JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 10 (2019) 54-61
Wysłane: 2018-11-28
Przyjęte: 2018-12-03

Wykorzystanie postprocesingu i jego wpływu na wydajność renderowania w silniku Unreal Engine 4

Eryk Puławski*, Marcin Tokarski
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Dzisiejszą grafikę 3D ciężko wyobrazić sobie bez efektów znanych nam z filmowych produkcji czy wysokobudżetowych gier wideo. Jednak użycie takich upiększeń wiąże się ze zwiększonym zapotrzebowaniem na moc obliczeniową. Żeby sprostać oczekiwaniom deweloperów i konsumentów na rynku aplikacji 3D powstały narzędzia do ich tworzenia w postaci kompleksowych silników. Jednym z tych rozwiązań jest silnik Unreal Engine. W poniższym artykule badano wpływ na wydajność wspomnianych efektów - postprocesów. W tym celu przygotowano testową scenę i dwa scenariusze testów.

Słowa kluczowe: postprocess; Unreal Engine; 3D

*Autor do korespondencji.

Adres e-mail: eryk.pulawski@pollub.edu.pl

The use of postprocessing and its impact on rendering performance in the Unreal Engine 4

Eryk Puławski*, Marcin Tokarski
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Today's 3D graphics are hard to imagine without the effects known to us from film productions or high-budget video games. However, the use of such embellishments is associated with an increased demand for computing power. In order to meet the expectations of developers and consumers on the 3D applications market, tools for their creation in the form of comprehensive engines were created. One of these solutions is the Unreal Engine. The following article examines the impact on the efficiency of these effects - postprocesses. For this purpose, a test stage and two test scenarios were prepared.

Keywords: postprocess; Unreal Engine; 3D

*Corresponding author.

E-mail addresses: eryk.pulawski@pollub.edu.pl

1. Wstęp

Tworzenie grafiki komputerowej 3D rozwija się nieustannie od lat 60-tych [1]. Na urządzeniach domowych zagościła już w końcówce lat 70 na przykład na komputerach Apple II [1]. Tworzenie takiej grafiki możemy podzielić na trzy podstawowe fazy:

- modelowanie – proces nadawania kształtu obiektu w postaci modelu matematycznego na komputerze,
- układ i animację – proces rozmieszczenia obiektów w przestrzeni (scenie) oraz ich poruszanie,
- renderowanie – proces komputerowego wyliczania obrazu 2D (pikseli) na podstawie ustalonych modeli 3D i kamery (projekcji) i np. typów materiałów, światła.

Rynek gier komputerowych przyczynił się do szybszego rozwoju grafiki 3D. Gracze pragnęli coraz to ładniejszej lub realistyczniejszej grafiki, szybkości działania czy dostępności produkcji w 3D. Dlatego też studia deweloperskie czy sami deweloperzy od początku istnienia tej dziedziny prześcigali się technologicznie, tworząc coraz to nowsze rozwiązania, by sprostać oczekiwaniom odbiorców. Jednak wraz z kolejnymi latami rosła też złożoność i skomplikowanie programów generujących grafikę i coraz trudniej było jednostkom nadążyć za trendami. Małe aplikacje renderujące rozrastały się w kompletne silniki. Taka postać rzeczy doprowadziła do sytuacji, w której stworzenie własnego rozwiązania jest nieopłacalne bądź nawet niekiedy niewykonalne. Na takie przedsięwzięcie decydują się już tylko nieliczni, posiadający ogromny kapitał.

Sytuacja na rynku silników 3D zdążyła się już ustabilizować. Część z największych studiów zmieniła podejście i zaoferowała darmowy dostęp do swoich produktów. Większość rynku posiadają silniki Unreal Engine, Unity i CryEngine [2].

W celu utworzenia fotorealistycznej czy ładnej oprawy graficznej kluczowym procesem są efekty nakładane po wstępnej fazie renderowania. Te metody (postprocessing) oferowane są współcześnie przez wszystkie silniki 3D.

Celem artykułu jest zmierzenie wpływu postprocesingu na wydajność renderowania wykorzystując silnik Unreal Engine 4. Analiza wykonywana jest używając narzędzi profilujących układy CPU i GPU skupiając się na czasie renderowania (wykonania) poszczególnych funkcji i obliczeń.

2. Obiekt badań

Badanie skupia się na oferowanych przez Unreal Engine 4 efektach postprocessingu. Silnik zapewnia implementację wielu różnych algorytmów i efektów, jednak w artykule skupiono się na kilku z nich opisanych w podrozdziałach poniżej.

2.1. Unreal Engine 4

Silnik Unreal Engine 4 to kompletny zestaw narzędzi deweloperskich stworzony dla wszystkich pracujących z technologiami 3D. Pozwala na uzyskanie fotorealistycznych obrazów renderowanych w czasie rzeczywistym (generowanych i wyświetlanych w tym samym czasie tworząc wrażenie animacji). Zachowuje przy tym relatywnie wysoką wydajność w porównaniu z podobnymi narzędziami.

