JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 47–52
Received: 8 October 2023
Accepted: 18 November 2023

A comparative analysis of transitions generated using the Unity game development platform

Analiza porównawcza przejść generowanych przy użyciu platformy do tworzenia gier Unity

Marek Tabiszewski*

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
This paper conducts a comparative analysis of transitions generated using the Unity engine. It selects fifteen animations featuring a humanoid character, introduces breaks in marker trajectories, and fills them with transitions generated by the game engine's animator. These transitions are then compared with the unmodified original character animation. The study compares animations by calculating the average deviation in bone rotation and position between the original and generated motion throughout the animation. The results show that the Unity engine excels in generating transitions for slow animations involving the lower body limbs, with the largest errors occurring in the bones at the extremities of the limbs.

Keywords: Unity; animation; character motion; animation quality

Streszczenie
W artykule przeprowadzono analizę porównawczą przejść generowanych przy użyciu silnika Unity. Do badań wyselekcjonowano piętnaście animacji postaci humanoidalnej, w których wprowadzono przerwy w trajektoriach markerów tak, aby można było zapełnić je przejściami wygenerowanymi przez animator użytego silnika gier, a następnie porównać przejścia z oryginalnym ruchem postaci pochodzącym z niezmodyfikowanej animacji. Animacje porównano obliczając średnie odchylenie rotacji oraz pozycji każdej z kości pomiędzy ruchem oryginalnym a wygenerowanym na przestrzeni całej animacji. Na podstawie otrzymanych wyników stwierdzono, że silnik Unity lepiej generuje przejścia pomiędzy animacjami, które są powolne i angażują dolne kończyny ciała oraz że największe błędy generują kości na końcach kończyn.

Słowa kluczowe: Unity; animacje; ruch postaci; jakość animacji

*Corresponding author
Email address: **Marek.tabiszewski@pollub.edu.pl** (M.Tabiszewski)

1. Wstęp

Naturalną tendencją gier wideo na przestrzeni lat jest ciągle polepszająca się ich jakość audiowizualna. Chcąc osiągnąć efekt jak najbardziej zbliżony do rzeczywistości stosuje się coraz bardziej szczegółowe modele, tekstury o coraz wyższej rozdzielczości oraz nowe, lepsze metody oświetlenia w pełni wykorzystując możliwości powszechnie dostępnych podzespołów komputerowych czy konsol [1]. Niemniej ważnym aspektem realizmu w grach jest jakość przedstawianych w nich animacji postaci. Istnieją prace [2,3] świadczące o tym, że wraz ze wzrostem jakości i szczegółowości przedstawianych modeli humadoidalnych, rośnie wrażliwość odbiorcy na ewentualne błędy bądź nienaturalność ich ruchu.

Współcześnie do jak najwierniejszego przedstawienia animacji humanoidalnych stosuje się nagrania w realizowane technologii motion capture. Ograniczenia technologiczne takie jak ograniczone miejsce i długość ujęcia, a także specyfika gier wideo, która często wymaga dynamicznego łączenia różnych ruchów postaci np. w reakcji na sterowanie gracza lub inne czynniki zewnętrzne, sprawia, że animacje tego typu muszą być ze sobą łączone aby otrzymać jeden płynny ruch. Takie połączenia, będące przejściem z jednej animacji w drugą są punktami narażonymi na wystąpienie nienaturalnych ruchów. Aby się przed nimi uchronić korzysta się z różnych algorytmów dobierających animacje i interpolujących pomiędzy nimi.

Zagadnienie generowania przejść pomiędzy nagraniami ruchu jest poruszane od końca lat dziewięćdziesiątych. Prace [4-8] opisują nowatorskie, jak na tamte czasy podejścia, które dawały zadowalające rezultaty, jednakże obliczenia potrzebne do wygenerowania ruchu trwały tak długo, że uniemożliwiały generację przejść w czasie rzeczywistym. Jednakże już w pracach [9-11] postęp technologiczny pozwalał na dynamiczne generowanie przejść pomiędzy nagraniami ruchu dla postaci sterowanej przez użytkownika w czasie rzeczywistym. Najnowszym podejściem do generowania przejść jest wykorzystanie sieci neuronowych. W pracach [12-14] osiągnięto niespotykane dotąd rezultaty przy użyciu dużo mniejszych zasobów obliczeniowych.

