CS432 Snow Wonderland Final Project

Joseph Budd

Alan Tsai

Introduction

In the scene of snow wonderland, there is a lodge built in wood and a tiled roof. A wooden door is attached to the lodge. There is a snow man and a campfire just outside the lodge. The snowman head will rotate if the user clicks on it. The lodge is located in the mountains, and while the moonlight shines on the surface, the snowflakes slowly falls to the ground covering the surface. The lodge is surrounded with many trees. The user can also throw snowballs at the direction where they are facing.

Users Guide

Standard arrow keys to walk around within the predefined area. Use “z”, “x”,’c” and “Z”, “X”, “C” to rotate the camera. Press “l” to throw snowball (Figure 2) and “spacebar” to switch between the static camera which follows the rotating snowman head, and the free control camera. (Figure 3) When the user is using the static camera, pressing “l” will also shoot a snowball and even faster than the normal free control camera snowball. The snowball will also skid on the ground just like it’s rolling. Use the left mouse click on the snowman head to start or stop the snowman head rotation.

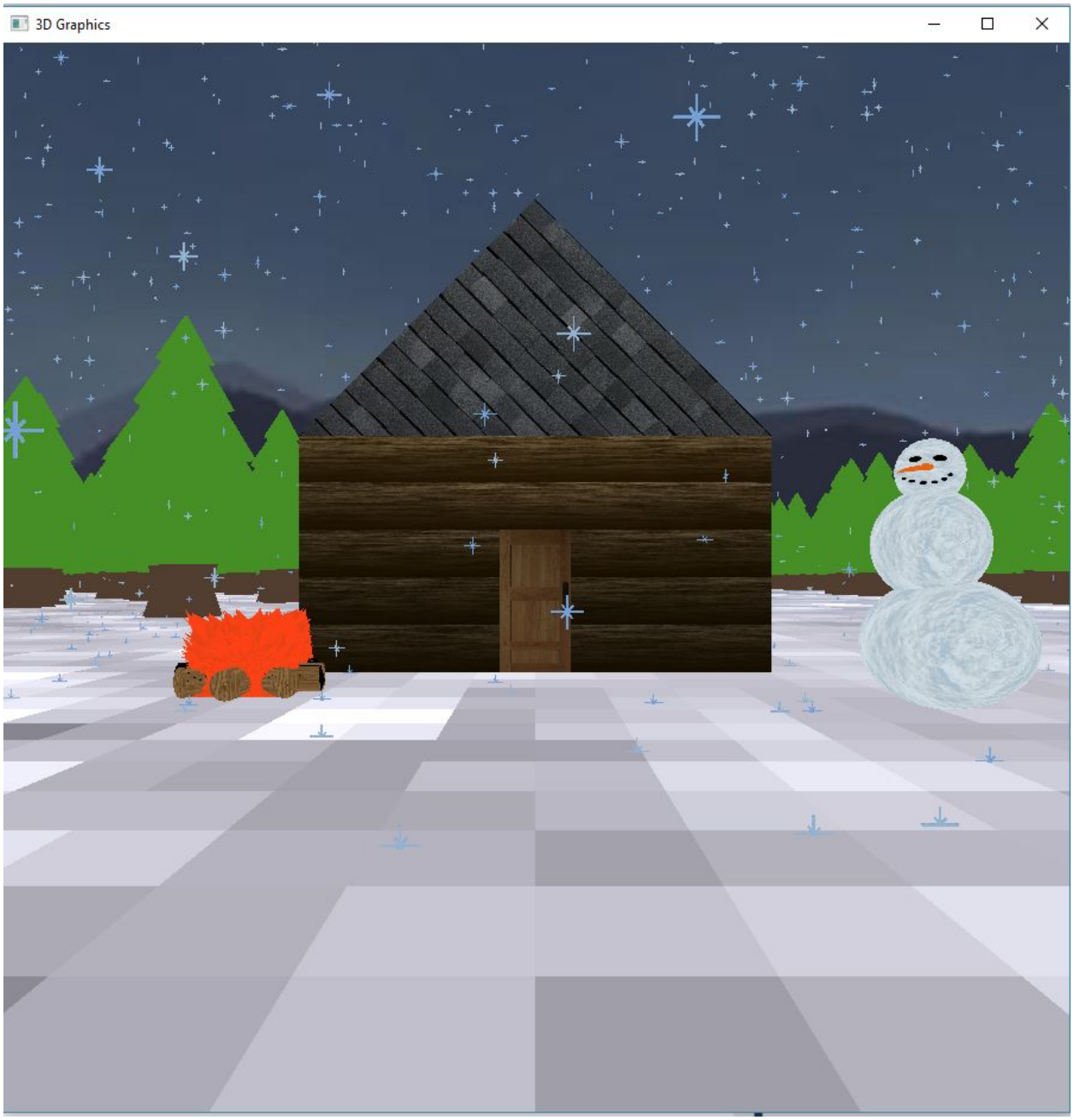

Figure 1 Overview of the scene

Figure 2 Press key “l” to throw snowball

Figure 3 Press Spacebar to toggle the camera for snowman head first person view

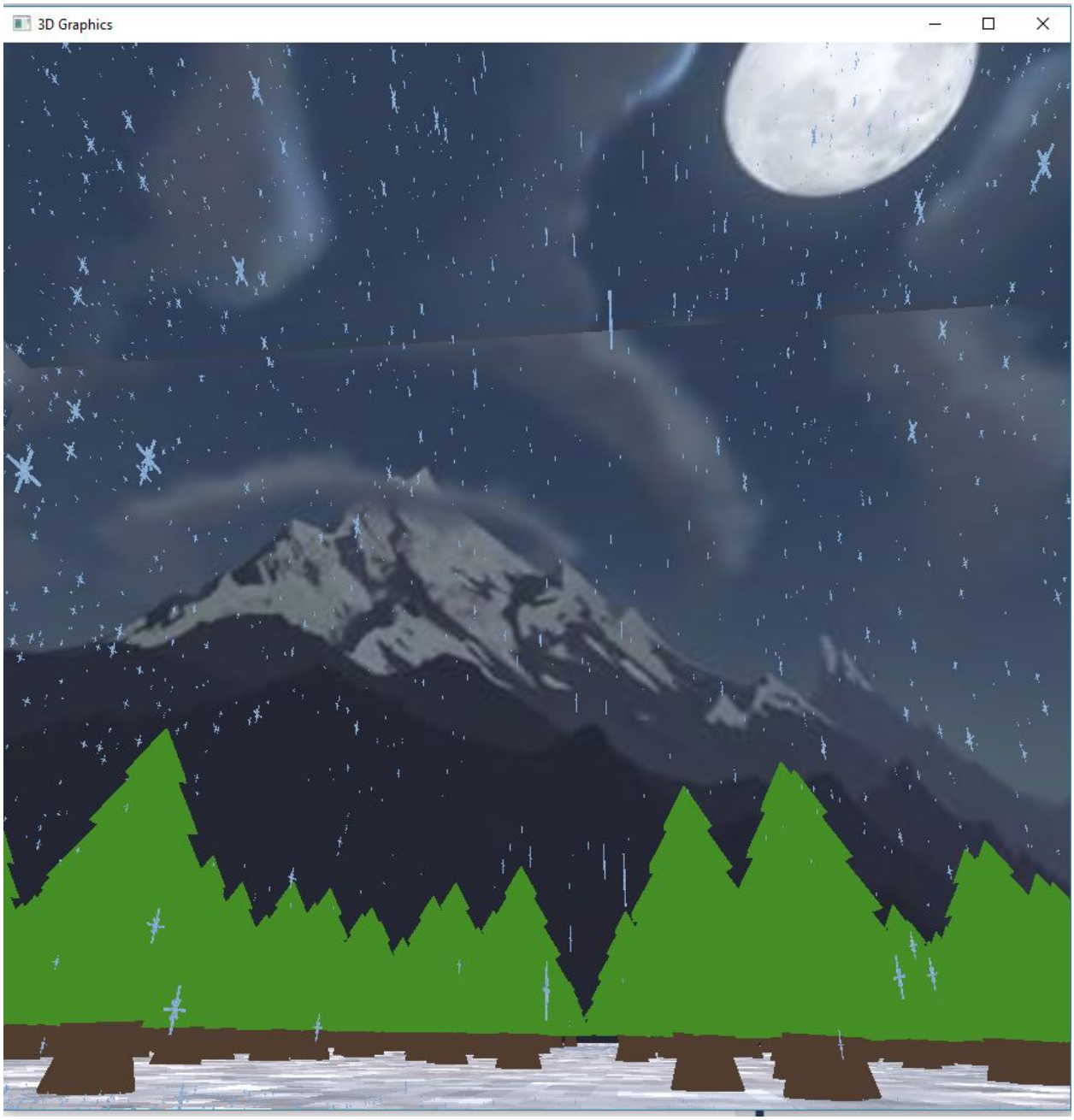

Figure 4 Overview of the surroundings

Technical Details

The scene is rendered using a number of objects. The walls of the house are building using a single elongated cube. The roof of the house is built with a triangular prism which is stretched to match the house. Additionally, a door is positioned on the house which is a cube which has been stretched vertically. All of these objects are textured accordingly by defining the

texture vertices as done in previous assignments and then mapping the images to the objects.

