JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 30 (2024) 53–60
Received: 17 October 2023
Accepted: 4 December 2023

Comparative analysis of the performance of Unity and Unreal Engine game engines in 3D games

Analiza porównawcza wydajności silników Unity i Unreal Engine w grach 3D

Kamil Abramowicz*, Przemysław Borczuk

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

Abstract
The article compared the performance of the Unity and Unreal Engine game engines based on tests conducted on two nearly identical games. The research focused on frames per second, CPU usage, RAM, and GPU memory. The results showed that Unity achieved a better average frame rate. Unreal Engine required more RAM and GPU resources. Analyzing CPU load values revealed that on the first system, Unity demanded less CPU usage. However, on the second system, Unreal Engine used over 10 percentage points less CPU. The conclusions from the research partially confirm the hypothesis that Unity requires fewer computer resources, although in some cases, Unreal Engine may demand fewer CPU resources.

Keywords: game engine; Unity; Unreal Engine; performance

Streszczenie
W artykule porównano wydajność silników gier Unity i Unreal Engine na podstawie badań przeprowadzonych na dwóch bliźniaczo podobnych grach. Badania skupiły się na: liczbie klatek na sekundę, użyciu procesora (CPU), RAM i karty graficznej. Wyniki wykazały, że Unity osiągnął lepszą średnią liczbę klatek na sekundę. Unreal Engine wymagał większych zasobów pamięci RAM oraz karty graficznej. Przeanalizowane wartości obciążenia CPU pokazały, że na pierwszym stanowisku silnik Unity wymagał mniejszego użycia CPU. Natomiast na drugim stanowisku Unreal Engine wykorzystywał ponad 10 punktów procentowych CPU mniej. Wnioski z przeprowadzonych badań częściowo potwierdzają tezę, że Unity wymaga mniej zasobów komputera, choć w niektórych przypadkach Unreal Engine może wymagać mniej zasobów CPU.

Słowa kluczowe: silnik gier; Unity; Unreal Engine; wydajność

*Corresponding author
Email address: **kamil.abramowicz@pollub.edu.pl** (K. Abramowicz)

1. Wstęp

Silniki gier, dzięki swojej wszechstronności oraz możliwości ich użycia przy wielu różnych projektach, odgrywają znaczącą rolę podczas tworzenia gier. Dzięki nim deweloperzy mogą skupić się na konstruowaniu jak najlepszych aplikacji wykorzystując dostępne funkcje oraz rozwiązania używane w poprzednich produkcjach. Dostępne na rynku silniki znacznie przyspieszają rozwój projektów, jak również zwiększają jakość produktu. Narzędzia te są rozwijane wraz z upływem lat, co poprawia ich wydajność. Darmowy dostęp do środowisk ułatwiających urzeczywistnianie własnych pomysłów na gry przyciąga licznych deweloperów specjalizujących się w wytwarzaniu zasobów przeznaczonych właśnie dla takich środowisk.

Wszystkie wyżej wymienione zalety przyczyniają się do usprawnienia procesu tworzenia gier, umożliwiając rozwijanie zoptymalizowanych aplikacji w krótszym czasie. Duży wpływ na działanie gry mają efekty cząsteczkowe, które używane są przy np. wybuchach czy opadach płatków śniegu. Innym czynnikiem wpływającym na wydajność aplikacji jest renderowanie. Przy generowaniu obiektów graficznych złożonych z dużej liczby wielokątów gra może wymagać zbyt dużo zasobów komputera. Nie można jednak skupić się wyłącznie na optymalizacji jednego obszaru projektu i zignorować pozostałych, ponieważ w takim przypadku gra nie będzie działać wystarczająco dobrze.

Biorąc pod uwagę popularność silników gier, jak również ich możliwości, do analizy wybrane zostały dwa: Unity oraz Unreal Engine. Na potrzeby porównania w podanych środowiskach programistycznych napisana została gra 3D. Następnie w każdej z nich zmierzono zużycie zasobów komputerowych podczas wykonywania przez gracza akcji takich jak np. strzał z pistoletu, rzut granatem lub nawigacja postacią.

Celem niniejszej pracy było wieloaspektowe porównanie wydajności silników Unity i Unreal Engine oraz przeanalizowanie wymaganych zasobów komputera przez utworzoną w nich grę 3D. W badaniach skupiono się na obciążeniu sprzętu wywieranym przez grę podczas wykonywania przez gracza różnych akcji.

Zakres pracy obejmował wybór obiektów badawczych (silników gier), metodyki badań, a także mierzonych zasobów komputera. Zaprojektowana została trójwymiarowa gra, która została wdrożona w dwóch środowiskach w taki sposób, aby była jak najbardziej zbliżona w wybranych środowiskach. Kolejnym krokiem

było wykonanie badań oraz analiza wyników. Z przeprowadzonych działań wyciągnięto oraz sformułowano wnioski.

Na potrzeby badania postawiono następującą hipotezę badawczą: Gry 3D tworzone z pomocą Unity wymagają mniej zasobów komputera niż te zaprogramowane przy użyciu Unreal Engine.

2. Przegląd literatury

W trakcie projektowania i programowania gier bardzo ważna jest optymalizacja, aby gra działała wydajnie na jak największej liczbie urządzeń. Wiele prac skupia się głównie na zestawieniu silników pod względem ich funkcjonalności i możliwości. Przedmiotem pracy [1] było porównanie efektywności aplikacji opracowanych przy pomocy środowisk programistycznych Unity oraz CryEngine. Wytworzone aplikacje posiadały zbliżone poziomy rozgrywki, a także podobne skrypty generujące obiekty. Projekty pozwoliły na wygenerowanie określonej liczby obiektów obciążających silniki. Przeanalizowano zużycie pamięci RAM, wykorzystanie procesora, liczbę klatek na sekundę oraz czas generowania obiektów przez narzędzie. Wybrano cztery poziomy obciążeń - 1000, 5000, 7500 oraz 10000 wygenerowanych obiektów. Odczyt mierzonych parametrów dokonano po minucie od wygenerowania obiektów, pomiary powtórzono trzykrotnie. Generowane obiekty były rozmieszczane losowo w obrębie sceny, posiadały one właściwości fizyczne: masę, podatność na grawitację oraz kolizje. Na podstawie uzyskanych wyników wyciągnięto następujące wnioski: Unity potrzebuje mniejszej ilości zasobów do sprawnego działania, niezależnie od liczby obiektów wymagane zasoby dla CryEngine są zbliżone, silnik fizyki CryPhysics nie radzi sobie z dużą liczbą obiektów fizycznych.