Każdy deweloper korzystający z tego silnika ma pełny dostęp do kodu źródłowego, który może dowolnie modyfikować oraz dodawać nowe funkcjonalności wykorzystując język C++. Programowanie wizualne za pomocą szkiców, zwanych blueprintami, w Unreal Engine 4 to elastyczny i potężny interfejs oparty na węzłach do tworzenia elementów rozgrywki [3]. Ten interfejs zapewnia projektantom i artystom możliwość programowania ich własnych gier wewnątrz edytora nie pisząc ani jednej linii kodu.

Blueprinty skonstruowane są na zasadzie diagramów, które zawierają liczne węzły połączone ze sobą. Połączenia te definiują ich działanie.

2.2. Post-processing

Nazwa post-processing używana jest w branżach wideo jako metody ogólnego przetwarzania obrazu w celu polepszenia jakości, ale także w renderingu 3D jako dodatkowe efekty, które można uzyskać już po zasadniczej fazie renderowania.

W przypadku gier wideo zamiast renderowania sceny bezpośrednio na ekran, obiekty najpierw renderowane są do bufora znajdującego się w pamięci karty graficznej. Następnie shadery (pixel i czasami vertex) używane są do zastosowania filtrów na obrazie w buforze. Dopiero wtedy następuje wyrenderowanie na docelowy ekran. Niektóre z postprocesów wymagają jednego “przejścia” po obrazie, inne zaś wielokrotnego. Post-processing pozwala na wykorzystanie takich efektów, które muszą mieć informacje o całym obrazie (standardowo obiekty renderowane są w izolacji od siebie).

W branży gier używa się kilkudziesięciu zdefiniowanych efektów, jednak ich liczba jest nieograniczona i każdy może wymyślić i stworzyć swoje własne. Do najpopularniejszych zaliczają się: Anti-Aliasing, Auto-Exposure, Blending, Color Grading, Depth of Field, Lens Flare, Panini Projection, Fog, Dithering, Texture Filtering, Bloom.

2.3. Antyaliasing

Antyaliasing jest to technika zmniejszania efektów schodkowania obrazu. W renderingu rezultatem tego działania jest wrażenie gładkich krawędzi wyświetlanych na ekranie rastrowym. Problem aliasingu dotyczy krzywych, pionowe i poziome linie są renderowane bez potrzeby wygładzania. W grafice wektorowej i na odpowiednich ekranach efekt schodkowania nie występuje i nie zachodzi potrzeba wykorzystywania tego efektu.

Antyaliasing w uproszczeniu może polegać na zmianie koloru pikseli bezpośrednio sąsiadujących z krzywą na kolor proporcjonalny do ich odległości od tej krzywej. Taka linia obiektu będzie wydawać się gładka, jednak stwarzała będzie wrażenie rozmazanej. Przykład efektu antyaliasingu widoczny jest na Rys. 1.

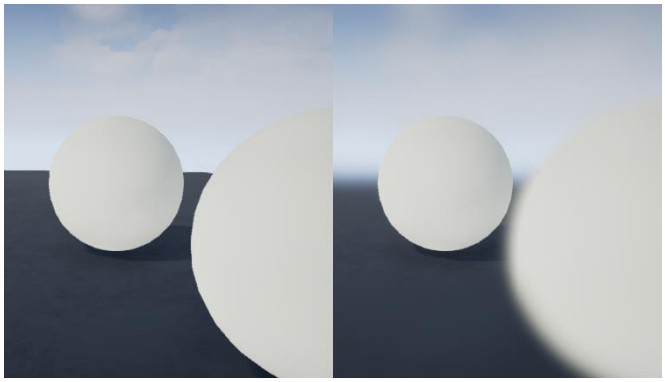
Rys. 1. Efekt antyaliasingu – wył./wł

2.4. Depth of field

W optyce głębia ostrości (ang. depth of field) jest to zakres odległości, w którym wszystkie obserwowane obiekty wydają się być ostre. Duża głębia ostrości charakteryzuje się tym, że większość elementów na zdjęciu jest ostra, natomiast mała głębia najczęściej spotykana jest podczas fotografii portretowych eksponując jeden obiekt [4].

Depth of Field w silniku Unreal Engine 4 nakłada rozmycie na obiekty, które są w określonej odległości przed lub za punktem centralnym. Ten efekt używany jest aby skupić uwagę obserwatora w jednym miejscu i sprawić żeby wyrenderowana scena przypominała fotografię lub film. Na Rys. 2 znajduje się zrzut ekranu z widocznym efektem DoF. Silnik zapewnia trzy metody do nakładania tego efektu:

- Bokeh DoF,
- Gaussian DoF,
- Circle DoF.



Rys. 2. Efekt Depth of Field – wył./wł

2.5. Lens flare

W optyce flara (ang. lens flare) odnosi się do zjawiska, w którym światło jest rozpraszane w systemie soczewek, gdy obiektyw skierowany jest na jasny punkt. Efektem wizualnym tego zjawiska są najczęściej różnokolorowe, przeźroczyste okręgi. Obiektywy z dużą liczbą elementów, takich jak zoomy, wykazują większą flarę.

