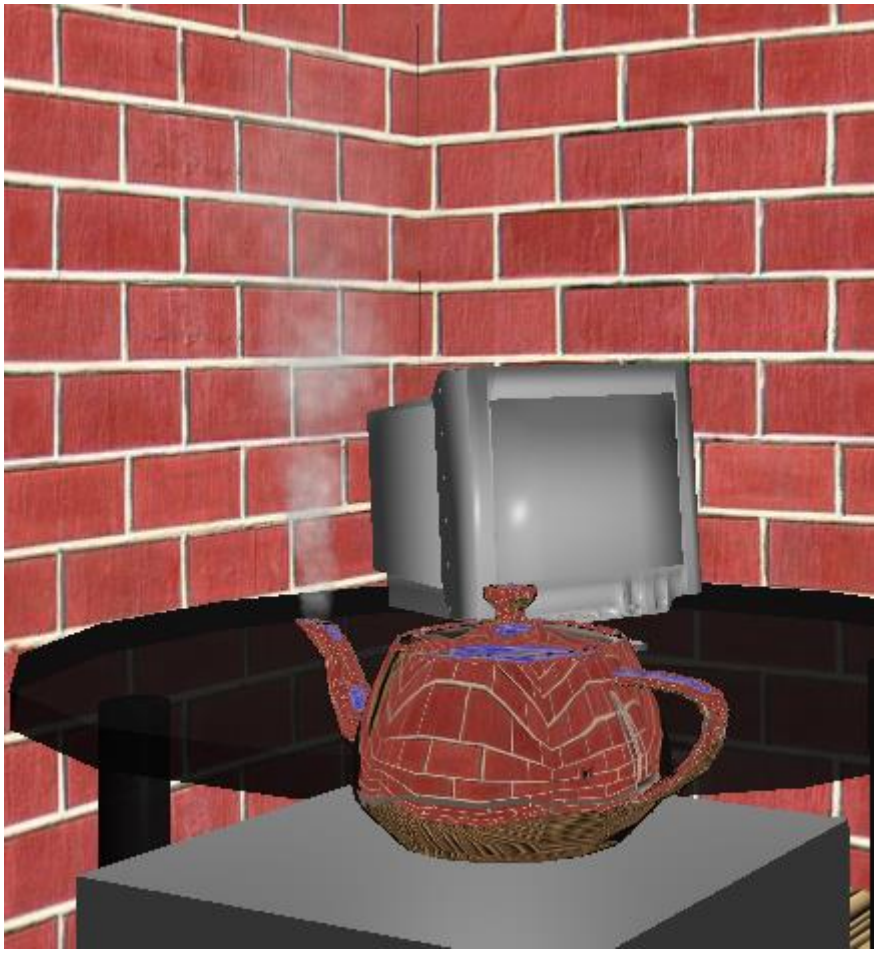
Zadanie 3. System cząstek

Zadanie

Należy wyświetlić animowany system cząstek reprezentujący dym wylatujący z czajnika. Trzeba zaktualizować funkcje „AddNewParticle”, „UpdateParticle”, „UpdateVertexBuffer” oraz „Update” w pliku „gk2_particles.cpp”, a także shader geometrii w pliku „particlesGS.hlsl”. Parametry systemu cząsteczek oraz użyte tekstury pochodzą z przykładu dołączonego do biblioteki OGRE (http://www.ogre3d.org/).



1. Tworzenie i usuwanie cząsteczek

W programie zdefiniowano następujące parametry systemu cząsteczek, które wyznaczają sposób tworzenia nowych i usuwania istniejących cząsteczek

```
static const XMFLOAT3 EMITTER_DIR;        //mean direction of particles' velocity
static const float TIME_TO_LIVE; //time of particle's life in seconds
static const float EMISSION_RATE;//number of particles to be born per second
static const float MAX_ANGLE;    //maximal angle declination from mean direction
static const float MIN_VELOCITY; //minimal value of particle's velocity
static const float MAX_VELOCITY; //maximal value of particle's velocity
static const float PARTICLE_SIZE;//initial size of a particle
static const float PARTICLE_SCALE;        //size += size*scale*dtime
static const float MIN_ANGLE_VEL;//minimal rotation speed
static const float MAX_ANGLE_VEL;//maximal rotation speed
static const int MAX_PARTICLES;  //maximal number of particles in the system
```

Nowo tworzonej cząsteczce należy nadać początkowe wartości wszystkich pól:

- Age – wartość 0.
- Size – wartość PARTICLE_SIZE.
- Velocity – losowany jest najpierw kierunek, który ze średnim kierunkiem emisji cząsteczek (EMITTER_DIR) tworzy kąt nie większy niż MAX_ANGLE, a następnie mnożymy przez skalar wylosowany z przedziału [MIN_VELOCITY; MAX_VELOCITY] (funkcja RandomVelocity).
- Pos – ustalona pozycja źródła cząsteczek (m_emitterPos).
- Angle – wartość 0.
- AngleVelocity – wartość wylosowana z przedziału [MIN_ANGLE_VEL; MAX_ANGLE_VEL].