Temat zestawienia ze sobą silników gier podjęty został w pracy [2], w której przeprowadzono analizę porównawczą środowisk CryEngine, Unreal Engine oraz Unity. Kryteriami porównania były możliwości techniczne oraz czynniki wpływające na popularność wśród użytkowników. Celem pracy było wykazanie słabych i mocnych stron oraz prezentacja różnic pomiędzy narzędziami. Badając różnice brano pod uwagę wiele czynników m.in. wieloplatformowość, języki programowania, modelowanie czy dokumentację. Wysnuto wnioski, że Unity jest najlepszym silnikiem dla początkujących, ponieważ posiada dobrą dokumentację, dostępnych jest wiele kursów i szablonów, wspiera wiele platform oraz jako jedyny z porównywanych narzędzi posiada dedykowany tryb 2D. Wadami tego środowiska są zaś najsłabsza jakość graficzna tworzonych na nim gier, niewielkie możliwości wbudowanych narzędzi do animacji, reżyserowania przerywników filmowych oraz dodawania efektów dźwiękowych. Unreal Engine posiada szeroki zakres opcji oraz edytorów, które mogą przytłoczyć początkującego użytkownika. Dzięki możliwości użycia Blueprint'ów Unreal Engine jest dobrym rozwiązaniem dla użytkowników bez umiejętności programowania. Silnik ten udostępnia dobry edytor materiałów, ułatwia tworzenie zaawansowanych animacji oraz posiada systemy wspierające implementacje sztucznej inteligencji. W większości eksperymentów gra zaimplementowana w tym środowisku była najmniej efektywna. Silnik CryEngine jest narzędziem, które nie jest polecane dla użytkowników bez doświadczenia w pracy nad grami, posiada on małą liczbę kursów oraz słabą dokumentację. Narzędzie to jest przeznaczone do produkcji gier typu FPS (strzelanka pierwszoosobowa, ang. First-person shooter). W testach wydajności gra zaimplementowana w CryEngine z najwyższymi ustawieniami graficznymi działała najbardziej optymalnie.

Innym sposobem porównania silników gier Unity i Unreal Engine jest znalezienie wspólnych części i porównanie ich z osobna tak jak zrobiono to w pracy [3]. Według autora wszystkie silniki gier składają się z silników dźwięku, fizyki, renderowania oraz animacji i sztucznej inteligencji. Moduł odpowiadający za renderowanie pozwala na wyświetlanie obiektów poprzez kontrolowanie karty graficznej. Gra potrzebuje także starannie dobranej lub skomponowanej muzyki, która jest wgrywana i dostosowywana przez silnik dźwięku. Natomiast silnik fizyki pozwala na zredukowanie czasu programowania każdej reguły fizyki, która jest zauważalna, ponieważ wystarczy uruchomić ją wybraną metodą klasy. Sztuczna inteligencja odpowiada za zachowania obiektów niekontrolowanych bezpośrednio przez gracza takich jak zwierzęta, ludzi itp. Niezbędną częścią nowoczesnych gier są animacje np. przeładowanie pistoletu czy poruszanie się postaci. Z badań wykonanych przez autorów wynika, że obydwa narzędzia są bardzo podobne z tym, że Unreal Engine ma np. lepsze efekty świetlne natomiast Unity produkuje lepszej jakości cienie. Wybór więc zależy od preferencji dewelopera.

W dzisiejszych czasach bardzo ważną częścią tworzenia gier jest zapewnienie jak najbardziej realistycznej grafiki współpracującej z jak najlepszą fizyką. Dodatkowe narzędzie, które pomaga nadać grze realizmu to efekty cząsteczkowe. Pozwalają one urzeczywistnić świat gry poprzez dodanie unoszących się pojedynczych płatków śniegu, odłamków powstających w wyniku wybuchów oraz wielu innych efektów. Zhang J. opisał w swojej pracy [4] proces tworzenia cząsteczek śniegu oraz jego optymalizację używając Unreal Engine 4. Silnik ten używa modułów opisujących każdy aspekt wytwarzanych cząsteczek. Wynik ustawień jest widoczny od razu po wprowadzeniu zmian ułatwiając osiągnięcie pożądanego zachowania i wyglądu. Optymalizacja efektów cząsteczkowych jest najczęściej wykonywana poprzez zredukowanie liczby systemów cząsteczek oraz ich zakresu działania, zmniejszenie liczby emiterów, liczby emitowanych cząsteczek i czasu ich istnienia. Lepszą wydajność można uzyskać także przy pomocy rzadszego obliczania kolizji lub innych zachowań cząsteczek. Kolejnym sposobem optymalizacji gry jest ustalenie odległości z jakiej cząsteczki stają się widoczne i nie wykonywać obliczeń, kiedy są one poza zasięgiem widoczności. Podsumowując, Unreal Engine znacząco ułatwia tworzenie efektów cząsteczkowych oraz ich optymalizację, pozwalając na zmniejszenie obciążenia urządzenia i osiągnięcie równowagi między

efektywnością a atrakcyjnością emitowanych cząsteczek.