We wszelkiego rodzaju produkcjach filmów animowanych, generowanych komputerowo efektach specjalnych oraz grach komputerowych powszechnie wykorzystywany jest efekt Lens Flare [5]. To technika oparta na obrazie, która symuluje rozpraszanie światła podczas oglądania jasnych obiektów z powodu niedoskonałości obiektywów aparatu. Ten efekt zobrazowany jest na Rys. 3.

Rys. 3. Efekt Lens Flare – górwa wył., dół wł

2.6. Bloom

Jest to efekt grafiki komputerowej wykorzystywany w grach wideo do odtworzenia artefaktu widzialnego na obrazie z rzeczywistych kamer. Wynikiem tego efektu jest obraz z obszarem jasnego światła, który wydaje się rozlewać poza jego faktyczne ramy. Tworzą w ten sposób iluzję wyjątkowo jasnego światła, które przytłacza aparat lub oko.

W silniku Unreal Engine 4 do uzyskania efektu Bloom wykorzystywane jest rozmycie Gaussa. Dla lepszej jakości takiego efektu łączonych jest kilka takich rozmyć z różnymi promieniami [6]. Przykład takiego efektu widoczny jast na Rys. 4.

Rys. 4. Efekt Bloom – wył./wł

3. Metodyka badawcza

W celu poprawnego zmierzenia wpływu postprocessingu na wydajność, wyniki pomiarów czasów nakładania poszczególnych efektów należy przedstawić w milisekundach, ze względu na:

- pomiary i porównywanie – ciężko porównać poszczególne efekty w FPS (ang. Frames per second);
- FPS jest końcowym wynikiem – nie nadaje się do mierzenia pojedynczych funkcji, jest odpowiedni do całościowych testów wydajności;
- trudność w wyrażeniu kosztu efektu w FPS;
- koszt niektórych efektów jest stały.

3.1. Platforma testowa

Wpływ postprocessingu na wydajność zbadany został na dwóch komputerach klasy PC działających na systemie Windows 8.1. Ich główne podzespoły ukazane zostały w Tabeli 1.

Tabela 1. Użyty sprzęt do testów

Lp.	CPU	GPU	RAM	Typ dysku
1.	i5-2500k 4 x 4,5GHz	GTX 960 4GB	8GB	SSD
2.	Pentium G4560 4 x 3,5GHz	Radeon R7 200 Series 2GB	8GB	HDD

Specyfikacja pierwszego komputera odpowiada mniej więcej przeciętnemu sprzętowi gracza w 2017-2018 roku, który określony jest w Tabeli 2. Drugą natomiast sklasyfikować można poniżej przeciętnej ze względu na jednostkę procesora graficznego.

Tabela 2. Najpopularniejsze podzespoły graczy na podstawie ankiety Steam [7]

Element	Most popular	Percentage
GPU	NVIDIA GeForce GTX 1060	11.89%
	NVIDIA GeForce GTX 1050 Ti	8.06%
	NVIDIA GeForce GTX 960	5.16%
VRAM	1024 MB	24.92%
	2047 MB	23.57%
	4095 MB	17.43%
RAM	8 GB	38.97%
	12 GB and higher	36.67%
	4 GB	11.51%
CPU speed	3.3 Ghz to 3.69 Ghz	22.75%
	3.0 Ghz to 3.29 Ghz	18.05%
	2.3 Ghz to 2.69 Ghz	15.31%
Physical CPUs	4 cpus	60.51%
	2 cpus	31.40%
	6 cpus	4.15%

Wykorzystane podzespoły nie są obecnie najwydajniejszymi podzespołami na rynku, jednak ich wydajność nie wpływa na wnioski z wyników badań.

3.2. Aplikacja testowa

W celu przetestowania wpływu wydajności przygotowano przykładową scenę przedstawiającą fragment lasu. Wykorzystane assety (modele 3D, aktorzy, tekstury itp.) przedstawiono w Tabeli 3.

Tabela 3. Liczebność assetów i ich typy

Nazwa	Typ	Liczebność
Kwiaty – Jaskier	Foliage, StaticMesh	312
Kępa trawy		488
Kwiaty - Krwawnik		65
Mirt bagienny	StaticMeshActor	9
Drzewa		14
Wielka skała - Wulkaniczna		2
Średni głaz		6
Platforma	StaticMeshActor	1
Martwe liście		39
Paproć		4
Gwiazdnik		6
Podłużna skała		9
Pionowa skała		2
Równinny głaz		2
Skała górska		27
Skała rzeczna		1
Kłoda		1
Pień		1
Światło puntowe	PointLight	6
Sfera odbić	SphereReflectionCapture	1
Kierunek wiatru	WindDirectionalSource	1

Na utworzoną scenę zostało nałożonych większość efektów postprocesowych, które oferuje silnik Unreal Engine. Łączna liczba trójkątów siatki wszystkich umieszczonych modeli na scenie wynosi 1761725. Umieszczone assety na scenie tworzą teren o wymiarach około: wysokość 50m, szerokość 30m, długość 30m.

