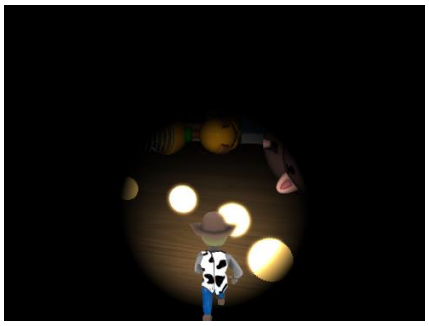
Running Woody (ToyStory)

2019-27605

김미정



1. 개요

본 project는 woody가 돌아다니며, 저금통 Hamm으로부터 쏟아진 동전을 수집하는 미니 게임이며, 동전 12개 이상 수집 시 방 안의 불이 켜지게 된다. 기술적으로는 Toystory 속 주인공 andy의 방을 사실적으로 표현, light 기법 사용, skeleton animation, depth buffer을 활용한 elevation map 구현 등을 목표로 한다. 본 보고서의 설명은 **reference한 코드에서 변경/추가한 부분을 위주**로 작성하였다.

2. 조작방법 및 작동법

- W, A, D, S: view point를 기준으로 각각 forward, left, right, backward으로 카메라 position이동(일부동시사용가능)
- J: 누르면 0.9초동안 점프를 하게 됨
- Mouse: 카메라 viewpoint이동
- Number 3: light 사용 여부 toggle
- Spacebar: bloom 사용 여부 toggle (light가 on일때만 적용)
- 작동: code/bin/main.exe 실행 or main.cpp 실행

3. Multi object in single .obj file

앤디의 방을 보다 사실적으로 구현하기위해서는 기존 과제에서 했던 것과 달리 하나의 obj 내에서 여러. Blender라는 프로그램을 사용하여 obj로 변환시켜 준 후, mtl 파일의 texture file 경로를 수정하였다. 여기서 learn-opengl에 있는 코드의 경우, mtl 파일에 있는 texturemap 정보만 불러오기 때문에, 추가적으로 ka, kd 등을 불어와서 **uniform buffer**까지 이용하는 코드, texture loading(mesh 당 카테고리(normal, specular...)별로 최대 하나의 texture map만 읽게 하였으며, shader에 적절히 binding 하였다. 추가적으로 벽면, poster, 천장 등의 quad 모양의 물체일 경우, VAO를 정의하여 반복적으로 shader에 넣어 계산량을 줄였다.

4. Skeleton Model

참고링크: http://ogldev.atspace.co.uk/www/tutorial38/tutorial38.html https://mathmakeworld.tistory.com/8

1) **Skeleton animation(Woody)** 참고 링크를 기반으로 작성하였으며, animation 정보가 들어있는, .mae file을 assimp을 통해 load한다. model_ani.h에서 본 링크와 달리, vertex에 연결된 bone의 정보는 glm::vec3의 형태의 vector로 받도록 하였고, 한 vertex 당 최대 4개의 bone에 영향을 받도록 코드를 작성하였다. (bone에 대한 weight 정보). 또한 assimp mat4 -> glm::mat4 변환, vertex binding 등은 독자적으로 하였다.
2) **Keyframe interpolation** animation은 일정한 duration동안 동작이 반복되는 것이므로, program의

currentTime%duration 에 해당하는, 시간의 bone의 움직임을 시간에 대해 interpolation하여, 해당 시간의 bone의 정보를 aniShader에 넘겨주었다.

3) **Collision** 5. 에서 언급하겠지만 본 project는 전체 공간에 대한 elevation map을 사용하여 animation 모델의 y 좌표를 설정한다. 따라서 본 project는 x, z 좌표가 정해지면, y좌표가 유일하게 정의된다는 가정하에 x, z축 기준으로 coin과 animation 거리가 1.0 이하일 때, collision이라 정의하여 해당 coin을 제거시켰다.
4) **Jump** "J" 를 누르게 되면, 0.9초동안 점프를 하도록 하였으며, 이는 중력장을 고려하여 좌표를 설정해주었으며, 해당 함수는 main.cpp의 cal_ani_location에서 구현하였다.
5) **이동방향** skeleton model(woody)의 이동방향에 맞추어 모델 자체의 rotation도 바꾸어 줘서, 움직이는 방향에 맞게 방향을 틀어 뛰는 것처럼 구현하였다. 몸통 회전 방향은 45°간격으로 총 8개이다.
6) **texture** project의 컨셉, woody에 맞게 카우보이 animation 모델의 texture을 직접 간단하게 수정하여 최대한 woody 같아 보이도록 detail한 요소를 살렸다.