W wielu przeanalizowanych pracach skupiono się na porównaniu ogólnych właściwości silników gier. Najlepszym przykładem jest praca [5], w której autorzy stworzyli tabelę porównawczą dla badanych narzędzi. W kolejnej pracy [6] na podstawie najbardziej znanych gier wyprodukowanych przy użyciu wybranych środowisk, przeprowadzono testy, które miały na celu ocenę efektywności silników gier. Badane w pracy gry zostały wyprodukowane w znacznym odstępie czasu, przez różne firmy, a także są to inne gatunki gier, dlatego w naszej pracy przyjęliśmy inną metodologię. Sposobem przyjętym do wykonania tej pracy był sposób pokazany w artykule [7] i [8], który polega na zbudowaniu bliźniaczych gier na dwóch silnikach, co eliminuje problemy z poprzedniej pracy. W pracy [9] autorzy przedstawili porównanie silników gier za pomocą tabel, w których zaznaczono różnice w wielu aspektach takich jak renderowanie, fizyka, a nawet platformy, na które można zbudować aplikację. Z kolei pozycja [10] skupia się na procesach teksturowania obiektów i określeniu najbardziej efektywnego sposobu pracy. Teksturowanie jest nieodzownym elementem procesu produkcji gier, lecz duża liczba tekstur o wysokiej rozdzielczości może niekorzystnie wpłynąć na działanie gry. Autor artykułu [11] wybiera czternaście silników gier i krótko opisuje każdy z nich. W porównaniu brane pod uwagę są aspekty takie jak platforma publikacji gry, koszt korzystania z narzędzia, języki programowania i wspierane platformy.

Autorzy niektórych prac badali tylko jeden wybrany aspekt silnika, przez co analiza narzędzi nie jest wyczerpująca. Dlatego w niniejszej pracy zdecydowano się na porównanie wydajności bliźniaczych gier utworzonych za pomocą dwóch silników.

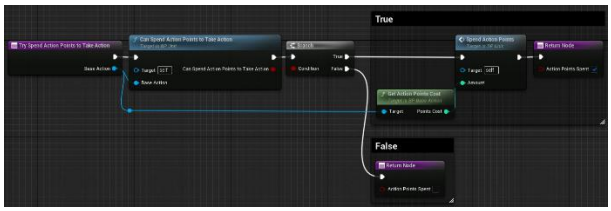
3. Metoda badawcza

Celem przeprowadzonych badań było porównanie wydajności silników gier Unity oraz Unreal Engine w zaprojektowanej strategicznej grze 3D. Gra w środowisku Unity została przygotowana przy użyciu języka programowania C#, podczas gdy w przypadku Unreal Engine skorzystano wyłącznie z Blueprint'ów - narzędzia do graficznego programowania, które pozwala stworzyć grę bez konieczności pisania kodu w języku C++. Wygląd przykładowej funkcji wykonanej z wykorzystaniem C# i Blueprint'ów widoczny jest na Listingach 1 i 2.

Listing 1: Wygląd przykładowej funkcji w C# (Unity)

```
public bool TrySpendActionPointsToTakeAction(BaseAction baseAction)
{
    if (CanSpendActionPointsToTakeAction(baseAction))
    {
        SpendActionPoints(baseAction.GetActionPointsCost());
        return true;
    }
    else
    {
        return false;
    }
}
```

Listing 2: Wygląd przykładowej funkcji w Blueprint'cie (Unreal Engine)



Do zmierzenia wydajności wykorzystane zostały narzędzia MSI Afterburner i RivaTuner Statistics Server, a mierzonymi wskaźnikami były: liczba klatek na sekundę, wykorzystanie procesora komputera, użycie pamięci RAM oraz procesora graficznego.

3.1. Obiekt badań

Obiektem badań była turowa gra strategiczna, w której użytkownik steruje postaciami na utworzonym poziomie. Obliczanie ścieżki poruszania się postaci wykonano za pomocą algorytmu A* Pathfinding [12]. Gracz ma także dostęp do innych rodzajów ruchów tj. przeczekania tury, strzału bronią palną, rzutu granatem, wykonania ataku z bliska, czy interakcji z drzwiami. Mapa odkrywana jest przed użytkownikiem wraz z otwarciem drzwi do kolejnych pomieszczeń. Dana gra została zaprogramowana na dwóch silnikach gier - Unity oraz Unreal Engine. Wykorzystano najnowszą wersję Unreal Engine 5.2.1 oraz najnowszą długo wspieraną odsłonę (LTS – ang. Long Term Support) Unity – 2021.3.18f1.

3.2. Stanowiska testowe

Zbudowane gry przetestowane zostały na dwóch komputerach, których parametry znajdują się w Tabeli 1. Na pierwszym stanowisku gra uruchomiona została w rozdzielczości 2560x1440 pikseli, na drugim zaś w rozdzielczości 1920x1080 pikseli.

Tabela 1: Stanowiska testowe

	Stanowisko nr 1 (S1)	Stanowisko nr 2 (S2)
System operacyjny	Windows 11	Windows 10
Procesor	Intel Core i7-13700K, 16 rdzeni (8 Performance + 8 Efficient)	Intel Core i5-9300HF, 4 rdzenie
Pamięć RAM	32 GB DDR4, 4000 MHz, CL 18-22-22-42	8 GB DDR4, 2400 MHz, CL 17-17-17-39
Karta graficzna	NVIDIA RTX 4080, 16 GB GDDR6X	NVIDIA GeForce GTX 1650, 4 GB GDDR5

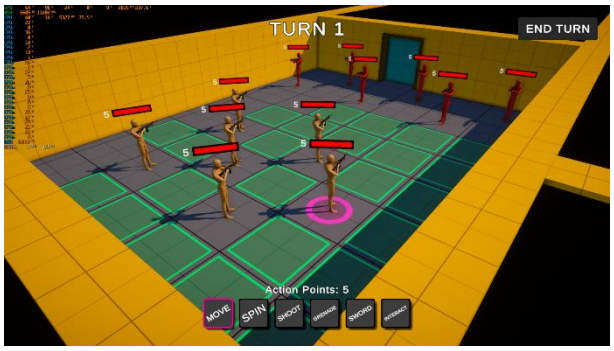
3.3. Przebieg testów

Dla porównania obu silników do tworzenia gier zostały zmierzone następujące zasoby:

- liczba klatek na sekundę,
- wykorzystanie procesora (CPU),
- średnie użycie pamięci RAM,
- średnie wykorzystanie pamięci karty graficznej (pamięć GPU).