W kontekście przytoczonych przykładów podejść do problematyki generowania przejść pomiędzy nagraniami ruchu, środowisko Unity oferuje stosunkowo prymi-

tywne rozwiązanie [15], jednakże prostota jego użycia oraz możliwość ograniczonej konfiguracji czyni go dobrym narzędziem do nieskomplikowanych zastosowań. Wyniki analizy ruchów wygenerowanych za pomocą tego systemu przedstawione w tym artykule mogą być pomocne w wyodrębnieniu animacji, które mogą być szczególnie narażone na błędy.

2. Eksperyment

Jedynym sposobem na obiektywne zbadanie jakości generowanego przejścia niezależnego od czynników zależnych od użytych animacji takich jak liczba klatek na sekundę lub charakterystyka ruchu jest generowanie przejścia pomiędzy dwiema pozami i porównywanie powstałego ruchu z ruchem pomiędzy dwiema takimi samymi pozami ale pochodzącym z animacji.

2.1 Użyte oprogramowanie

W celu przeprowadzenia eksperymentu użyto następującego oprogramowania:

- Blender 2.92 - do przygotowania animacji
- Unity 2020.3.4f1 - do przeprowadzenia pomiarów
- JetBrains Rider 2021.2 - środowisko programistyczne
- LibreOffice Calc - analiza wyników

2.2 Przygotowanie materiału badawczego

Animacje użyte do przeprowadzenia badań pobrano z bazy danych nagrań motion capture Uniwersytetu Carnegie Mellon [16] i przekonwertowano do formatu FBX tak, aby można je było edytować w programie Blender. Krótki opis charakterystyki ruchu każdej z animacji przedstawiono na Tabeli 1.

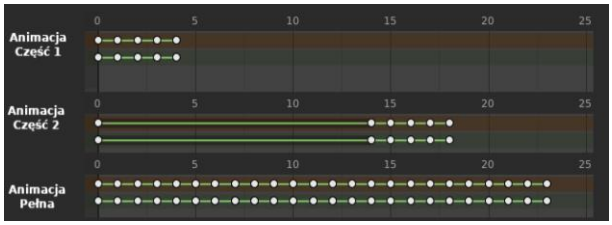
Tabela 1: Charakterystyki ruchu badanych animacji

Nazwa animacji	Opis charakterystyki ruchu
Crouch Jump	Przejście z pozycji kucającej do podskoku
Hands Forward	Podniesienie obu rąk do przodu
Hands Sides	Podniesienie obu rąk na boki
Hands Up	Podniesienie obu rąk do góry
Hour Check	Sprawdzenie godziny na zegarku ręcznym
Jumping Jack Begin	Przejście z pozycji neutralnej stojącej do pozycji wejściowej przy robieniu pajacyków
Jumping Jack End	Przejście z robienia pajacyków do pozycji neutralnej stojącej
Jump Begin	Wyskok do góry rozpoczynający się od neutralnej pozycji stojącej
Jump End	Lądowanie po wyskoku i przejście do pozycji neutralnej stojącej
Punch	Bokserski cios prawą ręką rozpoczynający się od gardy i kończący na gardzie
Side Step Begin	Przejście z pozycji neutralnej stojącej do kroku odstawno dostawnego
Side Step End	Przejście z kroku odstawno dostawnego do pozycji neutralnej stojącej
Step Up	Wejście na schodek rozpoczynające się od pozycji neutralnej stojącej
Walk Backward	Ciągły, trwający chód do tyłu
Walk Forward	Ciągły, trwający chód do przodu

Animacje zostały odpowiednio wycięte i ustawione tak, aby ruch był generowany z częstością sześćdziesięciu klatek na sekundę. Dla każdego badanego fragmentu ruchu przygotowano zestaw trzech animacji:

- animacji pełnej - wycinek trwający od piątej klatki przed badanym fragmentem do piątej klatki po badanym fragmencie
- animacji niepełnej część pierwsza - wycinek animacji zawierający 5 klatek przed badanym fragmentem
- animacji niepełnej część druga - wycinek animacji zmodyfikowany w taki sposób, że poza wyjściowa badanego fragmentu zostaje ustawiona i zachowana na czas równy czasowi trwania badanego fragmentu, następnie ruch kontynuowany jest przez pozostałe 5 klatek jak w animacji oryginalnej.

