Akademia Górniczo-Hutnicza
Wydział Fizyki i Informatyki Stosowanej

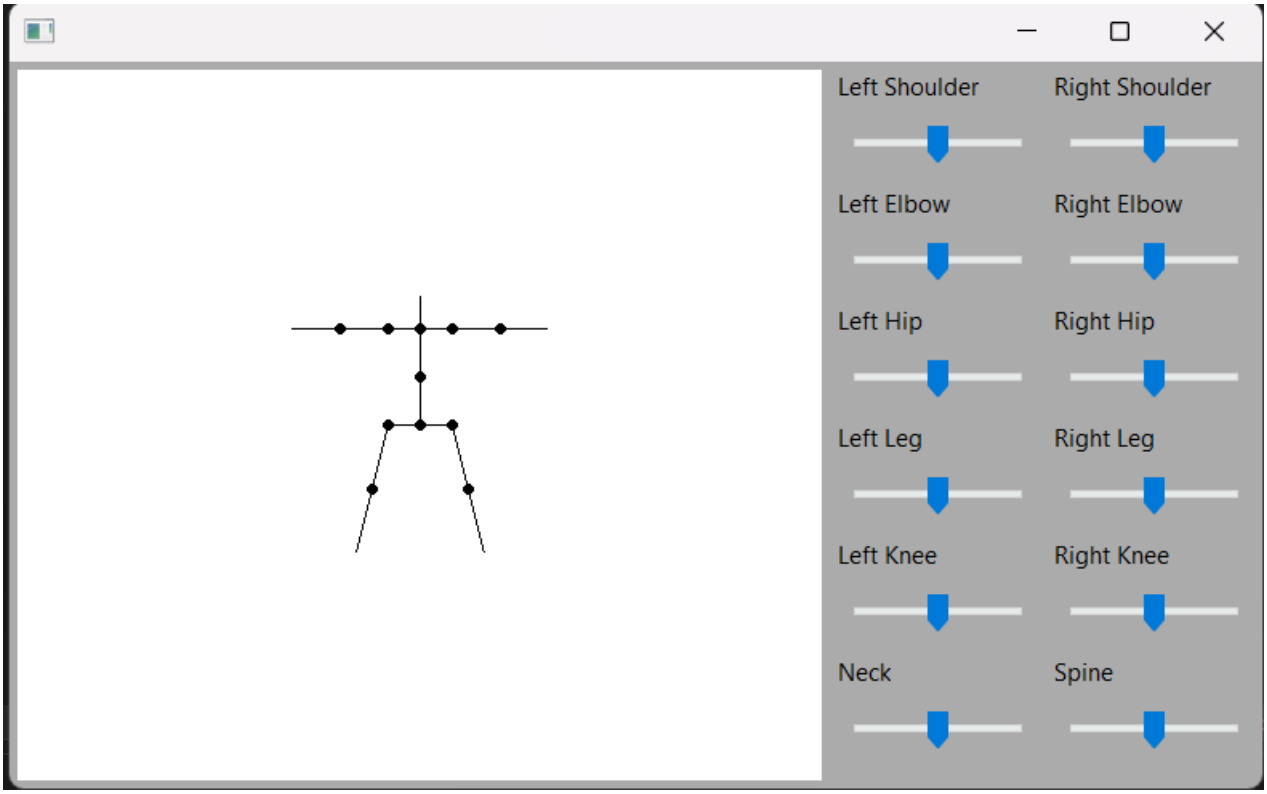
Porjekt nr 36

Animacja szkieletu

Jakub Pietrzyk, Michał Piwowarczyk, Mikołaj Słowikowski

1 Opis projektu

Celem projektu było napisanie programu, który pozwala na poruszanie szkieletu człowieka dzięki suwakom określającym kąty pomiędzy poszczególnymi segmentami szkieletu.



Rysunek 1: Interfejs użytkownika

2 Założenia wstępne przyjęte w realizacji projektu

W naszym zadaniu przyjęliśmy, że nasz szkielet może wykonywać tylko ruchy w płaszczyźnie 2D. Okno aplikacji podzielone jest na dwie części. Po lewej stronie widać nasz model rysowany na `wxPanel`. Po prawej stronie znajduje się 12 suwaków, które odpowiadają za poruszanie szkieletem. Dodatkowo, aby model był jak najbardziej realistyczny ograniczyliśmy kąt zginania kończyn, żeby uniemożliwić niemożliwe ruchy.

