`

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie oraz nie korzystałem ze źródeł innych niż wymienione w pracy.

…….

(czytelny podpis)

Spis treści

1. Wstęp

Już od najdawniejszych czasów ludzie interesowali się różnego rodzaju zmaganiami sportowymi. Organizowano rozmaite zawody sportowe, igrzyska, turnieje a na początku XX wieku powstały cyklicznie rozgrywki ligowe. Do najpopularniejszych rozgrywek można zaliczyć: piłkę nożną, koszykówkę, hokej, baseball czy piłkę siatkową. Kibice obok ekscytacji związanej z oglądaniem spotkań sportowych zaczęli interesować się również statystykami prowadzonymi podczas rozgrywanych spotkań. Zaczęto odnotowywać liczbę zdobytych punktów, bramek czy asyst na mecz przez danego zawodnika. Wylicza się różnego rodzaju średnie, które pozwalają wybrać najlepszego zawodnika ligi, stworzenie najlepszej drużyny sezonu czy ustalenie która drużyna ma najlepsza obronę lub atak. Zaczęły też powstawać rożnego rodzaju gry planszowe czy karciane, które poprzez opis parametrów danego zawodnika czy drużyny umożliwiały rozegrać kilku osobom rozgrywki zbliżonej do tej rzeczywistej.

Pod koniec XX wieku rozwój komputerów i oprogramowania pozwolił na tworzenie aplikacji, które miały na celu symulować dane rozgrywki. Powstały rożnego rodzaju gry symulujące rozgrywki piłkarskie czy innych sportów drużynowych. Odkąd pamiętam zawsze interesowałem się piłką nożną. Oglądałem wszystkie turnieje mistrzostw świata i Europy, śledziłem różne ligi europejskie i puchary. Jednocześnie byłem miłośnikiem symulatorów rozgrywek piłkarskich tak zwanych futbol menadżerów. Grałem w nie od czasów Commodore c-64 po dzień dzisiejszy. Zawsze zastanawiałem się jak działają i jak są skonstruowane. Tak narodziła się pasja do programowania, która w połączeniu z pasją do piłki nożnej pozwoliła mi stworzyć symulator rozgrywek piłkarskich.

2. Cel Pracy

Celem niniejszej pracy jest stworzenie aplikacji, która umożliwiła by symulowanie rozgrywek ligowych w tym przypadku rozgrywek piłki nożnej.

3. Wybór technologii

3.1 Przegląd technologii

W tym rozdziale skupię się na przeglądzie dostępnych technologii na rynku, które umożliwiają wytwarzanie oprogramowania. Przedstawię podział języków ze względu na kryterium przydatności oraz uargumentuję dlaczego jedna grupa jest lepsza od innej do wykonania tego projektu. Dodatkowo postaram się opisać biblioteki graficzne dedykowane do poszczególnych języków, które umożliwiają w krótkim czasie wytwarzanie aplikacji okienkowych.

Języki programowania możemy podzielić na kilka grup. Najpopularniejsze to:

- funkcyjne
- skryptowe
- proceduralne
- obiektowe

Przedstawicielami języków funkcyjnych jest Haskell, Lisp, Erlang czy F#. Programy napisane w językach funkcyjnych składają się jedynie z funkcji. Główny program jest funkcją, która przyjmuje argumenty i zwraca obliczoną wartość. Może składać się on z innych funkcji, które mogą wywoływać jeszcze inne funkcje. Funkcja może przyjmować jako parametr funkcję jak również zwracać inna funkcję. W programowaniu funkcyjnym zamiast zmiennych występują stałe a zamiast pętli stosowana jest rekurencję. Najważniejszą cechą tych języków jest to, iż nie pozwalają na żadne efekty uboczne. W czystym programowaniu funkcyjnym, raz zdefiniowana funkcja dla tych samych danych wejściowych zawsze zwróci ten sam wynik. Języki funkcyjne zostały zaprojektowane z myślą o zastosowaniach współbieżnych (Erlang) oraz pod kątem tworzenia systemów rozproszonych wymagających długotrwałej pracy oraz odporności na awarie. Jak również często stosowane są do tworzenia sztucznej inteligencji (Lisp).