Na Rysunku 1 przedstawiono przykładowy wygląd osi czasu dla każdej animacji.

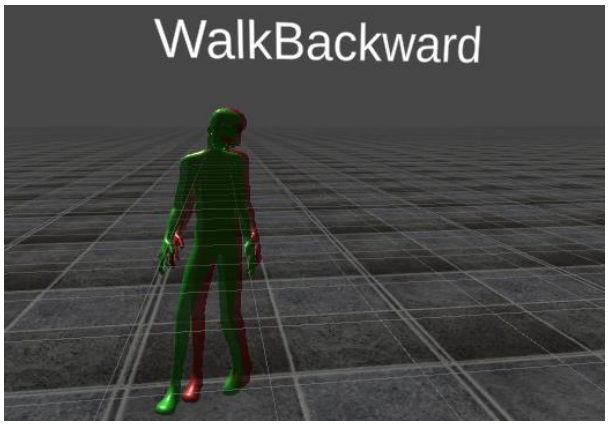


Rysunek 1: Oś czasu dla zestawu animacji w programie Blender.

Dodatkowe klatki otaczające badany fragment zostały dodane w celu upewnienia się, że animacja oryginalna i sekwencja animacji z wygenerowanym przejściem są prawidłowo zsynchronizowane. Przy prawidłowej synchronizacji we fragmentach poza generowanym przejściem różnice w pozycjach i rotacjach kości powinny być bardzo zbliżone do zera.

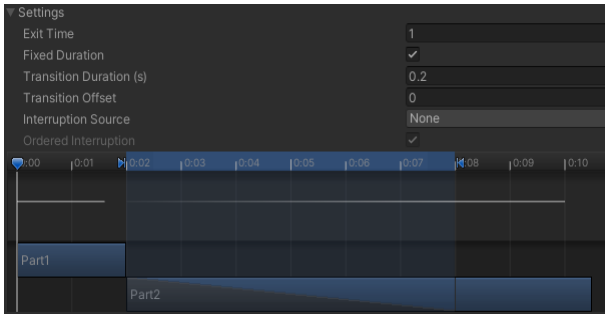
2.3 Przygotowanie środowiska badawczego w programie Unity

Tak przygotowane animacje wyeksportowano do Unity gdzie stworzono dla nich graficzne reprezentacje nazywane dalej awatarami. Przykładową parę awatarów pokazano na Rysunku 2.



Rysunek 2: Graficzna reprezentacja zestawu animacji, widoczne są niewielkie różnice powstałe w wyniku generowania przejścia.

Zielona postać prezentuje animację pełną natomiast czerwona sekwencje animacji niepełnej części pierwszej oraz części drugiej. Sposób konfiguracji sekwencji przedstawiono na Rysunku 3.



Rysunek 3: Widok osi czasu sekwencji animacji oraz jej ustawień w programie Unity.

W dolnej części rysunku widoczna jest oś czasu sekwencji. Podpisany 'Part1' przedstawia trwanie pierwszej części animacji niepełnej, następnie odgrywana jest część druga animacji niepełnej. We fragmencie podświetlonym na niebiesko fragment 'Part2' nie porusza postacią (zachowana jest postawa po badanym fragmencie) dzięki czemu silnik interpoluje pomiędzy początkową a końcową pozą badanego fragmentu co skutkuje generacją przejścia, następnie odgrywana jest końcowa część ruchu w celu sprawdzenia synchronizacji z animacją główną po wygenerowaniu przejścia.

2.4 Dokonywanie pomiarów

Procesem dokonywania pomiarów zarządza skrypt DataCollector. Jego główną funkcjonalność zaprezentowano na Listingu 1.

