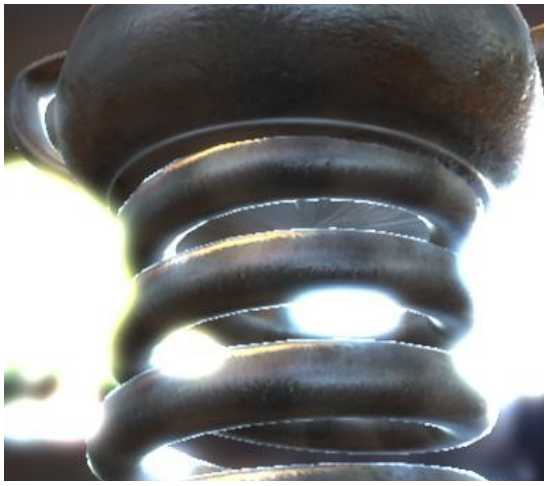
Postprocessing - rozmycie kierunkowe

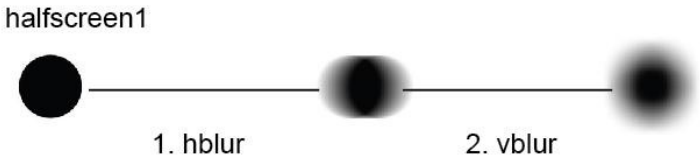
Wcześniej realizowaliśmy rozmycie gaussowskie obrazu – w każdym kierunku obraz rozmywany jest w przybliżeniu w takim samym stopniu. Tym razem będziemy chcieli zsumować rozmycia w zadanych kierunkach, aby uzyskać rozbłyski w kształcie gwiazdek. Dla uproszczenia wykorzystamy tylko dwa kierunki – przekątne kwadratu.



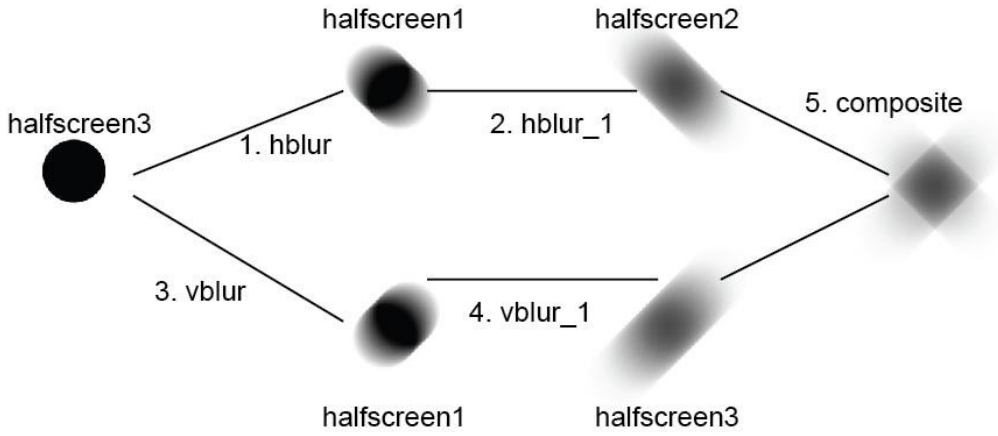
W tym celu zmodyfikujemy nieznacznie sposób realizacji rozmycia. Przebiegi HBlur i VBlur zostaną wykonane dwukrotnie, zmienią się kierunki rozmycia z poziomego i pionowego na skośne oraz zmieni się sposób zapisywania i pobierania wyników pośrednich do i z poszczególnych tekstur pomocniczych.

Na rysunkach poniżej ponumerowane zostały przebiegi filtrowania dla obu przypadków i oznaczone tekstury przechowujące dane źródłowe i wyjściowe dla każdego przebiegu.

