Jose Suarez
Alex Torrents

Deferred Shading

Nuevo 20/06/2017

G-Buffer

Rediseñado a 3 texturas:

- gDiffuse: las uv de los elementos, 0.0 y alfa 1.0
- gNormal: normales compactadas con sphere map, depth y alfa 1.0
- gPosition: la posición de los elementos

Sigo sin encontrar el bug del alfa, por lo tanto no he podido utilizar esos valores para pasar la información de las uv's y dejarlo en 2 texturas. También ha sido imposible reconstruir la posición respecto a depth, no he podido construir el 'ray direction'.

BRDF

Componente D pasado a GGX isotropic, y arreglo de la conservación de la energía, ya no se divide entre el número de luces.

Post Proceso

Bloom arreglado al usar un pasado de 5 texels y usar otros pesos gaussianos, de learnopengl.

Albedo y material de los elementos se envían directamente al shader pbr, cómo un texture atlas, se escala las nuevas uv's en el vertex shader del gbuffer. Cada elemento tiene guardado la posición en el texture atlas, y con esto se calcula el offset para conseguir las uv's nuevas . El atlas se encuentra en la carpeta resources/images.

Práctica original

Moverse por el escenario

wasdqe: para mover la cámara
ratón: para rotar la cámara, es necesario mantener el botón izquierdo pulsado.

G-Buffer

está compuesto por 4 texturas:

- gDiffuse: Color de la textura.
- gNormal: Las normales de la escena
- gPosition: La posición de los elementos en escena.
- gMaterial: El material de los elementos. El material se carga a partir de la textura RGB = 'specular', 'roughness' y 'metallic', en este orden.

Pasamos las posiciones y las normales descomprimidas porque cómo igualmente tendríamos que pasarle.

Por un bug que no se ha podido solucionar hemos tenido que dejar el alpha de todas las texturas a 1.0. Si no aparecía la escena transparente. Por eso pasamos el material en otra textura, cuando la podríamos haber guardado en el alpha de los otros 3 canales.

BRDF

Para la luz hemos colocado 2 luces, una luz azulada direccional para dar una sensación de noche y una luz anaranjada para la lámpara.

Fresnel: hemos usado el método schlick
D: Beckmann es el método usado para esta componente, es de los más fáciles de implementar.
G: En esta componente hemos usado cook-torrance, comparado con el implícito da mejores resultados y neumann daba un resultado bastante feo en este caso.

Post-proceso

Hemos puesto 3 tipos de post-Procesado que se pueden cambiar mediante Z, X y C.

Z: Tiene un post-procesado de Bloom y Tone Mapping. Para tener la imagen de Bloom sacamos una imagen de luminancia en el 'pbr.fshader' y lo pasamos dos veces por el 'blur.fshader' la primera pasada con la luminancia, y la segunda con el output de este. Sacando la versión 'blurreada'. Y al final del stack, 'shader.fshader' se suma la imágen con luz y la imágen bloom.

X: Mismo que Z pero añadiendo pixelación. Misma pasada de Tone Mapping y Bloom pero antes de sumar las dos texturas, escena iluminada y bloom calculamos las uv's para sacar una versión pixelada con floor().

C: Visión nocturna, pasamos el color a una escala de verde y le añadimos vignetting. Para pasar el color a una escala de verdes lo multiplicamos por un vec3() dando más peso al canal verde. Una vez esto le pasamos un filtro de vignetting.

Implementación adicional

Se ha añadido un cubemap para todo el escenario, por algún error extraño no se podía añadir al color albedo. Para activarlo se ha de ir al shader 'pbr.fshader', y descomentar la línea 114.

Se puede compilar el shader en vivo pulsando la tecla T dentro del juego.
Se puede recargar la escena en vivo pulsando R, carga el json que está dentro de resources/scenes/