3.3. Narzędzia pomiarowe

Profilowanie to mierzenie czasu potrzebnego do wykonania bloku instrukcji lub funkcji np. renderowania obiektów przeźroczystych. Do zmierzenia tego wpływu wykorzystać można wbudowane w Unreal Engine narzędzie GPU Visualizer oraz Session Fronted. Unreal umożliwia zmierzenie czasów wykonania poszczególnych etapów renderowania poprzez umieszczanie znaczników czasu (GPU Timestamps). Profilowanie przydatne jest podczas tworzenia każdej aplikacji 3D, ze względu na możliwość:

- porównania różnych rozwiązań;
- znalezienia wąskich gardeł;
- uniknięcia często niepotrzebnej, wczesnej optymalizacji.

GPU Visualizer pokazuje statystyki podzielone na kilka kategorii takich jak ShadowDepths (tworzenie mapy cieni), czy Lights (obliczenia światła). Jego jedynym wyraźnym ograniczeniem jest to, że nie zapewnia statystyk dla specyficznych obiektów lub niektórych świateł (niezwiązanych z cieniem). Żeby wywołać okno Visualizera można skorzystać ze skrótu klawiszowego ”Ctrl + Shift + ,”, w którym to ukazany jest przekrój pojedynczej klatki obliczanej przez GPU w postaci wykresu segmentowego pokazującego poszczególne kategorie oraz bardziej dokładnej listy kategorii wraz z ich efektami w scenie.

Narzędzie Session Frontend jest stworzone do uproszczania i przyspieszenia procesu tworzenia gier komputerowych. Pozwala na zdalne monitorowanie aktualnie aktywnych sesji gier. Wykorzystywane jest do zidentyfikowania możliwych źródeł spowolnień w grze.

3.4. Scenariusze testowe

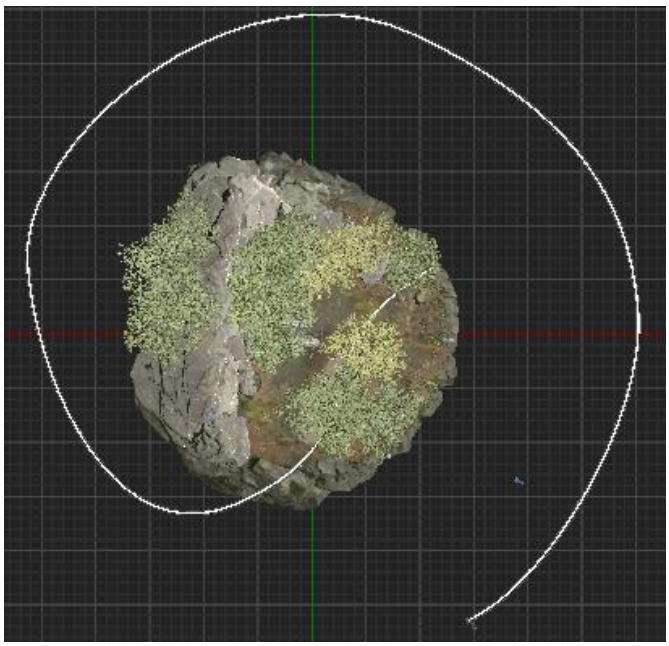
Przygotowane zostały dwa odrębne scenariusze testowe do przeprowadzenia na platformach testowych. Pierwszy nakierowany jest na zbadanie wpływu poszczególnych postprocesów na wydajność. Drugi zaś skupia się na zmierzeniu wpływu wszystkich efektów postprocesowych na długość renderowania pojedynczej klatki.

Na scenariusz nr 1 składają się następujące kroki:

1) rozstawienie kamer w różnych miejscach na scenie,
2) ustawienie aktualnego widoku na widok z n-tej kamery,
3) ustawianie rozdzielczości okna,
4) uruchomienie sceny w nowym oknie,
5) uruchomienie GPU Visualizer – pomiar z 1 klatki,
6) powtórzyć kroki od punktu 2 do 5 dla wszystkich badanych kamer i rozdzielczości,
7) zebranie danych do analizy.

W przeciwieństwie do poprzedniego testu – w którym badane są poszczególne klatki z różnych kamer – następny

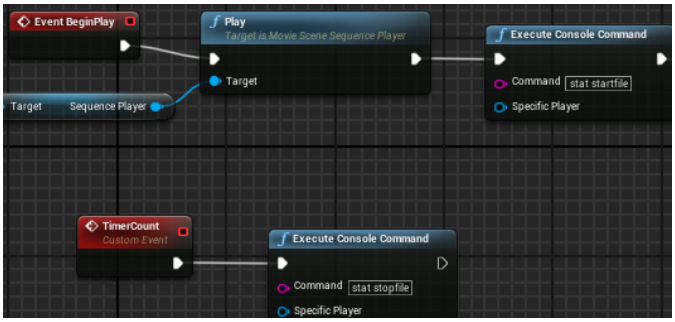
test obejmuje ciągły zapis danych przez czas działania aplikacji. W trakcie testu kamera porusza się dynamicznie po przygotowanym torze wokół sceny. Tor ruchu kamery widoczny jest na Rys. 5. Zebrane dane są w postaci pliku o rozszerzeniu UE4STATS zawierającego tekstowe wyniki dla poszczególnych klatek. Interpretacja rezultatów możliwa jest np. za pomocą narzędzia Session Frontend.