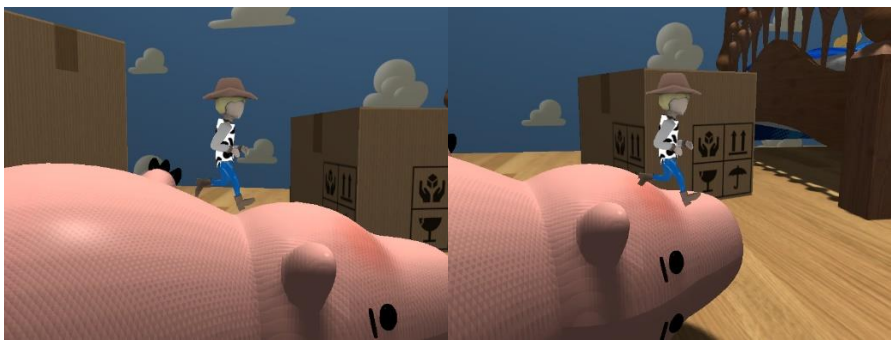
keyboard W & D, keyboard S & A

texture 수정 전후

5. Elevation map via depth buffer

이 section은 demo 이 후에 완성시켜, demo 발표에 포함 되어있지 않던 내용이다. 만약 animation model 앞에 장애물(e.g. 침대, 박스, 저금통 등)이 생겼을 때, 별다른 처리를 해주지 않으면 skeleton animation은 장애물들을 고려하지 않은 채 통과할 것이다. 따라서 이를 해결하기위해서, **depth buffer을 이용하여 elevation map을 구하고, 이를 현재 좌표와 비교하여 animation model의 y축 값을 translation하는 방법**으로 구현하였다. 보다 구체적으로는, lookat 지점을 천장의 중심, ortho 각도도 충분히 크게 하여 최대한 천장위에서 바로 바라보았을 때의 깊이로 구현하고자 하였다. 또한 기존의 hw4에서는 fragment shader에서 depth비교를 한 것과 달리 vertex shader에서 ((현재지점~look at 기준점) – (새로운 position ~look at 기준점))*max_depth 만큼 animation의 y축 좌표를 translation 시켰다. 본 방법은 물체의 위치들이 바뀌었을 때 매번 전체 elevation map을 직접 지정하지 않아도 자연스럽게 animation이 장애물들의 높이를 고려할 수 있다는 장점이 있다.

DepthMap



DepthMap부터 구현한 smooth한 elevation 변화

6. Light

Light의 경우, learnopengl을 보고 개념을 익힌 후 구현하고자 하는 게임의 상황에 맞게 적용시켜보았다.

1) **Spotlight** 불이 켜지기 전의 손전등, 책상 위의 lamp, 불이 켜진 후의 천장의 전등 세 군데에 모두 적용하였다. Spotlight는 빛의 direction, 광원에서 해당 point사이의 벡터 사이의 각도에 clamp를 주어 구현하는 것이며, flashlight의 경우, 빛의 방향과 위치를 각각 camera.position, camera direction으로 설정한다. 그리고 상황에 맞게 attenuation, cut off 등을 조정하였다.
2) **HDR** 1.0 이상의 빛을 구현하는 방법으로, 기존의 fragment shader에서는 clamp되던 것들의 정보를 더 활용하기 위한 것이다. 본 HDR을 사용하고, flashlight의 light색을 1보다 크게하여, hdr의 효과를 크게 적용시켰더니, 보다 자연스럽고 사실적인 flashlight를 구현할 수 있었다.
3) **Bloom** 빛의 양이 1.0 이상인 부분만 따로 buffer에 저장한 후, 가우시안 blurr등을 사용하여 해당 이미지를 blur 시킨 후, 원본 이미지에 덧대는 방법으로, 아주 밝은 빛인 경우 퍼져보이는 현상을 구현하는 것이다. 이로 책상위의 램프, 천장의 전등을 보다 사실적으로 표현하였다.

Bloom 전

Bloom 후

4) **Shadow** Shadow map을 엄밀히 표현하기 위해서는 전등에서 각도를 줄이면 되는데, 현재 rendering하는 물체의 수도 많아 계산량을 최소화하기위해 elevation map을 구할 때 얻었던 depth map을 그대로 shadow에 활용하였다. 그 결과 그림자가 천장의 중심의 spot light에 의한 효과가 아닌, 천장 전체에서 빛이 균일하게 내려올 때의 그림자이긴 하지만 시각적으로 부자연스러움이 없는 것을 확인하였다.