Zasoby były mierzone podczas:

- braku aktywności ze strony użytkownika,
- poruszania się postaci (chodzenie jednostkami po mapie, otwieranie drzwi, ruch kamery),
- wykonywania funkcji ataku (strzał z broni palnej, rzut granatem, uderzenie mieczem).



Rysunek 1: Wygląd gry wykonanej w Unity, z widoczną nakładką programu mierzącego statystyki w lewym górnym rogu.

Rysunek 2: Wygląd gry wykonanej w Unreal Engine, z widoczną nakładką programu mierzącego statystyki w lewym górnym rogu.

Wykonywane badania zostały podzielone na 3 grupy, mianowicie: funkcje ataku, poruszania oraz bezruch. Badania podczas braku aktywności użytkownika polegały na uruchomieniu gry, odczekaniu 10 minut, aby wykonały się ewentualne procesy w tle np. kompilacja shaderów, i 15 minutowym pomiarze przez narzędzie do monitorowania zasobów. W testach poruszania postacią zawierały się: akcja poruszania, interakcja z drzwiami w celu odblokowania dostępu do kolejnych pomieszczeń, ruch kamery - brak ruchu kamery ogranicza dostęp gracza do części mapy, która znajduje się poza zasięgiem kamery, czasami konieczne jest obrócenie i przybliżenie kamery, aby wskazać drzwi. Pomiary grupy poruszania polegały na wykorzystaniu wszystkich dostępnych punktów akcji poprzez poruszanie postaciami i otwieranie drzwi. Użytkownik ma do dyspozycji 6 sojuszniczych jednostek, każda z pięcioma punktami akcji. W trakcie jednego uruchomienia gry wykonano 30 ruchów. Badanie powtórzono 10 razy, co dało łącznie 300 działań. Badania akcji strzału i rzutu granatem wyglądały podobnie do siebie. Wykorzystano wszystkie dostępne punkty, w ramach badanej akcji, dla każdej sojuszniczej jednostki do momentu ich wykorzystania - łącznie 300 strzałów oraz 300 rzutów granatem. Ostatnie testy wykonywania uderzenia mieczem przeprowadzono poprzez wcześniejsze podejście jednostkami tak, aby przeciwnicy byli w zasięgu. Następnie włączono monitorowanie zasobów i wykonano każdym sojusznikiem dwa ataki mieczem na sześciu przeciwnikach, co dało łącznie 12 uderzeń mieczem. Mimo dostępnych punktów akcji nie został ani jeden przeciwnik, na którym można by było wykonać uderzenie. Pomiary wykonano 10 razy - łącznie 120 ruchów. Wszystkie powyższe testy zostały wykonane na dwóch wariantach gry (Unity oraz Unreal Engine) przy pomocy dwóch stanowisk.

4. Analiza wyników badań

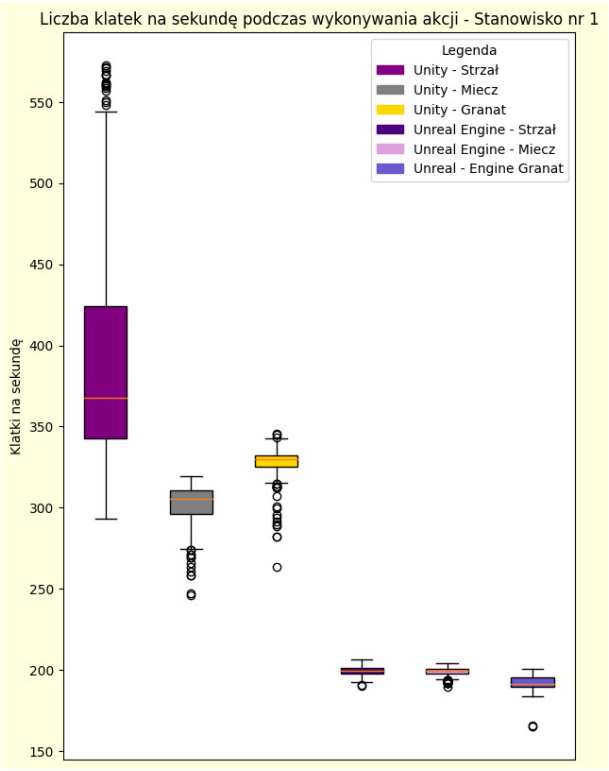
W celu wizualizacji zebranych danych zdecydowano się na wybór trzech różnych typów wykresów: pudełkowych, liniowych oraz słupkowych. Różnorodność tych diagramów umożliwiła dokładne przedstawienie i analizę wyników badań. Kierując się charakterystyką badanych zasobów komputera przydzielono do nich odpowiednie diagramy. Wykresy pudełkowe zostały wykorzystane jako narzędzie do przedstawienia zmian w liczbie klatek na sekundę (fps - ang. frames per second), co pozwoliło na zrozumienie rozkładu i zmienności tej istotnej metryki wydajności gier. Z kolei diagramy liniowe zostały zastosowane w przypadku analizy wykorzystania procesora, co umożliwiło obserwację dynamiki zużycia zasobu w trakcie badanych scenariuszy gry. Grafy słupkowe zostały użyte w kontekście analizy wykorzystania pamięci RAM oraz pamięci karty graficznej, co pozwoliło na klarowne i bezpośrednie przedstawienie zauważonych różnic w tych aspektach.