Rys. 5. Platforma testowa do scenariusza nr 2

Kroki scenariusza nr 2 przedstawiają się następująco:

1) ustawienie aktualnego widoku na widok z kamery umieszczonej na szynach,
2) ustawienie rozdzielczości okna,
3) uruchomienie sceny w nowym oknie,
4) powtórzyć poprzednie kroki dla wszystkich badanych rozdzielczości,
5) analiza plików z wynikami.



Rys. 6. Zrzut ekranu z Level Blueprint

Na Rys. 6 widoczny jest fragment blueprinta wykorzystanego w drugim scenariuszu, który po uruchomieniu aplikacji rozpoczyna zapis danych do analizy z ustalonej sekwencji ruchu kamery i po określonym czasie zatrzymuje logowanie.

4. Wyniki

Do realizacji pierwszego scenariusza wykorzystano cztery testowe kamery. Zrzuty ekranów z widoku tych kamer znajdują się na Rys. 7. Wyniki z pierwszego testu znajdują się w Tabeli 4 i Tabeli 5.

Rys. 7. Widok z czterech testowanych kamer

Wybrano następujące rozdzielczości testowe:

- HD - 1280x720 px,
- FHD - 1920x1080 px.

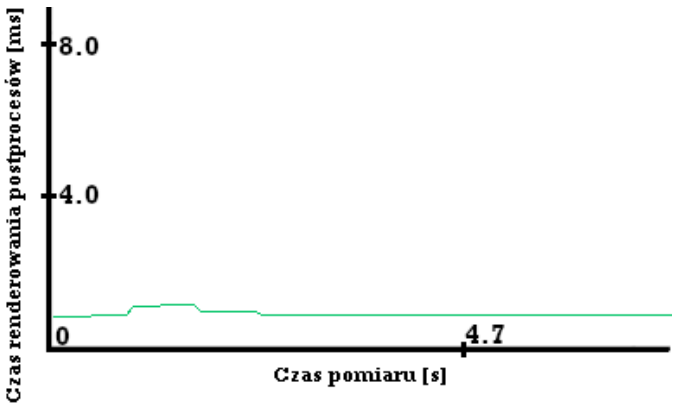
Tabela 4. Czas renderowania w rozdzielczości HD w ms – 1280x720 px

Nazwa procesu	**Kamera**							
	I	II	III	IV	I	II	III	IV
Antyaliasing	0,56	0,54	0,57	0,54	1,55	1,54	1,55	1,55
Bloom	0,03	0,03	0,03	0,03	0,08	0,09	0,09	0,09
Depth of Field	0,31	0,32	0,27	0,32	0,77	0,77	0,77	0,77
Lens Flare	0,16	0,16	0,16	0,16	0,36	0,36	0,36	0,36
Narzut postprocesów	1,56	1,56	1,56	1,6	3,79	3,8	3,79	3,79
Czas wyrenderowania klatki	25,1	26,55	16,8	24,72	46,98	54,36	35,03	48,46
	PC 1				PC2			

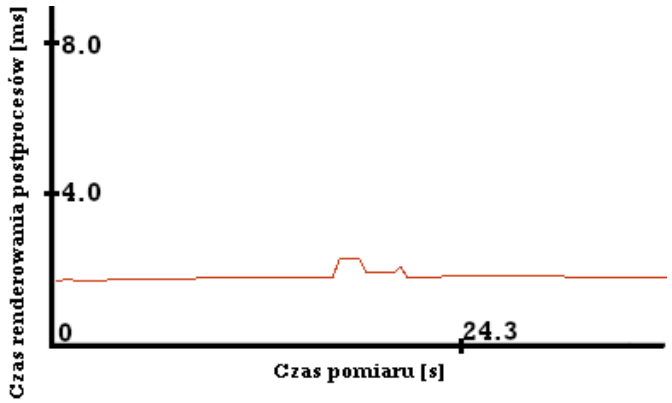
Tabela 5. Czas renderowania w rozdzielczości FHD w ms – 1920x1080 px

Nazwa procesu	Kamera							
	I	II	III	IV	I	II	III	IV
Antyaliasing	0,93	0,93	1	0,93	2	2,03	2,03	2,03
Bloom	0,04	0,04	0,04	0,04	0,12	0,13	0,12	0,12
Depth of Field	0,7	0,69	0,59	0,69	1,29	1,29	1,29	1,29
Lens Flare	0,35	0,35	0,35	0,35	0,79	0,79	0,79	0,79
Narzut postprocesów	3,5	3,53	3,52	3,6	7,12	7,17	7,17	7,15
Czas wyrenderowania klatki	39,88	41,66	25,24	38,88	76,5	82,81	51,28	73,67
	PC 1				PC2			