A. rozmycie gaussowskie



B. rozmycie w dwóch kierunkach



1. Na początek dla poprawy dokładności obliczeń zmienimy (o ile jeszcze nie zostały zmienione) formaty tekstur screen, halfscreen1, halfscreen2 na 16 bitowe zmiennoprzecinkowe. Nie korzystamy z 32 bitowych liczb zmiennoprzecinkowych nie tylko z powodu większego rozmiaru takich tekstur, ale także dlatego, że z reguły nie jest wspierane sprzętowo ich filtrowanie, co daje dużo gorsze rezultaty na ekranie.

dwuklik na screen, zmień Format na **A16B16G16R16F**, OK, powtórz to samo dla halfscreen1 i halfscreen2

2. Będziemy potrzebowali jeszcze jednej tekstury na wyniki pośrednie – **halfscreen3**, o identycznych parametrach jak halfscreen1 i halfscreen2.

prawy przycisk na Textured Phong, Add Texture, Add Renderable Texture, prawy przycisk na renderTexture3, Rename, halfscreen3, dwuklik na halfscreen3, odznacz Auto-Generate Mip Map, zaznacz Use viewport dimensions, wpisz Width równe 0.5, Height równe 0.5, zmień format na A16B16G16R16F

3. Zmodyfikujemy teraz shadery pikseli w przebiegach HBlur i VBlur tak, żeby filtrowanie odbywało się po skosie, a nie w poziomie i w pionie.

W piksel shaderze w przebiegu HBlur zamień instrukcję
```
tex2D( src, Tex + float2(((i-6)*2.0-0.5)*fTexelSize.x,0.0).xy )
```
na
```
tex2D( src, Tex + ((i-6)*2.0-0.5)*fTexelSize.xy )
```
Podobnie w piksel shaderze w przebiegu VBlur zamień instrukcję
```
tex2D( src, Tex + float2(0.0,((i-6)*2.0-0.5)*fTexelSize.y).xy )
```
na
```
tex2D( src, Tex + float2(-1.0,1.0)* ((i-6)*2.0-0.5)*fTexelSize.xy )
```

4. Dodaj kopię przebiegu HBlur o nazwie **HBlur_1** zaraz po przebiegu HBlur i analogicznie dodamy kopię przebiegu VBlur o nazwie **VBlur_1** (zmiana kolejności przebiegów odbywa się poprzez przeciąganie ich nazwy w odpowiednie miejsce drzewa projektu).

5. Zmienimy teraz powiązania tekstur w przebiegach Downsample, HBlur, HBlur_1, VBlur, VBlur_1, Composite.

a) **Downsample** powinien zapisywać (powiązanie dla render target) do tekstury **halfscreen3**
b) **HBlur** powinien czytać (powiązanie obiektu src z teksturą) z tekstury **halfscreen3** i zapisywać do tekstury **halfscreen1**
c) **HBlur_1** powinien czytać z tekstury **halfscreen1** i zapisywać do tekstury **halfscreen2**
d) **VBlur** powinien czytać z tekstury **halfscreen3** i zapisywać do tekstury **halfscreen1**
e) **VBlur_1** powinien czytać z tekstury **halfscreen1** i zapisywać do tekstury **halfscreen3**
f) **Composite** powinien teraz korzystać z trzech obiektów tekstury src, src2 i **src3** (ten trzeci trzeba utworzyć), przy czym **src2** powinno być powiązane z **halfscreen2**, a **src3** z **halfscreen3**

6. Pozostał do wykonania ostatni element. Zmień treść Pixel Shadera w przebiegu **Composite** na następującą:

```
sampler2D src, src2, src3;

float4 ps_main( float2 Tex : TEXCOORD0 ) : COLOR0
{
   return 0.4*(tex2D(src2,Tex)+tex2D(src3,Tex))+tex2D(src,Tex);
}
```

7. Gotowe. Można teraz zmniejszyć parametr blurscale do 0.8 i popatrzeć przez sprężynę na światło wpadające przez okna.