Projekt 3. Interaktywna woda.

1. Projekt jest indywidualny.
2. Termin oddania projektu to 29. maja.
3. Rozwiązanie należy pokazać na laboratoriach lub przesłać spakowany kod źródłowy razem z wykorzystywanymi danymi na adres P.Aszklar@mini.pw.edu.pl
4. Projekt można pisać w C# lub C++ korzystając z DirectX lub OpenGL i języka shaderów HLSL, GLSL lub Cg.
5. Zadanie składa się z kilku części. Nie trzeba realizować wszystkich.

Opis zadania:

Część I [5 pkt]

a) Symulowanie rozchodzenia się zaburzeń na powierzchni wody. Na kwadratowej siatce punktów należy animować wysokość w każdym z nich, rozwiązując w sposób przybliżony równanie różniczkowe cząstkowe zgodnie z opisem zamieszczonym na końcu dokumentu.
b) Zaburzenia powierzchni wody przypominające padające krople deszczu odpowiadają zaburzeniom wysokości w losowo wybranym punkcie siatki. W każdej klatce z pewnym prawdopodobieństwem należy zdecydować, czy kropla spadnie, czy nie. Jeżeli wynik losowania jest pozytywny, należy w pewnym miejscu zaburzyć wysokość.
c) W każdej klatce animacji, dla siatki wysokości, policzyć normalne w poszczególnych punktach i odpowiednio zapisać do tekstury w formacie RGBA, 8 bitów na składową koloru.
d) Wyświetlać animowaną wodę w postaci pojedynczego kwadratu cieniowanego przy pomocy shadera pikseli z mapowaniem wypukłości (bump mapping). Normalne w każdym punkcie kwadratu należy pobrać z tekstury, o której była mowa w punkcie poprzednim. Mając normalną wystarczy wykonać obliczenia oświetlenia zgodnie z modelem Phonga dla pojedynczego źródła światła.

Część II [+5 pkt]

a) Tym razem kwadrat reprezentujący powierzchnię wody jest przekrojem sześcianu. Należy przygotować dowolną mapę sześcienną i użyć jej do oteksturowania wnętrza sześcianu.
b) Zmodyfikować shader z pierwszej części tak, aby tym razem liczony był promień odbity i załamany, potem ich przecięcia z sześcianem, a kolory pobrane na podstawie punktów przecięć z sześcianem były mieszane na podstawie przybliżenia współczynnika Fresnela (jak w zadaniu 5 z Rendermonkey).

Część III [+3 pkt]

a) Generować krzywą sklejaną (B-Spline) dla równoodległych węzłów, parametryzowaną czasem, z losowymi punktami kontrolnymi odpowiadającymi położeniom w obrębie kwadratu wody.
b) Użyć siatki duck.txt oraz tekstury ducktex.jpg (do pobrania na stronie) do animacji kaczki pływającej zawsze stycznie do splajna. Zero układu lokalnego kaczki znajduje się mniej więcej na środku dolnej ściany prostopadłościanu otaczającego. Kaczka zwrócona jest w kierunku

ujemnym osi x. Wymiary kaczki to mniej więcej 200×100×70 jednostek. Format siatki jest następujący:

- W pierwszej linii liczba wierzchołków V
- W kolejnych V liniach: pozycja wierzchołka (3 floaty), normalna (3 floaty), współrzędne tekstury (2 floaty)
- W kolejnej linii liczba trójkątów T
- W kolejnych T liniach: trójka indeksów wierzchołków (numeracja rozpoczyna się od zera)

c) Zaburzać w każdej klatce animacji powierzchnię wody w punkcie, gdzie znajduje się kaczka.

Część IV [+3 pkt]

a) Należy poszukać informacji na temat shaderów oświetlenia anizotropowego, napisać taki shader i użyć go do cieniowania kaczki dla pewnego ustalonego źródła światła w scenie.

W razie problemów z wyobrażeniem jak ma wyglądać gotowa symulacja, można obejrzeć gotowy efekt pobierając przykładowy program ze strony www.

Interaktywna symulacja powierzchni wody.

Opis bazuje na artykule Miguela Gomeza pod tym samym tytułem, zamieszczonym w „Perełkach Programowania Gier Tom 1”.

Równanie struny wygląda następująco:

$$z_{tt} = c^2 z_{xx}$$

, gdzie $z(t, x)$ jest wychyleniem struny w punkcie x w czasie t, a c to prędkość rozchodzenia się fali (zaburzeń).