4.1. Analiza wydajności silników gier po kątem liczby klatek na sekundę

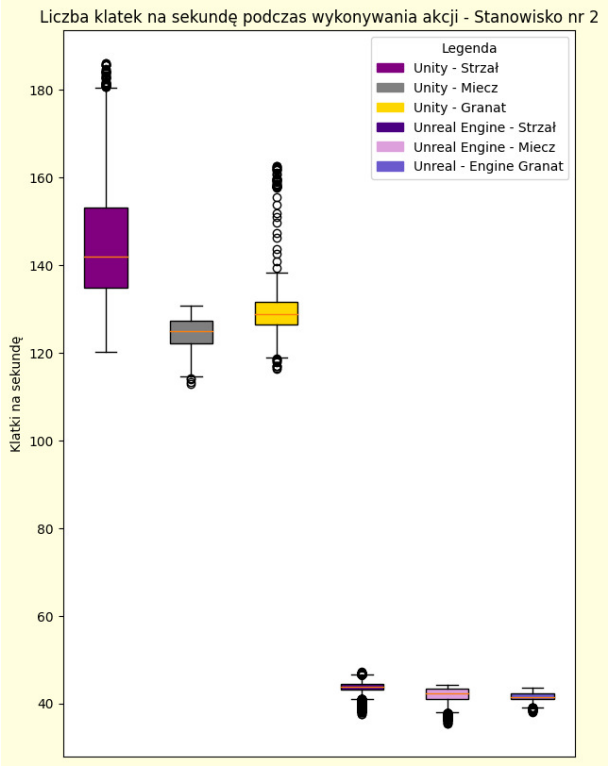
Liczba klatek na sekundę jest istotnym wskaźnikiem wydajności w grach komputerowych. Większa liczba klatek na sekundę oznacza płynniejszy obraz i poprawia komfort obcowania z grą. Jednakże istnieje pewien punkt, po osiągnięciu którego wzrost fps może przestać być zauważalny przez ludzkie oko, przy czym warto zaznaczyć, że ten punkt może różnić się w zależności od konkretnej osoby.

Analizując Rysunek 3, można zauważyć, że w przypadku stanowiska nr 1 (S1) gra wykorzystująca silnik Unity uzyskuje znacznie wyższe wartości. Porównując wyniki tych samych akcji na obu silnikach można zauważyć, że największa różnica median występuje w przypadku akcji strzału, gdzie Unity przewyższa Unreal Engine o 168 klatek. Warto również zaznaczyć, że największa różnica między pierwszym a trzecim kwartylem (Q1, Q3) dla Unreal Engine wyniosła niecałe 6 klatek dla ataku z użyciem granatu, podczas gdy dla Unity wynosiła aż 81 klatek w przypadku akcji strzału. Podsumowując, gra stworzona z użyciem Unity osiągała wyższe wartości fps, jednakże występowały pewne wahania w płynności w porównaniu do gry na Unreal Engine. Z pomiarów widocznych na Rysunku 4 dla stanowiska nr 2 (S2) wynikają podobne wnioski jak dla S1. W tym przypadku największa różnica pomiędzy wartościami środkowymi wyniosła 102 klatki, także dla akcji strzału. Największe odchylenie między pierwszym

a trzecim kwartylem wyniosło prawie 19 fps przy strzale dla Unity, podczas gdy dla Unreal Engine było ono równe 2,1 klatki. Dlatego wnioski są takie same jak dla S1.

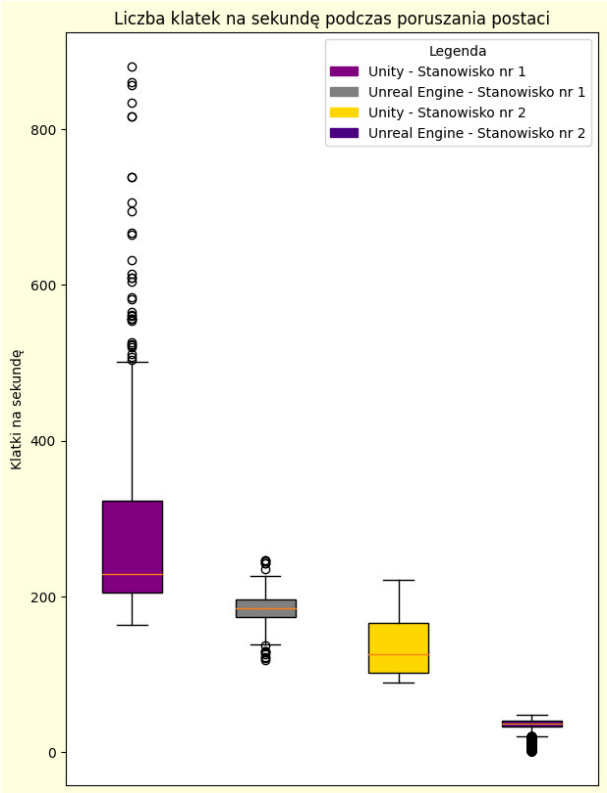


Rysunek 3: Wykresy pudełkowy dla testów funkcji ataku dla stanowiska nr 1.



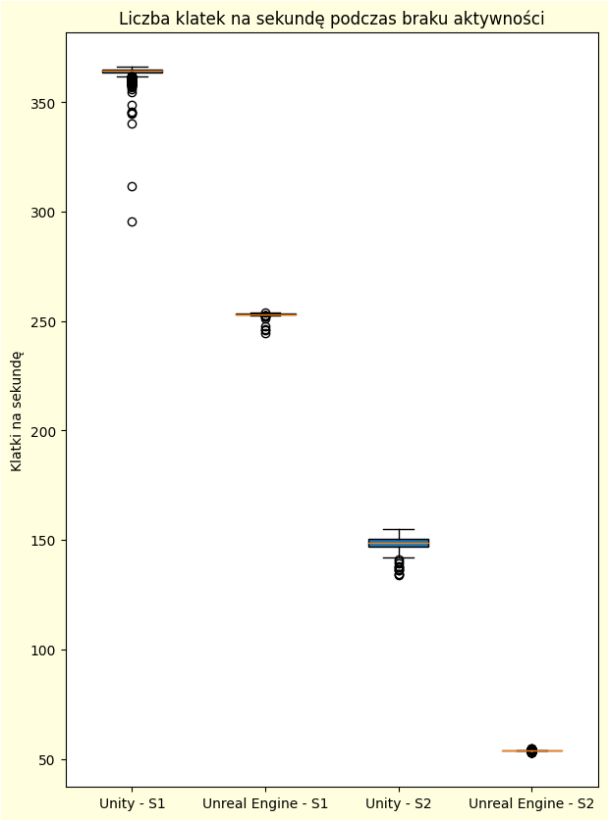
Rysunek 4: Wykresy pudełkowy dla testów funkcji ataku dla stanowiska nr 2.