Listing 1. Metoda Update wykonywana co klatkę, zawarta w skrypcie DataCollector

```
private void Update()
{
    if (fullAvatarAnimationHandler.ImActive &&
        partialAvatarAnimationHandler.ImActive)
    {
        SyncRootBones();
        fullAnimationData.Add(
            item: fullAvatarAnimationHandler.GetFrameData());
        partialAnimationData.Add(
            item: partialAvatarAnimationHandler.GetFrameData());
        fullAvatarAnimationHandler.Animator // Animator
            .Update(Time.deltaTime);
        partialAvatarAnimationHandler.Animator // Animator
            .Update(Time.deltaTime);
    }
    else
    {
        SaveData();
        enabled = false;
    }
}
```

Co klatkę działania programu następuje synchronizacja pozycji i rotacji obu awatarów tak aby zniwelować błędy w pomiarach wynikające z przesunięcia całego modelu w wyniku animacji, następnie informacje o kościach są pobierane i zapisywane z każdego z awatarów co zaprezentowano na Listingu 2.

Listing 2. Metoda odpowiadająca za pobieranie danych o stanie szkieletu zawarta w skrypcie AvatarAnimationHandler

```
public List<BoneFrameData> GetFrameData() {
    List<BoneFrameData> bonesFrameData = new List<BoneFrameData>();
    foreach (var keyValuePair in bones) {
        Transform boneT = keyValuePair.Value;
        bonesFrameData.Add(
            item: new BoneFrameData(boneT.position,boneT.rotation));
    }
    return bonesFrameData;
}
```

W dalszej kolejności oba animatory są aktualizowane. Po zakończeniu pracy przynajmniej jednego z animatorów obliczane są różnice w pozycjach oraz kąty pomiędzy rotacjami kości awatara animacji oryginalnej i z przejściem, dla każdej klatki. Implementację tych obliczeń zaprezentowano na Listingach 3 oraz 4.

Listing 3. Metoda odpowiadająca za obliczenie kątów pomiędzy rotacjami kości

```
private void CompareRotations(ref string[,] initialTable,
                List<List<BoneFrameData>> animationDataA,
                List<List<BoneFrameData>> animationDataB) {
    for (int j = 1; j < animationDataA.Count + 1; j++)
        for (int k = 1; k < animationDataA[0].Count + 1; k++) {
                float difference =
                    Quaternion.Angle(
                        a: animationDataA[j - 1][k - 1].rot,
                        b: animationDataB[j - 1][k - 1].rot);
                initialTable[j, k] = difference.ToString(culture);
        }
}
```

Metoda CompareRotations przyjmuje jako parametry dane z dwóch badanych awatarów oraz referencję do tablicy dwuwymiarowej, która przetrzymuje wyniki obliczeń. Funkcja iteruje po każdej parze danych z dwóch awatarów, biorąc pod uwagę każdą kość w każdej klatce generowania przejścia i oblicza kąt pomiędzy rotacjami kości należących do dwóch awatarów, następnie wpisuje wynik do tablicy.

Listing 4. Metoda odpowiadająca za obliczenie różnicy pozycji pomiędzy rotacjami kości

```
private void ComparePositions(ref string[,] initialTable,
    List<List<BoneFrameData>> animationDataA,
    List<List<BoneFrameData>> animationDataB)
{
    for (int j = 1; j < animationDataA.Count + 1; j++)
        for (int k = 1; k < animationDataA[0].Count + 1; k++)
        {
            float difference = Vector3.Magnitude(
                vector: animationDataA[j - 1][k - 1].pos -
                animationDataB[j - 1][k - 1].pos);
            initialTable[j, k] = difference.ToString(culture);
        }
}
```

Metoda ComparePositions działa analogicznie do metody CompareRotations i oblicza różnicę pomiędzy pozycjami kości należących do dwóch awatorów poprzez obliczenie długości wektora będącego różnicą

wektorów pozycji porównywanych kości, następnie zapisuje wynik.