Drugą grupą są języki skryptowe. Jak sama nazwa wskazuje są to języki, które obsługują skrypty. Najpopularniejsze to JavaScript, Pyton, PHP, Ruby on Rails, Perl czy Node.JS. Grupa ta odróżnia się od pozostałych tym, że kod napisany w tych językach może być tworzony nawet w notatniku i nie musi być kompilowany do pliku wykonywalnego - zrozumiałego dla komputera. Proces kompilacji w językach skryptowych odbywa się za każdym razem w momencie uruchomienia programu. Zaletą takiego programowania jest szybsze pisanie aplikacji a programy tworzone w tych technologiach mają zwykle małą objętość. Języki skryptowe często stosuje się do automatyzowania różnego rodzaju zadań najczęściej administracyjnych. Skrypty wykorzystywane są również do sterownia przebiegiem gry, kontrolują wtedy fabułę, dialogi czy sterują zachowaniem wirtualnych postaci. Języki takie jak PHP czy JavaScript wykorzystywane są w technologiach internetowych (HTML) do dynamicznej interakcji z użytkownikiem poprzez reagowanie na zdarzenia, zapewnieniu walidacji danych z formularzy czy dynamicznym manipulowaniu obiektami na stronach internetowych.

Kolejną grupą, której przedstawicielem jest język C, są języki proceduralne. Głównym założeniem tych technologii jest to, że program główny składa się z wielu procedur (podprogramów) lub funkcji realizujących określone zadania które mogą być wywoływane wile razy. Zadaniem procedur jest modyfikacja zmiennych globalnych, albo też modyfikacja wartości lub struktury przekazanej w parametrze, natomiast funkcja oprócz modyfikacji parametrów może je również zwracać. Możliwe jest również zagnieżdżanie odwołań do procedur dzięki czemu można korzystać z rekurencji. C jest językiem strukturalnym i nie wspiera programowania obiektowego, jednakże programowanie obiektowe jest w nim możliwe. Dodatkowo język C jest wśród języków wysokopoziomowych „najbliżej" maszyny co czyni go bardzo wygodnym narzędziem do tworzenia oprogramowania dla systemów czy mikrokontrolerów. Jeżyk C jest bardzo wydajny i lekki przez co chętnie jest wykorzystywany do programowania systemów wbudowanych stosowanych w urządzeniach domowych takich jak pralki, lodówki poprzez zegarki a kończąc na elementach samochodów takich jak: wyświetlacze, czujniki czy systemy zapewniające bezpieczeństwo. C został również do napisania jąder takich systemów jak Linux, Windows, Android czy biblioteki graficznej OpenGL.

Ostatnią grupą stanowią języki obiektowe. Do najpopularniejszych technologii z tej grupy należą: C++, Java, C#. Najważniejszym pojęciem z wiązanym z programowaniem obiektowym jest klasa, która stanowi szablon na podstawie którego tworzone są obiekty (instancje klas). Można powiedzieć, że klasa jest to typ złożony zdefiniowany przez użytkownika. Dodatkowo klasa w odróżnieniu od struktury posiada metody(funkcje) umożliwiające wpływanie na zachowanie obiektu. Podstawowymi założeniami paradygmatu obiektowego są cztery cechy:

- Abstrakcja - polega na ukrywaniu lub pomijaniu mało istotnych informacji a skupieniu się na wydobyciu informacji, które są niezmienne i wspólne dla pewnej grupy obiektów.
- Dziedziczenie - Podstawowym zastosowaniem dziedziczenia jest ponowne wykorzystanie kodu. Jeśli dwie klasy wykonują podobne zadania, możemy utworzyć dla nich wspólną klasę bazową, do której przeniesiemy identyczne metody oraz atrybuty.
- Enkapsulacja – czyli hermetyzacja polega na ukrywaniu implementacji obiektu przed użytkownikiem i zapewnia , że obiekt nie może zmieniać w nieoczekiwany sposób stanu wewnętrznego innych obiektów.
- Polimorfizm – uwolnienie się od typów czyli traktowanie różnych danych w jednolity sposób. Rozróżniamy polimorfizm statyczny (czasu kompilacji) i polimorfizm dynamiczny (czasu wykonania)

Zastosowanie języków obiektowych jest bardzo szerokie. Wykorzystuje się je do pisania gier, aplikacji desktopowych, internetowych jak również mobilnych czy nawet systemów wbudowanych (C++).