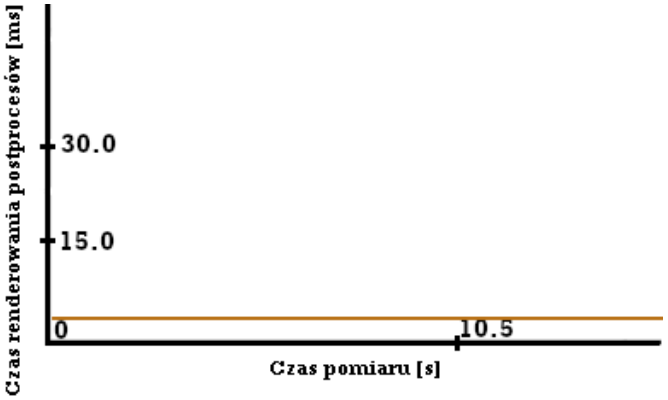
Do realizacji drugiego scenariusza wykorzystano jedną kamerę poruszającą się po określonym torze ruchu. Dane zbierane były przez 25 sekund. Wyniki z drugiego testu znajdują się w Tabeli 6 na podstawie Rys. 8, Rys. 9, Rys. 10 oraz Rys. 11.



Rys. 8. Czasy renderowania postprocesów na PC1 w rozdzielczości HD w ms – 1280x720 px



Rys. 9. Czasy renderowania postprocesów na PC1 w rozdzielczości FHD w ms – 1920x1080 px



Rys. 10. Czasy renderowania postprocesów na PC2 w rozdzielczości HD w ms – 1280x720 px

Rys. 11. Czasy renderowania postprocesów na PC2 w rozdzielczości FHD w ms – 1920x1080 px

Tabela 6. Czasy renderowania poszczególnych klatek - drugi scenariusz

	PC1		PC2	
	HD	FHD	HD	FHD
Min	0,68	1,30	2,95	6,72
Max	1,11	2,28	3,84	8,53
Średnio	0,87	1,74	3,31	7,59

Ilość zgromadzonych danych jest względnie niewielka lecz nie przeszkadzają w zauważeniu pewnych tendencji czy korelacji. Dane te pozwalają na analizę wpływu na wydajność renderowania wybranych postprocesów, która to znajduję się w kolejnym rozdziale.

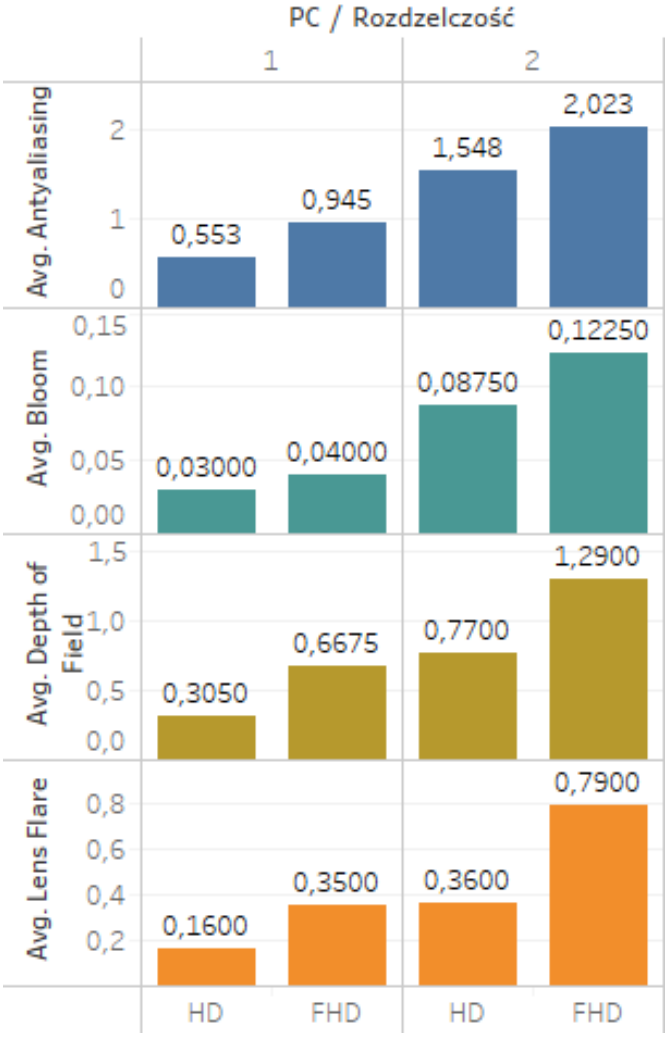
5. Analiza wyników

Dane zebrane są dla 4 wymiarów, z uwagi na chęć zbadania korelacji między nimi. Wymiary te stanowią: sprzęt testowy, rozdzielczość, kamera i postprocesy. Ich liczba powoduje trudność przedstawienia wszystkich wymiarów na jednym wykresie tak aby był on czytelny. Z uwagi na to wyniki przedstawiono na kilku wykresach różnego typu skupiając się na wybranych wymiarach.

Wyniki narzutu postprocesów pokazują, że widok z kamery ma bardzo mały wpływ na czas nakładania procesów. W teorii wpływ *Antyaliasingu* powinien być zauważalny, gdy liczba assetów się zmienia.