Dla membrany analogiczne równanie wygląda tak:

$$z_{tt} = c^2 \Delta z$$

, gdzie

$$\Delta z = z_{xx} + z_{yy}$$

to laplasjan funkcji $z(t, x, y)$.

Powyższe równania różniczkowe cząstkowe są liniowe i hiperboliczne, co oznacza, że można wyznaczyć, dla zadanych warunków początkowych i ewentualnie brzegowych, ich rozwiązanie analityczne, jednak tym nie będziemy się zajmowali.

Będziemy symulować powierzchnię wody jako membranę ograniczoną do kwadratu z nałożonymi warunkami brzegowymi. Cały kwadrat podzielimy na siatkę $N \times N$ punktów. Przyjmijmy $N = 256$. Niech oprócz prędkości rozchodzenia się fali c będzie dana odległość między sąsiednimi punkami siatki w wierszu (kolumnie) h oraz krok całkowania Δt.

Przybliżając pochodne cząstkowe drugiego rzędu przy pomocy różnic dzielonych otrzymamy przybliżoną postać równania membrany:

$$\frac{z_{i,j}^{n+1} - 2z_{i,j}^{n} + z_{i,j}^{n-1}}{\Delta t} = c^2 \left(\frac{z_{i+1,j}^{n} + z_{i-1,j}^{n} + z_{i,j-1}^{n} + z_{i,j+1}^{n} - 4z_{i,j}^{n}}{h^2} \right)$$

, gdzie $z_{i,j}^{n}$ oznacza wartość wysokości w n-tym kroku symulacji (każdy kolejny krok symulacji liczony jest dla chwili odległej o Δt od czasu poprzedniego kroku) dla punktu siatki w i-tym wierszu i j-tej kolumnie.

Z powyższej nierówności możemy już wyprowadzić wzór na $z_{i,j}^{n+1}$ w zależności od wysokości punktów siatki w poprzednim n-tym kroku i wartości $z_{i,j}^{n-1}$. Wynik będziemy jeszcze mnożyć przez współczynnik tłumienia $d_{i,j}$ ustalany dla każdego punktu siatki tylko raz na początku algorytmu. Otrzymujemy:

$$z_{i,j}^{n+1} = d_{i,j}\left(A\left(z_{i+1,j}^{n} + z_{i-1,j}^{n} + z_{i,j-1}^{n} + z_{i,j+1}^{n}\right) + B z_{i,j}^{n} - z_{i,j}^{n-1}\right)$$

, gdzie

$$A = \frac{c^2 \Delta t^2}{h^2}, B = 2 - 4A$$

Do przeprowadzenia symulacji potrzebne są dwie tablice dwuwymiarowe wartości wysokości oraz jedna wartość współczynnika tłumienia. W każdym kroku jedna z tablic wysokości jest wyliczana na podstawie swojej poprzedniej zawartości i drugiej tablicy, po czym obie tablice zamieniają się rolami. Po wyliczeniu nowych wysokości należy jeszcze policzyć normalne (znormalizować iloczyn wektorowy różnic dzielonych wysokości na siatce), zakodować ich składowe w bajtach i wynik zapisać do tekstury. Mając gotową teksturę należy renderować pojedynczy kwadrat, korzystając z shaderów do pocieniowania jego powierzchni z wykorzystaniem mapowania wypukłości.

Powyższa metoda całkowana staje się niestabilna dla zbyt dużych wartości Δt. Miguel Gomez podaje $\frac{c^2 \Delta t^2}{h^2} \leq \frac{1}{2}$ jako warunek stabilności. Poniżej podane są stałe, które zapewniają dobra działanie algorytmu:

$$N = 256,$$

$$h = \frac{2}{N - 1},$$

$$c = 1,$$

$$\Delta t = \frac{1}{N}$$

Współczynniki tłumienia niech będą dobrane w ten sposób, że

$$d_{i,j} = 0.95\, min\left(1, \frac{l}{0.2}\right)$$

, gdzie l to odległość (w metryce maksimum) od najbliższego brzegu kwadratu (bok kwadratu równy 2).

Efekt kropel deszczu lub śladu na wodzie pozostawianego przez kaczkę można uzyskać zaburzając (np. o 0.25) wartość wysokości w pojedynczym punkcie siatki.