3 Analiza projektu

3.1 Specyfikacja danych wejściowych

Dane wejściowe są z góry narzucone przez nasz program. Podczas tworzenia obiektu `GUIMainFrame`, zostaje utworzony obiekt `Frame`, który jest odpowiedzialny za przechowywanie zmiennych określających położenie segmentów szkieletu. Do jego stworzenia zostają przekazane punkty początkowe dla każdego węzła, a konstruktor przechowuje je w ciele klasy. Punkty te mogą zostać przesunięte o odpowiedni kąt, przy użyciu odpowiednich suwaków umieszczonych z prawej strony aplikacji,

3.2 opis oczekiwanych danych wyjściowych

Dane wyjściowe są prezentowane z lewej strony aplikacji w postaci szkieletu rysowanego na `wxPanel`. Do prezentacji wyników służy funkcja `Draw()` w klasie `GUIMainFrame`, która na podstawie pozycji węzłów rysuje odpowiednie węzły i segmenty.

3.3 zdefiniowanie struktury danych

3.3.1 Klasa Point

Do przechowywania informacji o położeniu punktu zdefiniowaliśmy własną klasę `Point` której składnikami były dwie zmienne typu zmiennoprzecinkowego. Zdecydowaliśmy się zastosować taką strukturę aby nie tracić dokładności jak gdyby punkty były przechowywane jako typ int jak ma to miejsce w klasie `wxPoint`. Dopiero podczas rysowania następowało rzutowanie klasy `Point` na `wxPoint`.

3.3.2 Klasa Frame

Do opisu szkieletu zdefiniowaliśmy klasę `Frame`. klasa ta składa się z obiektów klasy `Point` które opisują położenie każdego węzła szkieletu oraz ze zmiennych typu double opisujących o jaki kąt nastąpiła rotacja w danym węźle. Klasa ta dostarcza również zestaw getterów oraz setterów do poszczególnych składników. Do obsługi rotacji wewnątrz klasy zostały zdefiniowane metody dla poszczególnych węzłów służące do wykonywania rotacji. Każdy węzeł ma swoją metodę. W klasie znajduję się również uniwersalna funkcja odpowiadająca za wykonanie rotacji jednego punktu względem drugiego o zadany kąt.

3.4 specyfikacja interfejsu użytkownika

Projekt przedstawiliśmy w aplikacji okienkowej, która była podzielona na dwa głowne segmenty. Z lewej strony przedstawiony jest aktualna postać szkieletu. Natomiast z prawej strony widnieje zestaw suwaków odpowiedzialnych za poruszanie odpowiednimi segmentami modelu. Ruch suwakiem powoduje wywołanie odpowiedniego zdarzenia, które zostaje obsłużone poprzez wykonanie odpowiedniej funkcji odpowiedzialnej za ruch odpowiedzialnym segmentem. Przekazana zostaje ustawiona wartość z suwaka, która następnie w funkcji `rotate(...)` zostaje zamieniona na kąt. Następnie zostaje wywołana funkcja `rotateNode` odpowiedzialna o zmianę położenia segmentu o przekazany wcześniej kąt. Na koniec wywołana jest funkcja `Draw()`, która prezentuje nam wynik zmian po lewej stronie okienka.

3.5 wyodrębnienie i zdefiniowanie zadań

W projekcie można wyodrębnić następujące bloki zadań:

1. Utworzenie schematu i planu aplikacji
2. Stworzenie interfejsu użytkownika przy pomocy wxFormBuilder
3. Opracowanie klasy Point
4. Opracowanie algorytmu rotacji punktu
5. Utworzenie i połączenie wydarzeń zmiany suwaków z funkcjami rysującymi dane segmenty
6. Debugowanie funckji

7. Testy programu
8. Opracowanie Dokumentacji

3.6 decyzja o wyborze narzędzi programistycznych

Nasz projekt został stowrzony przy wykorzystaniu języka C++ oraz biblioteki wxWidgets. Wybralismy tą bibliotekę ze względu na doświadczenie nabyte podczas wykonywaniu poprzednich projektów na zajęcia labolatoryjne.

Również skorzystaliśmy z narzędzia do projektowania GUI - wxFormBuilder. Ułatwiło ono znacząco pracę nad interfejsem użytkownika poprzez wygenerowanie plików `GUI` i `GUIMainFrame`.

Wybór środowiska programistycznego padł na Microsoft - Visual Studio, ze względu na zdobyte doświadczenie.