W każdej klatce animacji należy tworzyć tyle cząstek, aby powstawało średnio EMISSION_RATE cząsteczek na sekundę (można posłużyć się polem m_particlesToCreate, zwiększając je z każdą klatką o EMISSION_RATE pomnożony przez upływ czasu). Nie należy też tworzyć nowych, jeśli liczba wszystkich cząsteczek jest równa MAX_PARTICLES.

Usuwać należy cząsteczki, których wartość pola Age przekroczyła TIME_TO_LIVE.

2. Aktualizacja parametrów pojedynczej cząsteczki.

W każdej klatce animacji należy uaktualnić wiek, położenie, kąt obrotu i wielkość cząstki.

- Age – zwiększa się o upływ czasu dt.
- Pos – zwiększa się o prędkość Velocity pomnożoną przez upływ czasu dt.
- Angle – zwiększa się o prędkość AngleVelocity pomnożoną przez upływ czasu dt.
- Size – zwiększa się o PARTICLE_SCALE * PARTICLE_SIZE*dt.

Pozostałe parametry są stałe.

3. Aktualizacja całego systemu cząsteczek.

Składa się ona kolejno z trzech etapów omówionych powyżej

- Aktualizacji parametrów istniejących cząsteczek

- Usunięcia cząsteczek, których czas życia się skończył
- Dodaniu odpowiedniej liczby nowych cząstek

4. Wyświetlanie

Cząsteczki wyświetlanie są jako bilbordy, obrócone w płaszczyźnie ekranu o pewien kąt. Przed ich narysowaniem, należy je posortować w kolejności od najdalszych do najbliższych kamery. Ma to związek z przezroczystością, aby scena wyświetlona była poprawne, obiekty półprzezroczyste znajdujące się dalej od kamery musza być wyświetlone wcześniej. Po przekopiowaniu cząsteczek do np. vectora (uwaga, kopiowane jest tylko pole Vertex struktury Particle), można go posortować standardowym algorytmem sort z użyciem klasy ParticleComparer.

W przypadku opisywanego tutaj systemu bilbord cząstki powinien być rysowany w płaszczyźnie równoległej do ekranu, obrócony o wartość Angle w tej płaszczyźnie, mieć w scenie wymiary Size x Size. Bilbordy są być teksturowane przy użyciu dwóch tekstur – pierwsza załadowana z pliku smoke.png zawiera kanał alfa i reprezentuje półprzezroczysty obraz, który należy powinien być nałożony na powierzchnię bilbordu. Druga – ładowana z pliku smokecolors.png również zawiera kanał alfa, którego wartość jest mnożona przez wartość kanału alfa pierwszej tekstury. Nieco inne współrzędne tekstury powinny zostać użyte do adresowania drugiej tekstury. Po pierwsze powinny one mieć jednakową wartość dla wszystkich rogów bilbordu, po drugie druga współrzędna powinna być równa zawsze 0.5f (ponieważ ta tekstura zawiera tylko jeden wiersz tekseli), po trzecie pierwszą współrzędną powinien być wiek cząstki podzielony przez czas życia cząstki (a więc w trakcie życia cząstka z czasem będzie zmieniała wartość tej współrzędnej tekstury od 0 w momencie narodzin do 1 w momencie śmierci).

Cząsteczki przekazywane są do potoku renderowania jako lista pojedynczych punktów. Za stworzenie na podstawie każdego z nich dwóch trójkątów do wyświetlenia bilbordu reprezentującego daną cząsteczkę służy shader geometrii („main” w pliku „particlesGS.hlsl”). Warto zauważyć, że pozycja przekazywana do shadera geometrii jest już przekształcona do układu współrzędnych kamery, tak więc jeżeli punkt ten będziemy przesuwać tylko na płaszczyźnie XY, bez zmiany współrzędnej z, automatycznie otrzymamy bilbord równoległy do ekranu.

W programie zdefiniowane zostały już dwie zmienne, dx i dy, określające położenie prawego górnego rogu kwadratu o środku w punkcie (0; 0) i boku Size obróconego o kąt Angle. Poszczególne rogi bilbordu można otrzymać przesuwając położenie cząsteczki o:

- Prawy górny róg – (dx, dy)
- Lewy dolny róg – (-dx, -dy)
- Lewy górny róg – (-dy, dx) (warto zastanowić się dlaczego)
- Prawy dolny róg – (dy, -dx)

Pozycję każdego z rogów trzeba jeszcze przekształcić przez macierz rzutowania.

Współrzędne pierwszej tekstury dla każdego z rogów są w miarę oczywiste

- Prawy górny róg – (1, 0)

- Lewy dolny róg – (0, 1)
- Lewy górny róg – (0, 0)
- Prawy dolny róg – (1, 1)

Natomiast współrzędne drugiej tekstury są dla wszystkich rogów takie same (sposób wyznaczania podano powyżej). Po wypełnieniu pól, wierzchołki należy dołączyć do strumienia wynikowego ostream, za pomocą metody Append, tak aby tworzyły one dwa trójkąty (należy szczególną uwagę zwrócić na kolejność wierzchołków, tak aby trójkąty zwrócone były w kierunku ekranu). Wierzchołki dodawane do strumienia traktowane są domyślnie jako TriangleStrip (w odróżnieniu od TriangleList), aby uzyskać poprawny wynik można np. dodać wierzchołki w kolejności {Lewy Dolny, Lewy Górny, Prawy Dolny, Prawy Górny}.