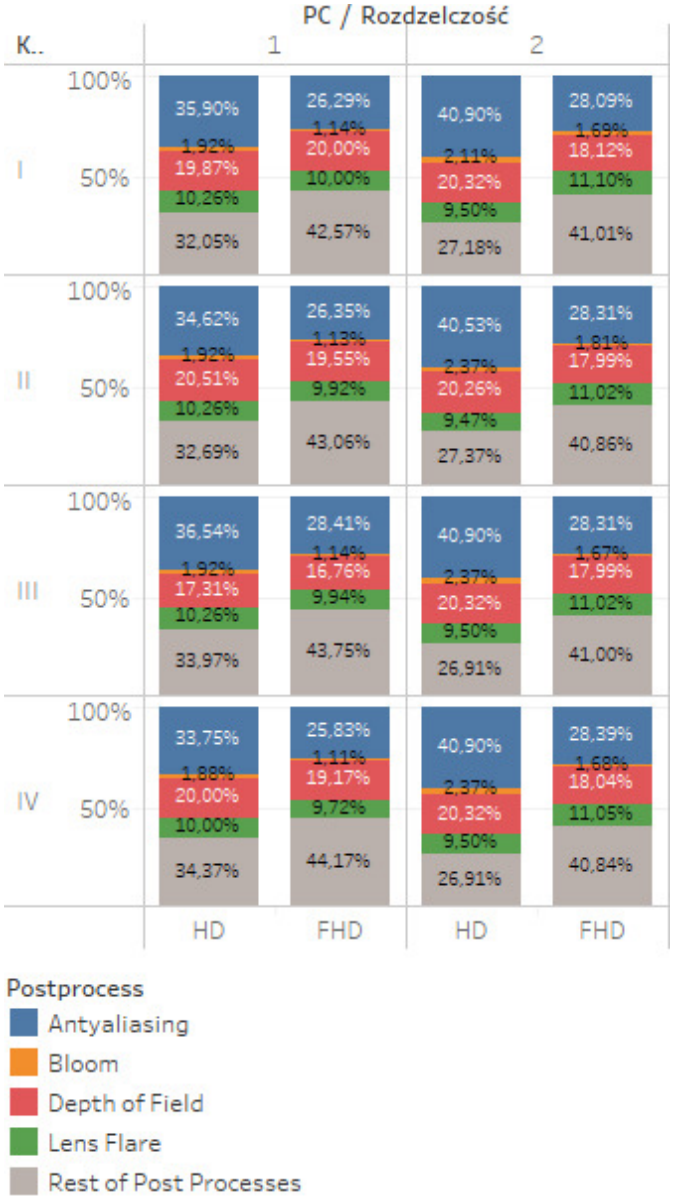
Analizując wyniki dla poszczególnych efektów faktem staje się, że ich wartości rosną wraz ze wzrostem rozdzielczości. Wynika to z liczby pikseli jakie muszą zostać przetworzone dla danego postprocesu.

Przyglądając się wykresom, będących efektem drugiego scenariusza testowego, ukazanych na Rys. 8, Rys. 9, Rys. 10 i Rys. 11 można zauważyć małe zmiany wartości czasu nakładania post procesów przez cały czas trwania. Prezentuje się to prostą linią na wykresach.



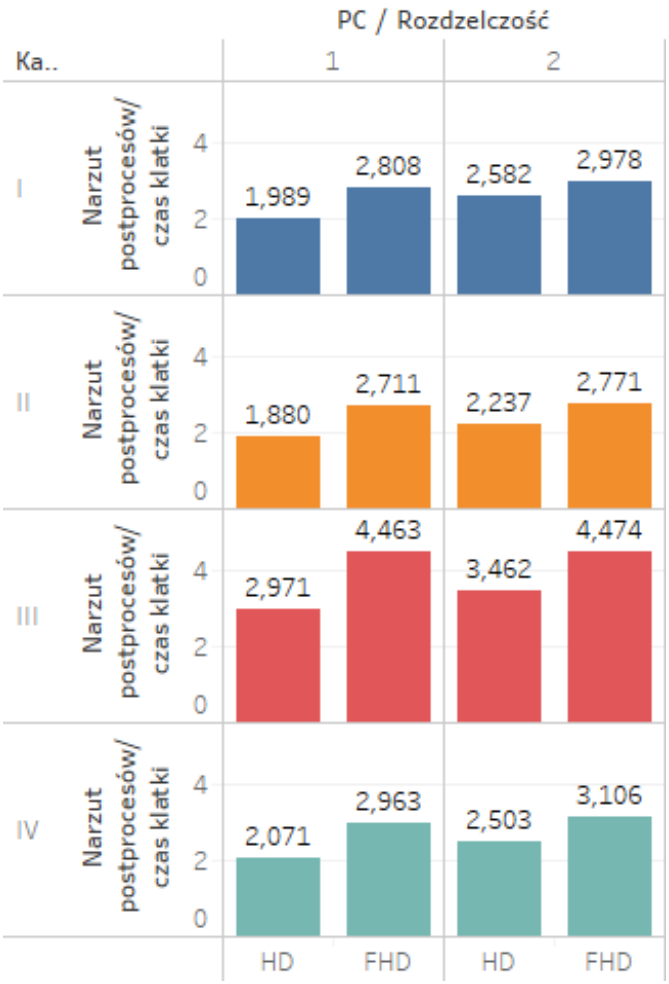
Rys. 12. Średnie czasy [ms] renderowania wybranych postprocesów