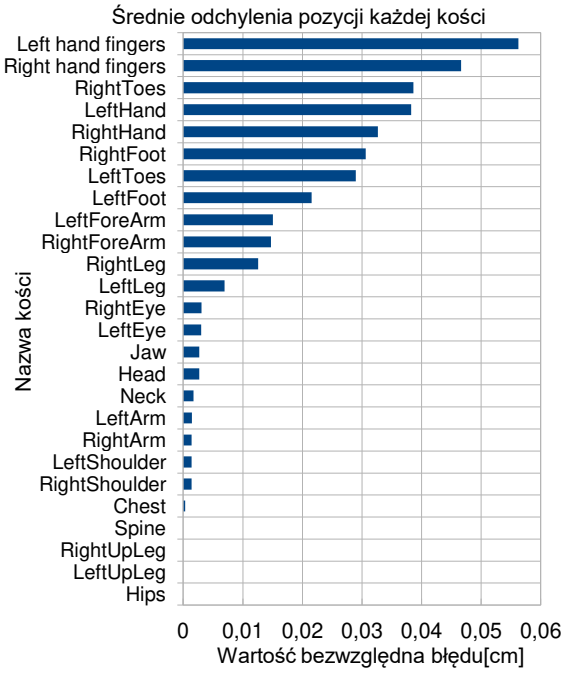
Otrzymane w ten sposób dane są parsowane do postaci tabelarycznej i zapisywane pliku CSV. Dla każdej z badanych animacji generowany jest oddzielny plik zawierający dane dotyczące rotacji oraz drugi zawierający dane dotyczące pozycji. Pliki importowane są do programu LibreOffice Calc, gdzie poddawane są analizie, oraz generowane są wykresy w celu graficznego przedstawienia wyników badań.

3. Wyniki badań

Podczas eksperymentu przeanalizowano 15 animacji przedstawiających różnorodne ruchy postaci humanoidalnej. Do analizy wykorzystano wartości bezwzględne błędów. Zostały one przeliczone na centymetry według instrukcji podanej przez twórców animacji źródłowych [17].

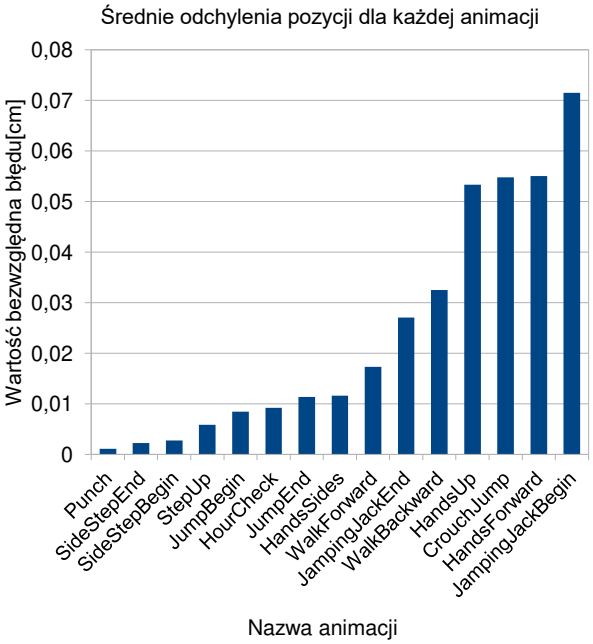
Zdecydowano się na pominięcie danych pochodzących z kości animujących części twarzy, ponieważ żadna z badanych animacji nie dotyczyła mimiki. Ponadto uśredniono dane pochodzące z kości palców u obu dłoni i potraktowano jako pojedyncze kości, po jednej na każdą dłoń, gdyż błędy generowane przez kości palców były bardzo zbliżone do siebie i ich liczba utrudniała analizę wyników. Wyniki przeprowadzonych badań zostały przedstawione w formie czterech porównań:

- porównania średnich odchyleń pozycji na klatkę dla każdej kości (Rysunek 4)
- porównania średnich odchyleń pozycji na klatkę dla całego szkieletu (Rysunek 5)
- porównania średnich odchyleń rotacji na klatkę dla każdej kości (Rysunek 6)
- porównania średnich odchyleń rotacji na klatkę dla całego szkieletu (Rysunek 7)



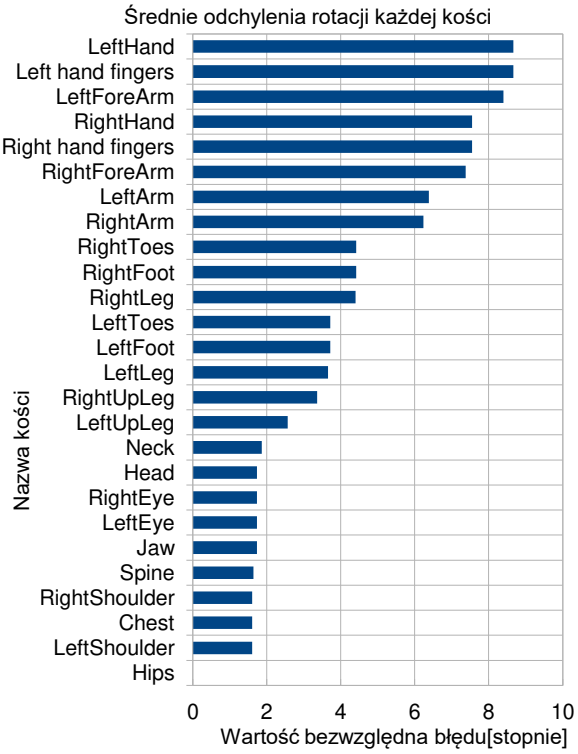
Rysunek 4. Wykres przedstawiający średnie odchylenia pozycji dla każdej kości.