Pomiary przeprowadzone podczas akcji z grupy poruszania, widoczne na Rysunku 5, wyraźnie wskazują na przewagę Unity nad konkurentem pod względem liczby wyświetlanych klatek na sekundę. Na stanowisku pierwszym różnica między pierwszym a trzecim kwartylem wyniosła prawie 120 klatek dla Unity oraz 23 dla Unreal Engine, przy wyższej medianie o 44 klatki na korzyść Unity. Testy przeprowadzone na drugim stanowisku potwierdzają przewagę silnika Unity. Szczególnie interesujące są różnice między Q1 i Q3 dla aplikacji działającej w środowisku Unreal Engine, wynoszące 23,2 fps dla S1 i 6,8 fps dla S2. Wartości te są wyższe w porównaniu z akcjami związanymi z atakiem, co może wynikać z faktu, że podczas tej akcji wykonywana jest bardzo duża liczba obliczeń. Blueprint'y, używane w Unreal Engine, działają na maszynie wirtualnej, która tłumaczy je na natywny kod języka C++, co może prowadzić do zwiększania czasu potrzebnego na wykonanie obliczeń i w rezultacie zmniejszenia liczby generowanych klatek.



Rysunek 5: Wykresy pudełkowe dla testów funkcji poruszania.

Wyniki pomiarów liczby klatek na sekundę uzyskane podczas uruchomionej gry, gdy użytkownik nie wykonywał żadnych czynności, zostały przedstawione na Rysunku 6. Warto zauważyć, że różnice między pierwszym a trzecim kwartylem nie przekraczały 2 klatek na sekundę dla żadnego ze stanowisk i silników. Spadki wartości były niewielkie, z wyjątkiem kilku obserwowanych spadków przy S1 i Unity. Dla S1 różnica median wyniosła ponad 110 klatek na korzyść Unity, podczas gdy dla S2 była niższa i wynosiła poniżej 100 fps, także na korzyść Unity.

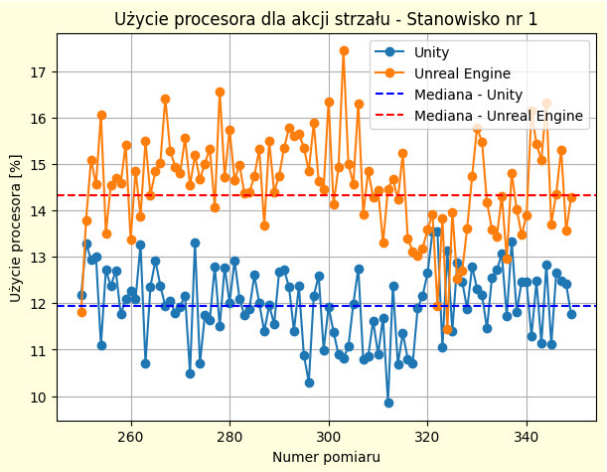


Rysunek 6: Wykresy pudełkowe dla testów bezczynności.

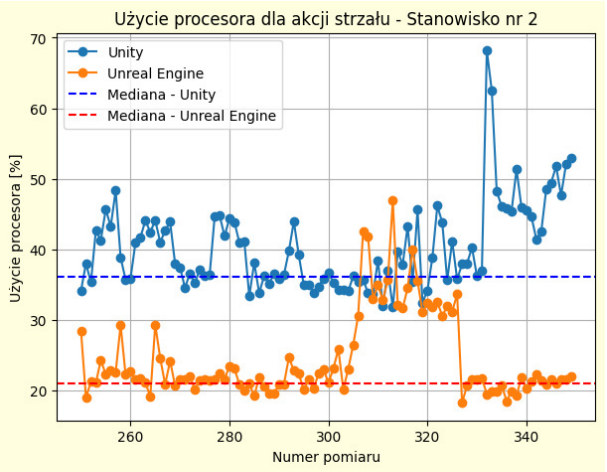
4.2. Analiza wydajności silników gier pod kątem zmian użycia procesora

Procesor (CPU) stanowi jedną z kluczowych jednostek obliczeniowych w komputerze i odgrywa niezwykle istotną rolę w zapewnieniu płynności i wydajności gier komputerowych. W niniejszym podrozdziale przeprowadzono analizę wydajności badanych silników gier pod kątem obciążenia procesora. Przedstawiono wykresy liniowe, które obrazują procentowe użycie procesora, pozwalając na lepsze zrozumienie, jak testowane silniki reagują na różne scenariusze gry. Warto zaznaczyć, że prezentowane na grafach mediany zostały obliczone na podstawie wszystkich zgromadzonych pomiarów, a w celu zachowania czytelności wykresów w pracy ograniczono przedstawienie wyników do pomiarów z zakresu od 250 do 350.

Wyniki otrzymane na stanowisku nr 1, przedstawione na Rysunku 7, wykazują niższe minimalne i maksymalne wartości użycia procesora w Unity w porównaniu do Unreal Engine podczas akcji strzału. Minimalna wartość wynosi mniej niż 10% dla Unity i około 11,4% dla Unreal Engine, podczas gdy maksymalna wartość wynosi 13,5% dla Unity i 17,4% dla Unreal Engine. Mediana dla Unreal Engine jest wyższa o ponad dwa punkty procentowe. Natomiast Rysunek 8 przedstawia wyniki dla stanowiska drugiego, gdzie Unity osiąga wyższe minimalne (około 31,8%) i maksymalne (około 68,2%) wartości w porównaniu do Unreal Engine (minimum około 18,2%; maksimum około 46,9%). Mediana uzyskana dla wyników silnika Unity również była wyższa o ponad 15 punktów procentowych.



