Laboratorium Zaawansowanych Technik Przetwarzania Grafiki Komputerowej

Temat: Post-processing

Kamil Tulczyjew
Marcin Tatoń

Wydział Automatyki Elektroniki i Informatyki
Politechnika Śląska
7 maja 2020

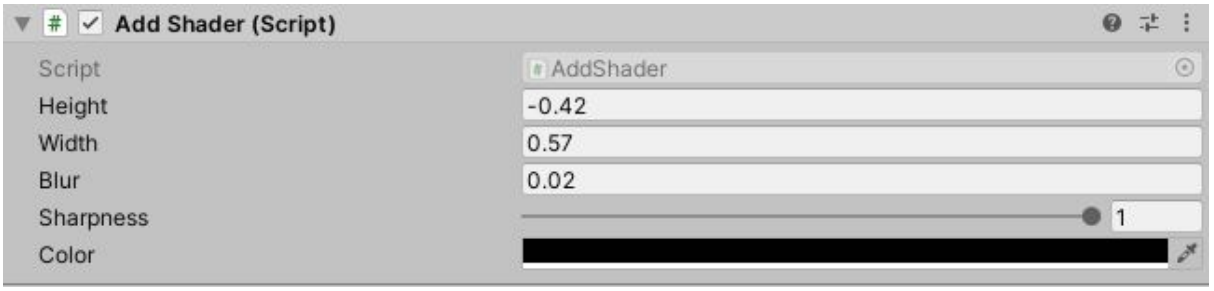
Zadanie
Efekt przebudzenia (otwieraie i zamykanie oczu, zamglenie które zmniejsza się z każdym mrugnięciem i zanika po kilku, zrandomizowane)

Na potrzeby wykonania efektu przebudzania został utworzony:

1. Skrypt shadera **EyeBlink.shader.**
2. Skrypt obsługujący wykonywanie shadera oraz randomizację parametrów do animacji **AddShader.cs.**
3. **Animacja** w silniku Unity, która posłużyła do ustawienia w określonym czasie parametrów shadera w celu symulacji animacji otwierającego się oka przy budzeniu.

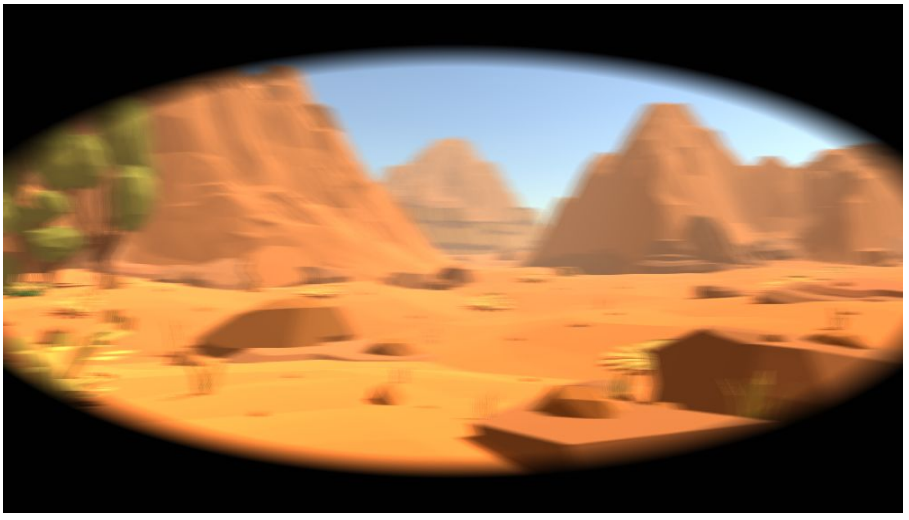
link do repozytorium:
https://github.com/kamitul/eye_blink

Komponent **AddShader.cs**



rys.1

Służy on do przechowywania właściwości shadera i ustawiania ich w konkretnym czasie przez animację. ***Height*** - ustawienie wysokości elipsy powieki, ***Width*** - ustawienie szerokości elipsy powieki. ***Blur*** to parametr odpowiadający za siłę rozmycia obrazu - potrzebne do symulowania zamglenia ekranu. ***Sharpness*** - parametr służący do wyostrzania obwódek powieki, które dla celów wizualnych zostały przez nas dodane oraz ***Color*** - kolor zamkniętej powieki (domyślnie czarny ale umożliwiono możliwość zmiany koloru). Całość efektu przedstawia się następująco (rys.2):



rys.2

Skrypt ten dodatkowo obsługuje randomizowane i losowanie wartości prędkości otwierającego się oka kontrolowanego przez animację. Poprzez zmianę tych prędkości każde nowe uruchomienie aplikacji powoduje pojawienie się lekko zmodyfikowanej animacji otwierającej się powieki pod kątem szybkości jej zamykania i modyfikacji rozmycia obrazu.