Movement throughout the scene is done using the standard Camera model as was previously implemented. An improvement which was made specifically for our project is that the camera is constrained to a specified range. As part of this range, the lower bound vertically is the ground plane, so that the user is not able to fly through the ground plane. No additional clipping is done against any objects, so the Camera will still fly through objects as seen previously.

Snowballs are the simplest model we have in our scene. Snowballs are textured spheres which are scaled to be the appropriate size. Snowball throwing is handled according to the camera location, and based off of the particle system model. When the user selects to throw a snowball, a new snowball is initialized according to the camera's current location and normal. Snowball updates are handled based on a timercallback and are translated some percent of the normal every tick of the timer. Additionally, there is some rudimentary checks in place so that snowballs will skid along the ground for a certain number of ticks before falling through. Additionally, when you are in the snowman camera view, the snowballs travel significantly faster, essentially making a snowman turret.

The visible portion of the snowman is built using 3 spheres which are scaled to better match the traditional snowman format. The Snowman head texture was a tricky one, since it is build with many triangles. We initially tried to use the algorithm defined in the slides to generate s & t based on the x & y coordinate of the sphere but that gave us a texturing which was not correct. In doing some testing we found that this algorithm was providing us with s values between 0.25 and 0.75 and t values between 0.5 and 1.0, which ultimately led to the only the bottom of the texture being used to texture the sphere and it was doubled in order to occupy the entire sphere's surface. Ultimately we moved to using the algorithm which we found from another source (http://cse.csusb.edu/tongyu/courses/cs520/notes/texture.php) . This