Kay Men Yap 19257442

CG Assignment Report

I have used the sample_2 file as my base file and used the resources and includes file given to us from blackboard in tutorial 1. I also used the light attenuation code from https://learnopengl.com/Lighting/Light-casters .

All update animation methods are called from the update_animations method for readability purposes.

Light attenuation

I copied the additional struct variables for the Lighting struct and added the attenuation calculation inside the fragment shader and the attenuation only affects the diffuse and specular lighting as a multiplicative factor.

Brightness radius

The way I implemented the functionality of increasing and decreasing the brightness radius of the torch by storing linear and quadratic values needed for light attenuation formula implemented from the learnopengl website as mention earlier. Then I have an index variable that is initialized to 4 and at every loop cycle of the most outer loop that keeps running until the window closes will set the linear and quadratic value with the value in the array at the current index of the light struct of the lighting shader. <K> and <L> keys will decrement and increment the index number each time they are pressed until the index reaches the min or max index of the array.

Jumping

I modified the camera header file to stop the camera from flying by setting the y component of the camera position to be at 0.9 when not jumping. When the player is jumping, I have method called update_jumping_animation which increment the camera position's y component to the max jump height I specified and once it reaches the max, it will decrement until it reaches the min jump height which is the original camera position's y value which is 0.9. This increment/decrement is done in every iteration of the while loop that loops until the window is closed so that the player can still move while jumping.

Image Textures

I have used more than 6 image textures for the assignment which include image textures given to us from the tutorial 1 file and image textures I found on google images that would match the appearance of the objects I'm creating.

Objects

The objects in the assignment program, which are all made by myself with trial and error and some used the table from sample_2 file as the base object before adjusting the object to look like what it is supposed to be, are Sven, Sheep, Tree, Torch, Spike and Sign (which represent the msg screen for winning or losing). For all objects, I have a scale array that specify the size of each part of the object and a parts_position arrays that specify where does the part of the object is supposed to be located at for that object. Codewise, I do all of the translations then the rotations then the scale operations for all objects since opengl uses column major transformation. With the scale and part_position arrays, I simply iterate through them and render the objects like how the sample 2 file renders the table.

Sven

Sven is made using 8 cubes (4 legs. 1 head, 1 face, 1 body, 1 fur coat, 1 tail). Every part except for the face will appear like rectangles whereas the face is like a thin sheet that is on the head rectangle which is what I did for the sheep face and the message image on the Sign object. Sven remains in the camera's view by making Sven's position be the addition of camera's Position and Front vector as the value. I did make Sven's position's y component to 0 after the addition to make sure Sven stays on the ground.

I have a toggle_sven_pickup_distance method that checks if the player is close enough to pick up sven with the <E> key.

Sheep

The Sheep is made using a similar design of 7 objects (4 legs, 1 head, 1 face, 1 body). The legs of the Sheep will rotate as if it is actually walking when it is moving towards the camera by having a method that increments and decrements the angle to rotate the legs by. The angle for the legs are simply incremented by 2.25 until it reaches the angle 22.5 then it decrements by 2.25 until it reaches –22.5 and then it increments again and continues as long as Sven is picked up by the player. I used the value -22.5 and 22.5 as the range as from trial and error, this range makes the legs rotate to an angle that seems normal for a leg to rotate by. I used the increment/decrement factor of 2.25 as this makes the legs move at a pace that isn't too slow or too fast. All of the code to increment/decrement the leg angle is done in the update_leg_animation method.

The sheep will also face the camera when sven is picked up and this is achieved by defining a vector where the sheep will original look at which is glm::vec3 (0.0f, 0.0f, 1.0f), and then calculate the vector which is the direction of the sheep current position to the camera. If the camera position's x is greater than the sheep current position's x then the direction vector of the sheep to the camera is camera.Position – sheep current position to and vice versa is sheep current position's x is greater than camera posiiton's x to ensure the angle is positive. This only calculates the smallest angle from the sheep's original direction vector to the direction vector of the sheep to the camera position so if sheep current position's x is greater than the camera position's x, the angle is more than 180 hence we add 180 degrees to the angle variable before adding it with the angle between the sheep's original direction and current direction to camera but if camera position's x is greater than sheep position's x then the angle is initialized to 0 before adding the angle between the sheep's original direction and the current direction to the camera. The calculation for the angle is done using the acos function and the glm::rotate method and glm::normalize method the current direction vector of the sheep to the camera. After we find this total angle to rotate by, we create the model variable than scale it appropriately, rotate every part by that angle we derived then translate every part of the sheep to its correct place then translate the sheep to its current position.