Na Rysunku 4 widać, że największe odchylenia występują w kościach dłoni oraz stóp, natomiast najmniejsze w kościach tułowia i ud.



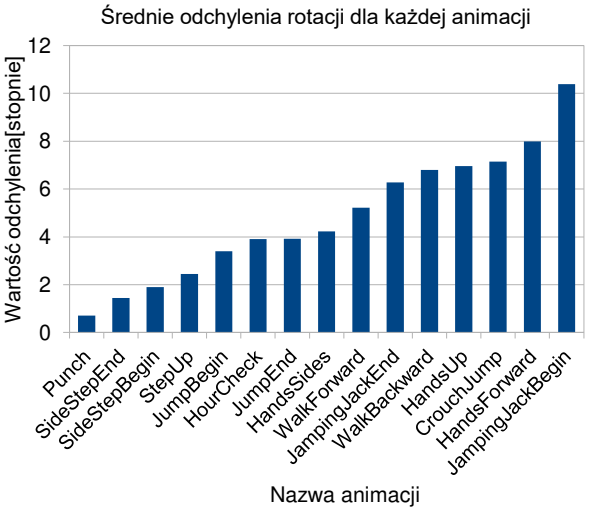
Rysunek 5. Wykres przedstawiający średnie odchylenia pozycji dla każdej animacji.

Po analizie średnich odchyleń dla pojedynczych animacji zaobserwowano, że największe błędy występują w animacjach, w których ruch jest dynamiczny i angażowane są górne kończyny postaci. Mniejsze błędy występują w animacjach powolnych, bądź angażujących tylko jedną kończynę (Rysunek 5).



Rysunek 6. Wykres przedstawiający średnie odchylenia rotacji dla każdej kości.

W przypadku odchyleń w rotacji zaobserwowano około dwukrotnie większe wartości błędów dla kości kończyn górnych niż dla kości kończyn dolnych. Najmniejsze wartości odchyleń, podobnie jak w przypadku odchyleń pozycji, zanotowano dla kości tułowia oraz głowy, zostało to zaprezentowane na Rysunku 6.



Rysunek 7. Wykres przedstawiający średnie odchylenia rotacji dla każdej animacji.

Na wykresie z Rysunku 7 można zauważyć, że występuje silna korelacja wartości błędów odchylenia pozycji z wartością błędów odchylenia rotacji. Animacje, które mają tendencję do generowania dużych błędów odchyleń pozycji, generują również duże błędy odchyleń rotacji.

4. Podsumowanie

Celem niniejszego artykułu była analiza porównawcza przejść generowanych przez silnik Unity w przypadku różnych rodzajów ruchów. Przeprowadzono badanie odchyleń rotacji oraz pozycji dla każdej kości awatarów użytych do badania. Wyniki przeanalizowano w kontekście porównania średnich błędów każdej animacji oraz w kontekście średnich błędów każdej kości.

Zauważono, że największe błędy generują ruchy angażujące górne kończyny postaci, oraz prezentują dużą dynamikę ruchu. Ponadto zaobserwowano tendencję do kumulowania się błędów w kościach, które występują na końcach kończyn (np. palców lub dłoni). Warto również zauważyć, że w przypadku badania błędów w kontekście całych animacji wystąpiła korelacja pomiędzy wartościami błędów rotacji oraz pozycji, czego nie można powiedzieć o kontekście pojedynczych kości.