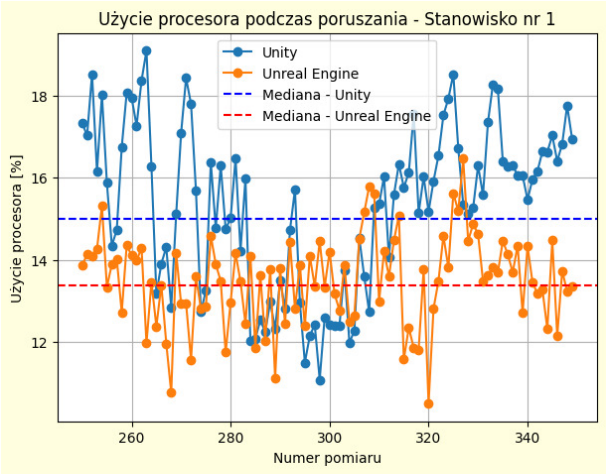
Rysunek 7: Wykresy liniowe testów strzału dla stanowiska nr 1.



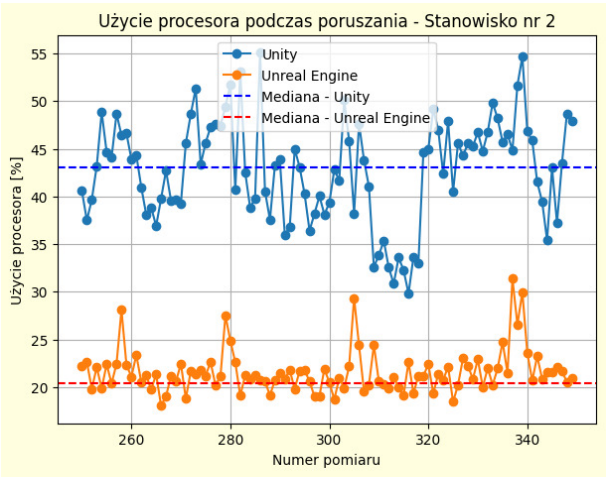
Rysunek 8: Wykresy liniowe testów strzału dla stanowiska nr 2.

Wyniki testów grupy poruszania na S1 widoczne są na Rysunku 9. W przypadku tych pomiarów Unity wykazywało większe obciążenie procesora. Minimalna wartość dla Unity (około 11,1%) była wyższa w porównaniu do konkurencyjnego silnika (minimum około 10,5%). Maksymalna wartość wynosiła około 19,1% dla Unity i około 16,5% dla Unreal Engine, a mediana była na poziomie około 15% dla Unity i około 13,4% dla Unreal Engine. Warto zauważyć, że dla stanowiska pierwszego, akcja poruszania była jedyną, przy której Unity wymagało większego użycia procesora niż Unreal Engine. Może to wynikać, tak jak w przypadku pomiarów fps tej akcji, z ograniczeń maszyny wirtualnej, która tłumaczy Blueprint'y na natywny język C++. Jeśli maszyna wirtualna ogranicza prędkość wykonania funkcji, to obliczenia wykonywane są wolniej i nie jest wymagana w krótszym czasie większa moc procesora. Silnik Unity zaś przy tej akcji osiągał wyższą liczbę klatek na sekundę co oznacza, że obliczenia były wykonywane w krótszym czasie i przy wyższym wykorzystaniu CPU. W przypadku S2 (Rysunek 10), Unity również osiąga wyższe wartości użycia CPU, niż Unreal Engine. Na stanowisku drugim, dla każdej akcji, odnotowano wyższe wartości środkowe w grze stworzonej przy pomocy Unity.

Podsumowując, na stanowisku nr 1, Unreal Engine zazwyczaj wykorzystywał procesor w większym stopniu niż Unity, natomiast na stanowisku drugim, Unreal Engine zawsze korzystał z mniejszej mocy procesora.



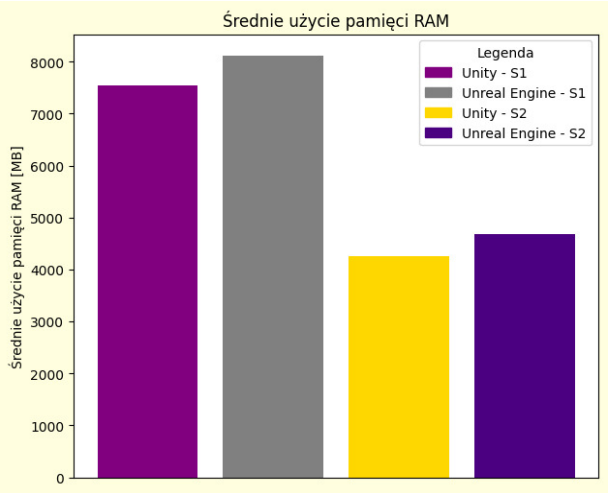
Rysunek 9: Wykresy liniowe testów poruszania dla stanowiska nr 1.



Rysunek 10: Wykresy liniowe testów poruszania dla stanowiska nr 2.

4.3. Analiza wydajności silników gier pod kątem użycia pamięci RAM

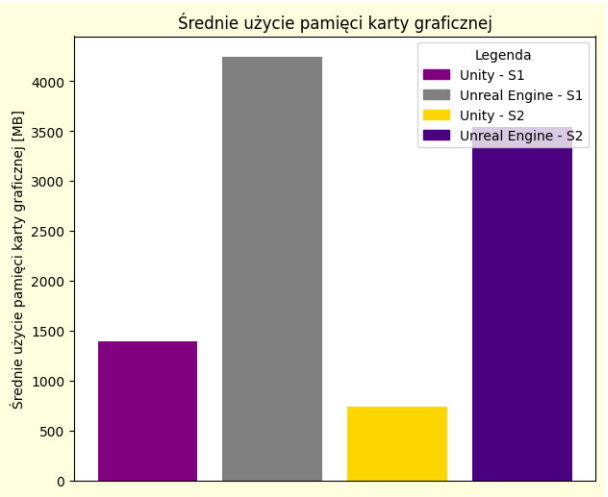
Pamięć RAM (ang. Random Access Memory) jest jednym z kluczowych komponentów w komputerach. Stanowi ona miejsce przechowywania danych aktualnie wykonywanych programów. Rysunek 11 przedstawia średnie użycie pamięci RAM przez stworzone gry. Wartości na wykresach zostały wyliczone na podstawie średnich ze wszystkich badań, ponieważ wartości wszystkich akcji dla danego silnika i stanowiska były podobne. Na stanowisku nr 1 Unreal Engine zużywał o 588 MB więcej pamięci w porównaniu do Unity. W przypadku drugiego stanowiska również wymagał więcej pamięci - o 409 MB. Analiza użycia pamięci RAM jednoznacznie wskazuje na mniejsze średnie użycie dla silnika Unity, aczkolwiek różnica ta nie jest znacząca.