provides us the algorithm to create the points, normals, and texture coordinates. We then mapped a 2D texture to the sphere using the texture coordinates. This algorithm was used to generate vertices, normals, and vertex texture locations for all three snowman parts. Additionally, we utilized the same ray casting and collision detection that was used in previous assignments to detect when a user clicks the snowman's head. The change made here was that instead of changing the color of the triangle clicked, it would toggle a boolean to start/stop rotation.

The campfire is one of the more interesting models as it is not an object/shape that we had in previous assignments. The logs of the campfire were designed in Blender and then the vertices and normals were exported and converted to the format of vec4's as is expected in our code. The campfire logs are then drawn based off of these vertices, and textured based on the vertex texture locations. For the actual fire portion of the campfire, this was rendered using 1000s of particles all of which are colored the same. Upon initialization, each particle gets a variety of attributes assigned to it, each of which are randomly generated. The first is that the particle is assigned a random location within the bounds of the fire and logs. Then, each particle has a randomized velocity between 0 and 1. Finally, the particle is assigned a random step

maximum value. Based on a timercallback function, the smoke particles' positions are updated based on their individual velocity, which is only in the y direction. Also, a step value is incremented within each particle. A check occurs here as well to determine if the current step value is greater than the maximum step value, and if it is, the particle's position is updated to return it to the ground. All of these particles are then drawn as a triangle fan, as opposed to individual points. All of this allows for a more randomized look to the fire which provides more realism.

