji, możliwe jest wstrzyknięcie wczytanej nieskompresowanej klatki do standardowego procesu odtwarzania filmu i jego wyświetlenie z użyciem biblioteki *libVLC*. Szczegółowy opis implementacji tego sposobu zostanie przedstawiony w dalszej części rozdziału.