Rysunek 11: Wykres słupkowy dla średniego użycia pamięci RAM.

4.4. Analiza wydajności silników gier pod kątem użycia pamięci karty graficznej

Karta graficzna (GPU) jest elementem komputera odpowiedzialnym za przetwarzanie i renderowanie obrazów oraz grafiki. Karta graficzna jest w stanie obsłużyć intensywne obliczenia związane z grafiką, co odciąża CPU i poprawia wydajność gier i aplikacji.



Rysunek 12: Wykres słupkowy dla średniego użycia pamięci karty graficznej.

Rysunek 12 przedstawia średnie użycie pamięci GPU w testowanych grach. Wartości przedstawione na wykresach wyliczono, podobnie jak w średnim użyciu pamięci RAM, poprzez wyciągnięcie średniej ze wszystkich badań. W przypadku S1 można zauważyć, że silnik Unity wykorzystywał 2844 MB pamięci GPU mniej niż Unreal Engine. Dla stanowiska nr 2 także odnotowano mniejsze o 2796 MB użycie pamięci GPU. Przeprowadzone badania wykazały ponad trzykrotnie większe użycie pamięci GPU przez silnik Unreal Engine.

Tak jak w przypadku analizy wyników dla zużycia pamięci RAM okazało się, że średnie wartości wykorzystania pamięci GPU są bardzo zbliżone dla testów wykonywanych na danej grze, więc wyliczono średnią z poprzednio wyliczonych średnich i wyniki przedsta-

wiono na powyższym wykresie oraz przeanalizowano. Na tej podstawie zauważono bardzo dużą różnicę w użyciu pamięci GPU na korzyść Unity.

5. Wnioski

Na podstawie analizy wydajności silników gier, przeprowadzonej w ramach niniejszej pracy, można wyciągnąć następujące wnioski:

1. Pomiar klatek na sekundę**:**
   a) silnik Unity osiągał wyższe średnie wartości klatek na sekundę przy każdej akcji na obu stanowiskach testowych,
   b) wyniki pomiarów wskazały mniejsze odchylenia standardowe (mniejsze wahania fps) dla gry napisanej w Unreal Engine,
2. Użycie procesora:
   a) na pierwszym stanowisku testowym silnik Unity wykazywał mniejsze użycie procesora w większości akcji, lecz różnica median nigdy nie przekraczała 4%,
   b) w przypadku S1 i akcji poruszania zmierzono wyższe użycie procesora dla Unity, co może wynikać z charakterystyki działania Blueprint'ów w konkurencyjnym silniku,
   c) na drugim stanowisku, przy każdej akcji, Unity wykazywało wyższe użycie procesora,
3. Użycie pamięci RAM:
   a) średnie użycie pamięci RAM było niższe zarówno na pierwszym, jak i drugim stanowisku dla silnika Unity,
   b) różnica w użyciu pamięci RAM między silnikami nie była znacząca,
4. Użycie pamięci GPU:
   a) średnie użycie pamięci GPU było niższe zarówno na pierwszym, jak i drugim stanowisku dla silnika Unity,
   b) różnica użycia pamięci GPU była ponad trzykrotnie wyższa dla Unreal Engine.

Wyniki badań częściowo potwierdzają postawioną we wstępie tezę, która twierdzi, że Unity wymaga mniej zasobów komputera niż Unreal Engine. Testy pamięci RAM i GPU potwierdzają mniejsze zapotrzebowanie na te zasoby przez silnik Unity. Jednak analiza użycia procesora wykazała, że na stanowisku nr 2 Unity wymagało większej mocy procesora. Dane zebrane podczas akcji poruszania na stanowisku pierwszym również wskazują, na wyższe użycie zasobów CPU przez Unity.

Literatura

[1] H. Żukowski, Comparison of 3D games' efficiency with use of CRYENGINE and Unity game engines, Journal of Computer Sciences Institute 13 (2019) 345–348.

[2] A. M. Barczak, H. Woźniak, Comparative Study on Game Engines, Studia Informatica, Systems and Information Technology 23(1-2) (2020) 5–24.

[3] H. A. J. Al Lawati, The Path of UNITY or the Path of UNREAL? A Comparative Study on Suitability for Game Development, Journal of Student Research (2020) 1–7.

[4] J. Zhang, Implementation and Optimization of Particle Effects based on Unreal Engine 4, Journal of Physics: Conference Series 1575 012187 (2020) 1–7.

[5] A. Pattrasitidecha, Comparison and evaluation of 3D mobile game engines, Department of Computer Science and Engineering Göteborg, Sweden, 2014.

[6] P. Mishra, U. Shrawankar, Comparison between Famous Game Engines and Eminent Games, International Journal of Interactive Multimedia and Artificial Intelligence 4 (2016) 69–77.

[7] P. Skop, Comparison of performance of game engines across various platforms, Journal of Computer Sciences Institute 7 (2018) 116–119.

[8] A. Šmíd, Comparison of unity and unreal engine, Czech Technical University in Prague (2017) 41–61.

[9] E. Christopoulou, S. Xinogalos, Overview and comparative analysis of game engines for desktop and mobile devices, International Journal of Serious Games 4(4) (2017) 21–36.

[10] A. Neppius, 3D Game Texturing: Comparative Analysis Between Hand-painted and PBR pipelines, South-Eastern Finland University of Applied Sciences, Finland, 2022.

[11] A. Andrade, Game engines: A survey, EAI Endorsed Transactions on Serious Games 2(6) (2015) 1–6.

[12] X. Cui, H. Shi, A*-based pathfinding in modern computer games, International Journal of Computer Science and Network Security 11(1) (2011) 125–130.