The trees are also interesting as they are a model we did use in class and they are rendered using a method we did not discuss in class, Instancing. For the trees, like the campfire, the model itself was created in blender and exported. Each tree is made up of two parts, the leaves part and the trunk part. This allows for the leaves and the trunk to be easily individually textured. All of the initialization is standard to what we do for a normal object, what makes these trees interesting is that they are rendered using instancing. Instancing allows you to put one set of the vertices onto the GPU and then basically pass in offsets for all of the trees. For our application we use the single set of tree vertices where the bottom of the tree is at the origin, and put these into the buffer. Additionally, we put offsets for these trees which are randomly generated, into the buffer. When the Shader program pulls data from the buffer we tell it to pull the same offset for every vertex of the same tree, and only pull a new offset when drawing a new tree. What this allows us to do is easily and computationally inexpensively draw all of our trees, without performance implications.

Finally, the snowflakes are the last piece of our scene. At first, we had issues with the amount of snowflakes falling due to us trying to render many very complex snowflakes, each having a high number of vertices. After narrowing down the vertices, the snowflakes amount was much bigger but the falling density was uneven due to random number generation. We randomly generated the position of the snowflakes within a specified range, which allowed for a better spread of the snowflakes. Snowflakes was the reason that Instancing rendering was initially explored. When rendering all of the snowflakes that we wanted initially, it had a huge performance impact, which instancing solved. Similar to the trees explanation, each snowflake has a randomly x,y, and z coordinate which is the offset of the snowflake from the origin. Again, as with trees, these offsets are placed into the buffer and when drawing are passed into the shader program. This allows all snowflakes to use the same vertices on the GPU, and apply their corresponding offset. Updating the snowflake position is handled with a timercallback function, which updates to the position to be slightly lower than it is now, giving the appearance of the snowflake falling. A check occurs here which prevents this update to the y coordinate happening if the snowflake has reached the ground plane. Additionally, based on a random number generation, each snowflake's position may be updated with an x and/or z increment or decrement, which gives the appearance of drifting as the snowflake falls. Rendering the snowflakes using the instancing technique had a huge impact on performance and allowed us to render many more snowflakes, giving the scene a better appearance. Previously, using traditional techniques, we were able to render about 1500 snowflakes with minimum performance impact,

though these barely covered the scene and it was difficult to tell that the objects were supposed to be snow. Snowflakes are rendered in groups called clouds. Instancing has allowed us to render over 5,000 snowflakes per cloud, and have multiple cloud instances being rendered on screen with very minimum performance impacts. Finally, on a timercallback, new cloud instances are generated up to a limited number, to prevent too many being on screen at once. Also in a timercallback function, existing cloud instances are checked to see if all snowflakes in it have landed, and if they have, it will be removed from the scene to prevent performance degradation.

Work Allocation

Joseph Budd: lodge, campfire, snowflakes, texture mapping, animation, Debug

Alan Tsai: Resize all the objects to fit scene scale, position the object in place, texture mapping, animation, Debug, documentation

Citation

Snowman head texture:
https://s-media-cache-ak0.pinimg.com/564x/51/cd/16/51cd1626473533257ee1ab4dd362ddfb.jpg
Tree branch texture: http://spiralgraphics.biz/packs/plant_bush_tree/?22
Tree trunk texture (though only a small section is used for color):
http://www.spiralforums.biz/index.php?showtopic=2428
Door Texture:
http://www.doordecorate.net/top-21-pictures-room-door-texture/from-simpson-high-performance-wood-doors-with-a-10-year-warranty/

Campfire log: http://s10.postimg.org/3tl3fhq7t/Wood01_c_pia.png

Cabin Walls:

http://s759.photobucket.com/user/3corrie/media/log%20cabin%20textures/211_tile_Log_Cabin.jpg.html?sort=3&o=5

Cabin Roof: http://www.3dshare.org/2016/09/mapping-asphalt-roof-textures.html

Ground Plane Ice/Snow:

http://www.cadhatch.com/seamless-snow-and-ice-textures/4588167780

Snowflake:http://spiralgraphics.biz/packs/snow_ice/?2

Snowman Lower:

https://s-media-cache-ak0.pinimg.com/564x/c2/48/6f/c2486f49be929cee14e9973d411abc88.jpg

Instancing References:

https://learnopengl.com/#!Advanced-OpenGL/Instancing

http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/

Tree Model:

http://tf3dm.com/3d-model/low-poly-tree-96065.html