Shader **EyeBlink.shader**

rys.3

Shader został zaimplementowany przy użyciu matematycznego równania na elipsę o punkcie środkowym (rys.4).

```
float p = (pow((c.x - center.x), 2) / pow(_width, 2)) + (pow((c.y - center.y), 2) / pow(_height, 2));
```

rys.4

W kodzie odpowiedzialnym za obróbkę fragmentów zachodzi sprawdzenie warunku czy dany punkt znajduje się wewnątrz elipsy czy też poza nią (wskazuje na to wartość zmiennej ***p***) (rys.4). W przypadku gdy punkt znajduje się wewnątrz elipsy następuje jego rozmywanie a w przypadku gdy znajduje się poza elipsą kolor pozostaje taki, jaki został przez użytkownika wybrany w komponencie **AddShader** (rys.1). Ciekawym elementem shadera jest fragment kodu odpowiedzialny z tworzenie “otoczki” powieki (rys.5). Polega on na odpowiednim przeskalowaniu wartości ***p*** do zakresu pomiędzy <0, 1> a następnie wykorzystaniu tej wartości do obsługi “*blendowania”* ze sobą dwóch kolorów w danym punkcie tekstury dostarczonej z kamery. Wartość ***p*** wskazuje na siłę z jaką do łączenia kolorów zostaje pobrany kolor z zewnątrz elipsy a jaką z wewnątrz niej. Dzięki takiemu zabiegowi obraz na obwódkach powiek jest lepszy wizualnie (rys.2).

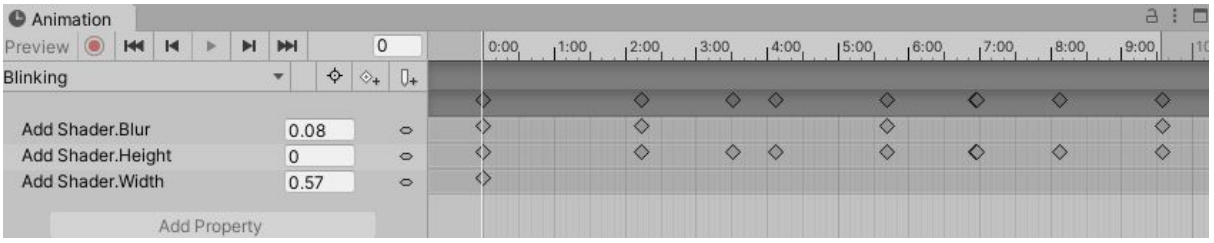
```
if (p < 1)
{
    for (float i = 0; i < 10; i++) {
        blur += tex2D(_MainTex, c + float2((i / 9 - 0.5) * _blurSize, 0));
    }
    blur = blur / 10;

    resultCol = blur;
}

if(p > 0.8 && p < 1)
{
    float div = (1 - p)/ 0.2f;
    p = div * 1 * _sharpness;
    resultCol.r = (i.color.r * (1 - p)) + blur.r * p;
    resultCol.g = (i.color.g * (1 - p)) + blur.g * p;
    resultCol.b = (i.color.b * (1 - p)) + blur.b * p;
    resultCol.a = i.color.a;
}
```

rys.5

Animacja **Blinking.anim**

rys.6

W celu wizualizacji wybudzającego się oka została stworzona animacja ustawiająca parametry shadera w odpowiedniej chwili czasowej. W przypadku budzącego się oka implementacja zmiany wartości parametrów przez skrypt byłaby bardzo męcząca i niewygodna - przy budzeniu się ludzkie oko zachowuje się nieregularnie i dlatego obsługa czasowa otwierania powieki została wprowadzona do animacji (rys.7). Użytkownik może w każdej chwili zmienić wartości szczytowe parametrów shadera w danej jednostce czasu. Elementem randomizującym efekt mrugania jest szybkość parametrów - ustawiana już w skrypcie (rys.8), która zmienia się z każdym uruchomieniem aplikacji oraz mrugnięciem oka.



rys.7

```
private void Update()
{
    if(material.GetFloat("_height") == 0)
    {
        heightSpeed = UnityEngine.Random.Range(0.7f, 1f);
        blurSpeed = UnityEngine.Random.Range(0.25f, 0.6f);
    }
}
```

rys.8