Odnosząc się do powyższej charakterystyki technologii językowych stwierdzam, że najodpowiedniejszą grupą do stworzenia symulatora rzeczywistych rozgrywek ligowych w oparciu o graficzny interfejs użytkownika jest grupa języków obiektowych. Języki funkcyjne jak wcześniej napisałem mają główne zastosowanie w systemach rozproszonych czy projektowaniu sztucznej inteligencji. Języki skryptowe są w ostatnich czasach bardzo często wykorzystywane przy wytwarzaniu aplikacji internetowych. Języki proceduralne nadają się do

programowania tego typu aplikacji, jednakże przewagą języków obiektowych nad językami proceduralnymi jest fakt, że taki sposób programowania jest bardziej zgodny rzeczywistością a ludzki mózg jest lepiej przystosowany do takiego podejścia podczas przetwarzania informacji. Podstawowe założenia paradygmatu obiektowego takie jak abstrakcja czy enkapsulacja przyczyniają się do zwiększenia czytelności kodu natomiast dziedziczenie i polimorfizm są zwykle używane w celu redukcji zbędnego kodu. Dodatkowo w ostatnich latach języki obiektowe zostały wyposażone w potężne darmowe biblioteki graficzne , które ułatwiają i przyśpieszają znacząco tworzenie kodu.

3.2 C++ i Qt

Język C++ to potężne narzędzie łączące w sobie trzy różne kategorie programowania: programowanie proceduralne charakterystyczne dla języka C, obiektowe wyrażone przez dodanie klas oraz programowanie uogólnione znajdujące wyraz w szablonach języka C++. Charakteryzuje się wysoką wydajnością kodu wynikowego, bezpośrednim dostępem do zasobów sprzętowych i funkcji systemowych, łatwością tworzenia i korzystania z bibliotek (napisanych w C++, C lub innych językach), niezależnością od konkretnej platformy sprzętowej lub systemowej (co gwarantuje wysoką przenośność kodów źródłowych) oraz niewielkim środowiskiem uruchomieniowym.

C++ został zaprojektowany przez Bjarne Stroustrupa jako rozszerzenie języka C o obiektowe mechanizmy abstrakcji danych i silną statyczną kontrolę typów. Zachowanie zgodności z językiem C na poziomie kodu źródłowego pozostaje jednym z podstawowych celów projektowych kolejnych standardów języka.W latach 90. XX wieku język C++ zdobył pozycję jednego z najpopularniejszych języków programowania ogólnego przeznaczenia.

Qt jest to wieloplatformowy faramework służący do tworzenia aplikacji desktopowych, wbudowanych (embedded) i mobilnych które, mogą być uruchamiane na wszystkich podstawowych systemach (Windows, Linux, iOs, Android, BlackBerry i inne). Qt używa standardów C++ z rozszeżeniami takimi jak sygnały i sloty. Posiada również własny zestaw szablonów kontenerów (QVector, QList, QMap i inne). Qt wyposażony jest w Qt Creator narzędzie do szybkiego tworzenia graficznego interfejsu urzytkownika (GUI)

Ponadto Qt posiada bardzo dobrze napisaną dokumentację, oraz liczne przykłady i turtoriale ułatwiające pracę z tym środowiskiem. Biblioteki Qt, oprócz obsługi interfejsu użytkownika, zawierają także niezależne od platformy systemowej moduły obsługi procesów, plików, sieci, grafiki trójwymiarowej (OpenGL), baz danych (SQL), języka XML, lokalizacji, wielowątkowości, zaawansowanej obsługi napisów oraz wtyczek.

Bardzo łatwy sposób tworzenia GUI przy pomocy kreatora, rozbudowana dokumentacja z licznymi przykładami, jak również bardzo pomocny edytor powodują, że jest to najbardziej intuicyjne środowisko programistyczne z jakim do tej pory się spotkałem.

3.3 Java i Swing

3.4 C# WPF