Na wykresie Rys. 12 zauważyć można wzrost średniego czasu nakładania każdego z postprocesów wraz ze wzrostem rozdzielczości. Dla *Lens Flare* i *Depth of Field* skok ten był w przybliżeniu dwa razy większy. Pozostałe dwa zanotowały mniejszy wzrost. Świadczy to o tym, że *Antyaliasing* oraz *Bloom* są mniej zależne od rozdzielczości niż pozostałe.



Rys. 13. Procentowy narzut wybranych postprocesów względem czasu renderowania wszystkich

Z wykresu na Rys. 13 można wywnioskować, że niezależnie od kamery, sprzętu testującego i rozdzielczości największy narzut na czas nakładania postprocesów ma *Antyaliasing* a najmniejszy *Bloom*. Dodatkowo tylko czas nakładania efektu *Lens Flare* nie zależy od widoku z kamery a reszta badanych zmienia się nieznacznie.



Rys. 14. Średni narzut postprocesów względem średniego czasu renderowania klatki

W procesie optymalizacji renderowania (zwiększania liczby klatek na sekundę przy zachowaniu tych samych ustawień rozdzielczości) czas poświęcony na postprocesy jest wartością szczególną, ponieważ można przyjąć iż jest stały. Oznacza to, że im więcej klatek na sekundę chcemy osiągnąć, tym większy narzut w całości będzie miał post-processing. Dla przykładu mając scenę o średnim czasie renderowania 30 FPS (co najwyżej 33,(3) ms na klatkę), gdzie post-processing zajmuje 10 ms to jego narzut wynosi około ⅓. Optymalizując scenę by działała w co najmniej 60 FPS (co najwyżej 16,(6) ms na klatkę) postprocesy będą stanowić około ⅝ całego czasu. Analogicznie dla 120 FPS postprocesy stanowią 83% potrzebnego czasu na wyrenderowanie klatki. Łatwo zauważyć, że post-processing jest elementem blokującym w procesie optymalizacji dla aplikacji dążących do działania w dużej liczbie FPS.

6. Wnioski

Na podstawie otrzymanych wyników i analizy z dwóch scenariuszy wyciągnięto następujące wnioski:

- liczba assetów widocznych z kamery nieznacznie wpływa na czas, jaki jest potrzebny na nałożenie postprocesów na poszczególną klatkę;
- im większa rozdzielczość okna tym dłuższy czas nakładania każdego z postprocesów przez GPU;
- czas nakładania postprocesów podczas działania aplikacji oscyluje wokół określonej wartości, generalizując - jest stały;
- *Antyaliasing* oraz *Bloom* są bardziej odporne na wzrost rozdzielczości. Oznacza to niewielki w stosunku do innych postprocesów wzrost czasu nakładania efektu;
- mający największy wpływ na wydajność z badanych postprocesów jest *Antyaliasing*;
- spośród badanych postprocesów najmniejszy wpływ na wydajność wykazał *Bloom*;
- post-processing jest procesem, którego nie można efektywnie zoptymalizować;
- udział postprocesów jest odwrotnie proporcjonalny do czasu renderowania klatki[ms];
- wraz ze wzrostem rozdzielczości udział czasu renderowania postprocesów w czasie całej klatki będzie zwiększał się im wydajniejsza jednostka graficzna.

Literatura

[1] Salustri F. A., „A Brief History of Computer Graphics,” 30 Marzec 2018. [Online]. Available: deseng.ryerson.ca/dokuwiki/mec222:brief_history_of_computer_graphics [25.10.2018].

[2] TNW DEALS, „This engine is dominating the gaming industry right now,” 24 Marzec 2016. [Online]. Available: https://thenextweb.com/gaming/2016/03/24/engine-dominating-gaming-industry-right-now/ [25.10.2018].

[3] Satheesh P.V., Unreal Engine 4 Game Development Essentials, Packt, 2016.

[4] Grey E., „Understanding Depth of Field – A Beginner’s Guide,” 11 Luty 2018. [Online]. Available: https://photographylife.com/what-is-depth-of-field. [22.05.2018].

[5] Underdahl K., Premiere Elements 8 For Dummies, For Dummies, 2009.

[6] Kalogirou C., „How to do good bloom for HDR rendering,” 20 Maj 2006. [Online]. Available: http://kalogirou.net/2006/05/20/how-to-do-good-bloom-for-hdr-rendering/. [22.05.2018].

[7] Steam, „Ankieta dotycząca sprzętu i oprogramowania: September 2018,” [Online]. Available: https://store.steampowered.com/hwsurvey/ [25.10.2018].