Obserwacje wynikające z przeprowadzonego badania, pozwalają wyciągnąć wniosek, że silnik Unity lepiej generuje przejścia pomiędzy animacjami, które są powolne i angażują dolne kończyny ciała. Może to stanowić wstęp do bardziej szczegółowej analizy podjętego tematu.

Literatura

[1] M. Masuch, N. Röber, Game graphics beyond realism: Then, now and tomorrow, Level UP: digital games research conference (DIGRA), Faculty of Arts, University of Utrecht, 2004, http://www.digra.org/wp-content/uploads/digital-library/05150.48223.pdf.

[2] J.K. Hodgins, J.F. O'Brien, J. Tumblin, Perception of human motion with different geometric models, IEEE Transactions on Visualization and Computer Graphics 4 (4) (1998) 307-316, https://doi.org/10.1109/2945.765325.

[3] M. Oesker, H. Hecht, B. Jung, Psychological Evidence for Unconscious Processing of Detail in Real-time Animation of Multiple Characters, The Journal of Visualization and Computer Animation 11 (2) (2000) 105-112, https://doi.org/10.1002/1099-1778(200005)11:2<105::AID-VIS222>3.0.CO;2-Q.

[4] J. Lee, J. Chai, P.S. Reitsma, J.K. Hodgins, N.S. Pollard, Interactive control of avatars animated with human motion data, ACM Transactions on Graphics 21 (3) (2002) 491-500, https://doi.org/10.1145/566654.566607.

[5] C. Rose, B. Guenter, B. Bodenheimer, M.F. Cohen, Efficient generation of motion transitions using spacetime constraints, In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (1996) 147-154, https://doi.org/10.1145/237170.237229.

[6] T. Polichroniadis, N. Dodgson, Motion blending using a classifier system, In Proceedings of the 7th International Conference in Central Europe on Computer Graphics 1 (1999) 225-232, http://www.neildodgson.com/pubs/WSCG99.pdf.

[7] G. Ashraf, K.C. Wong, Generating consistent motion transition via decoupled framespace interpolation, Blackwell Publishers Ltd. Oxford, UK and Boston, USA, Computer Graphics Forum 19 (3) (2000) 447-456, https://doi.org/10.1111/1467-8659.00437.

[8] L. Kovar, M. Gleicher, Flexible automatic motion blending with registration curves, In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '03), Eurographics Association, Goslar, DEU, (2003) 214–224, http://dx.doi.org/10.2312/SCA03/214-224.

[9] M. Gleicher, H.J. Shin, L. Kovar, A. Jepsen, Snap-together motion: assembling run-time animations, ACM SIGGRAPH (2008) 1-9, https://doi.org/10.1145/641480.641515.

[10] V.B. Zordan, A. Majkowska, B. Chiu, M. Fast, Dynamic response for motion capture animation, ACM Transactions on Graphics 24 (3) (2005) 697-701, https://doi.org/10.1145/1073204.1073249.

[11] H.J. Shin, H.S. Oh, Fat graphs: constructing an interactive character with continuous controls, In Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation (2006) 291-298, http://dx.doi.org/10.2312/SCA/SCA06/291-298.

[12] D. Holden, T. Komura, J. Saito, Phase-functioned neural networks for character control, ACM Transactions on Graphics (TOG), 36 (4) (2017) 1-13, https://doi.org/10.1145/3072959.3073663.

[13] F. Gaisbauer, P. Fröhlich, J. Lehwald, P. Agethen, E. Rukzio, Presenting a Deep Motion Blending Approach for Simulating Natural Reach Motions, Eurographics (Posters) (2018) 5-6, http://dx.doi.org/10.2312/egp.20181010.

[14] F. Gaisbauer, J. Lehwald, J. Sprenger, E. Rukzio, Natural posture blending using deep neural networks, In Proceedings of the 12th ACM SIGGRAPH Conference

on Motion, Interaction and Games (2019) 1-6, https://doi.org/10.1145/3359566.3360052.

[15] Dokumentacja Unity dotycząca generowania przejść, https://docs.unity3d.com/Manual/class-Transition.html, [01.10.2023].

[16] Baza danych animacji używanych w badaniu, http://mocap.cs.cmu.edu/, [01.10.2023].

[17] Informacje dotyczące przekonwertowania jednostek miary w animacjach na centymetry, http://mocap.cs.cmu.edu/faqs.php, [01.10.2023].