The way I updated the sheep's position to move towards the camera is a simple comparison between the sheep's current position and the camera's current position and then increment/decrement the sheep's current position's x and z components so that the sheep's position's x and z components are closer to the value of the camera's current position's x and z components.

I call the check_lose method before rendering the sheep with the sheep_position vector to check if the player has lost but where I check in the code whether the sheep has killed the player does not matter as long as the check_lose method is called in the while loop of window not close.

Tree

The tree object is comprised of 1 tree bark and 1 tree leaf rectangles. I used 2 for loops (1 nested within the first one) to create multiple tree objects within the world by translating each tree with the 2 for loop counters I used as the x and z translation but I did not draw the tree when both loop counters result in a position at origin. This allowed me to not need an array of tree positions to know where to draw a tree and simply draw all trees at origin then move it to its position in the world.

Torch

The only light source in the assignment is the torch. I made the torch using the existing light cube from the sample 2 file as the light source and tip of the torch and another rectangle that represents the body of the torch. The torch's position is at origin at the start and will do an animation of spinning and going up and down on the same spot when not picked up. This is done using 2 variables (1 for rotating and 1 for translation in y coordinates) where I increment or decrement the values in those 2 variables in my method called update_torch_animations method.

When the torch is picked up, its position will be the sum of camera's position and camera's Front vector just like how Sven's position is derived when picked up but I set the y component of the sum to be 0.5 so that the torch appears to be floating and "is held" by the player. When the torch is no longer picked up, it continues to do the floating and rotation animation at where the player left it.

I have a toggle_torch_pickup_distance method that checks if the player is close enough to pick up sven with the <E> key.

Spike trap (in the code, it is just called spike for simplicity)

The spike trap is made using 1 rectangle as the base plate and 1 very long and thin rectangle as the spike. I use a similiar method in rendering the spike traps just like the trees where I used a single for loop for the spike traps to appear as a straight line of traps just before the winning floor area by translating them by the loop counter in the x coordinates. I used 2 nested for loops when it was time to render the spike part of the trap to make 25 spikes in each spike trap without needing to store the position value of each spike.

The trap is only rendered in the world when sven is picked up. The traps will also have the spikes going up and down the base plate as animation and the translate in y for the spikes is done using the method called update_spike_animation which will increment the translate in y until it reaches the max spike height then decrement until the min spike height and repeat that cycle. I call the check_lose method on the coordinate of the base plate as that is the center of the trap but the y component of the coordinate to check for lose is set to the translate in y value.

Sign

I made a sign that composes of a stick that is the bottom of the sign, a wooden panel on top of the stick and a very thing rectangle for placing the win or lose screen message on the sign. The sign is only rendered when a player has lost or won and it is rendered at the corner of the map so that nothing is blocking the player from looking at the sign. The image that has the win or lose screen message is generated using an online website I found on google with the url (https://onlinetexttools.com/convert-text-to-image).

Reset

I have every object's initial position saved into constant glm::vec3 variables and when <R> key is pressed, all of the object's position variables are reset to their original vector and camera is resetted with a method I added into camera.h which is a reset method that sets all classfield of the Camera object back to their defaults. All of the animation and angle of the sheep to face the camera is not resetted as they are computed as the game runs.

Orthographic mode

I have an if else statement where the projection variable is either using the glm::perspective method or the glm::ortho method depending on the ORTHOGRAPHIC_MODE Boolean to generate the projection matrix.

Bright mode

I have an if else statement that checks on the BRIGHT_MODE Boolean to either set the ambient light of the lighting shader to 1.0 if BRIGHT_MODE is true and 0.3 if BRIGHT_MODE is false.

Game win/lose screen

The game is lost if the player touches the sheep or the spikes while they have sven picked up.

The game is won if the player is carrying sven and walked to the win floor area which is blue in colour and as behind the player when the player spawns.

I simply teleport the camera to the corner of the map and render the sign at the direction the camera is looking at which I specified by resetting the camera and disable all camera mouse input and keyboard input for movement when game is lost or won with an if statement.