Dodatkowo korzystaliśmy z systemu kontroli wersji - Git. Umieścilismy nasz projekt na platformie https://github.com/, co znacząco ułatwiło pracę w grupie nad kodem.

4 Podział pracy i analiza czasowa

Zadania	Czas [h]
Utworzenie schematu i planu aplikacji	1,5
Stworzenie interfejsu użytkownika przy pomocy wxFormBuilder	2
Opracowanie klasy Point i Frame	2,5
Opracowanie algorytmu rotacji punktu	1
Utworzenie i połączenie wydarzeń zmiany suwaków z funkcjami rysującymi dane segmenty	1,5
Debugowanie funckji	1
Testy programu	2
Opracowanie Dokumentacji	4

Każda członek projektu dokonał wyboru interesującego go zagadnienia. Praca została podzielona na dwie główne części odpowiedzialne za UI oraz back-end. Osoba która wykonała swoje zadanie w krótszym czasie niż to przewidziała przechodziła do pozostałych zadań lub pomocy pozostałej części.

5 Opracowanie i opis niezbędnych algorytmów

5.1 Algorytm rotacji punktu względem innego punktu

Do wykonywania rotacji wokół punktu o zadany kąt posłużyliśmy się algorytmem dla którego danym wejściowymi były dwa punkty oraz kąt. Jeden z tych punktów był środkiem obrotu. Do opracowania tego algorytmu posłużyliśmy się obrotem wektorów. Gdzie współrzędne po obrocie o kąt α przyjmują takie wartości:

$$x' = xcos\alpha - ysin\alpha \tag{1}$$

$$y' = xsin\alpha + ycos\alpha \tag{2}$$

6 Kodowanie

6.1 Klasa Point

Klasa reprezentująca punkt przechowująca dwie współrzędne typu double x i y. Stworzyliśmy ją, ponieważ klasa wxPoint przechowywała zmienne typu int co powodowało niedokładne działanie programu. Klasa Point była rzutowana na klasę wxPoint podczas rysowania. Klasa Point składała się również z konstruktora domyślnego, konstruktora dwuargumentowego i operatora konwersji z Point(double, double) do wxPoint(int,int).

6.2 Klasa Frame

Klasa wykorzystywana do opisu szkieletu. Posiada składniki typu Point opisujące konkretne punkty szkieletu takie jak dłoń, łokieć, bark itp. oraz składniki typu double odpowiadające kątom w stawach. Klasa posiada również konstruktor domyślny, konstruktor zawierający wszystkie punkty, destruktor, gettery do każdego punktu, settery każdego punktu oraz funkcje rotujące konkretne węzły. W funkcji rotateNode(Point, Point, double) na początku przypisujemy zmiennym double x i double y różnice w położeniu odpowiednio na osi x i y między podanymi punktami. Następnie do zmiennych x1 i y1 przypisujemy odpowiednio do x1 różnice między wartością x pomnożoną przez cos kąta a wartością y pomnożoną przez sin tego kąta, a zmiennej y1 sumę iloczynu zmniennej x oraz sin kąta i iloczynu zmiennej y i cos tego kąta. Na koniec przypisujemy współrzędne pierwszemu podanemu punktowi równe sumie wartości sumy zmiennym x1/y1 i wartości współrzędnej x/y drugiego podanego punktu.

7 Testowanie

Aplikacja została przetestowana przez niezależną komisję złożoną z nas es. Przeprowadziliśmy szereg testów manualnych, które miały na celu wskazanie potencjalnych defektów w naszym programie. Pomimo, wszelakich prób ”zepsucia” programu, nie udało nam się wywołać, żadnego błędu. Wnioskujemy, że aplikacja działa poprawnie i zgodnie z założeniami.

8 Wdrożenie, raport i wnioski

Udało nam się uzyskać działającą aplikację zgodnie z zaplanowanym schematem. Niezależnie od zmieniania danych poprzez ruszanie suwakami, szkielet rusza się poprawnie z zachowaniem naturalnych ruchów, które może wykonać człowiek. Możemy zdecydowanie stwierdzić, że udało się wykonać wszystko, co zostało zaplanowane.

Program można rozwinąć o ruch w płaszczyźnie 3D, co umożliwiłoby przedstawienie szeregu animacji szkieletu, na przykład przedstawienie wykonywaniu przysiadów. Jednakże, taki program byłby dużo bardziej skomplikowany i ciężki do realizacji w wymaganym terminie.