COSC363

Computer Graphics

Assignment 1
Graphics Factory

Yumeng Shi
74341182

Declaration

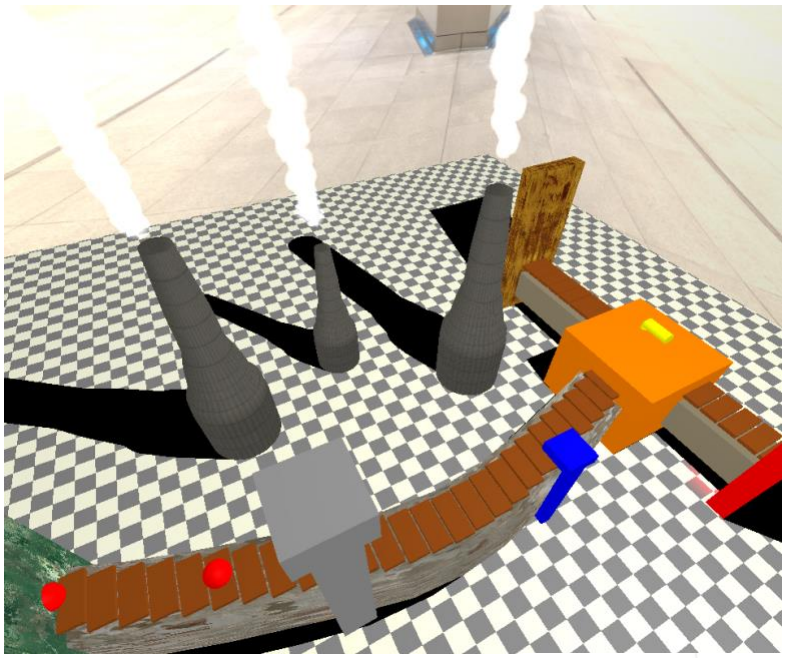
I declare that this assignment submission represents my own work (except for allowed material provided in the course), and that ideas or extracts from other sources are properly acknowledged in the report. I have not allowed anyone to copy my work with the intention of passing it off as their own work.
Name: Yumeng Shi
Student ID: 74341182
Date: 27/3/2025

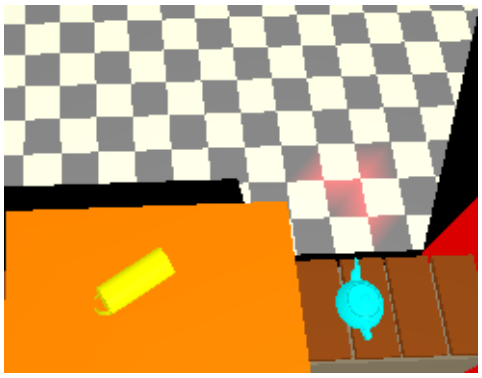
The factory scene is in a 3D animation that showcases a factory environment with various objects and animations using OpenGL with C++. The scene includes a conveyor belt with moving bricks and items, a hammer that moves up and down, a sprinkler system, and three chimneys with smoke particles. The scene also features a spotlight that rotates around the conveyor belt. The user can interact with the scene using keyboard and special keys to rotate the view, move the camera, and toggle between different polygon modes with disabled lighting in wireframe mode. A skybox texture is set around in 5 side of the wall with a cheeseboard flood project the shadowed objects.

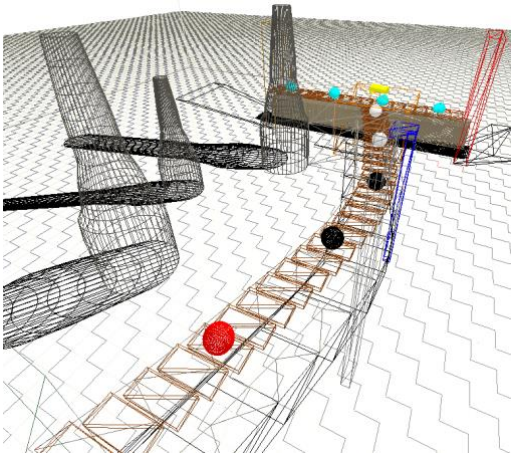
Red molten material starts from a “green” textured gate and is hammered down by a gray hammer into black carbonized spheres. These spheres are then cooled by a blue sprinkler, turning into white semi-finished material balls. Detection lights on a gray sorting box scan materials coming from all directions. After sorting in the box, the materials are combined with colored materials from the “rusty” textured gate to form finished teapots, which are then transported out through a red gate.

In the background, there are three chimneys releasing the energy produced with smoke animation build by particle systems. The conveyor belt is divided into two parts: the X-axis section is achieved using a curved surface, and the Y-axis section is a straight line. The bricks on the conveyor belt smoothly simulate the animation of operation, moving along the conveyor belt simultaneously with the material (spheres).

• **At least two screenshots showing important aspects of the scene or animation**







• **Extra features implemented:**

3.1 Planar shadows cast by at least two objects

Chimneys, conveyor belt base ,Red door

3.2 Moving conveyor

The bricks on the belt base have animation controlled by the timer function.
`brick_positions_h[numBricks]` *is a list to holding all the bricks we use, and use* `initializeBricks()` *to generate the position of this bricks before add the cube in it.*
`drawConveyorBeltBricks()` *function build the entity cube and translate to the right position.*

3.3 Non-linear conveyor

`xpts[13]` *and* `zpts[13]` *hold the turning point of the conveyor, use* `normal(int i)`, *to calculate the normal and* `drawConveyorBeltBottom()` *to full the each side by* `GL_QUAD_STRIP` *function.*

3.4 A sky box or sky dome/sphere

Build the wall on 5 surfaces by `walls()`*, use* `loadTexture()` *to add texture on each side on them. Keep the bottom floor to achieve shadows.*

3.5 A dynamic spotlight

The top part of yellow cylinder in `drawSpotlight()` *generate the red spotlight with the turning around animation in timer by update the dynamic variable* `spotlightAngle`*. Use LIGHT1 be this spotlight with a focus cut-off 2.0 and facing down make it more visible.*

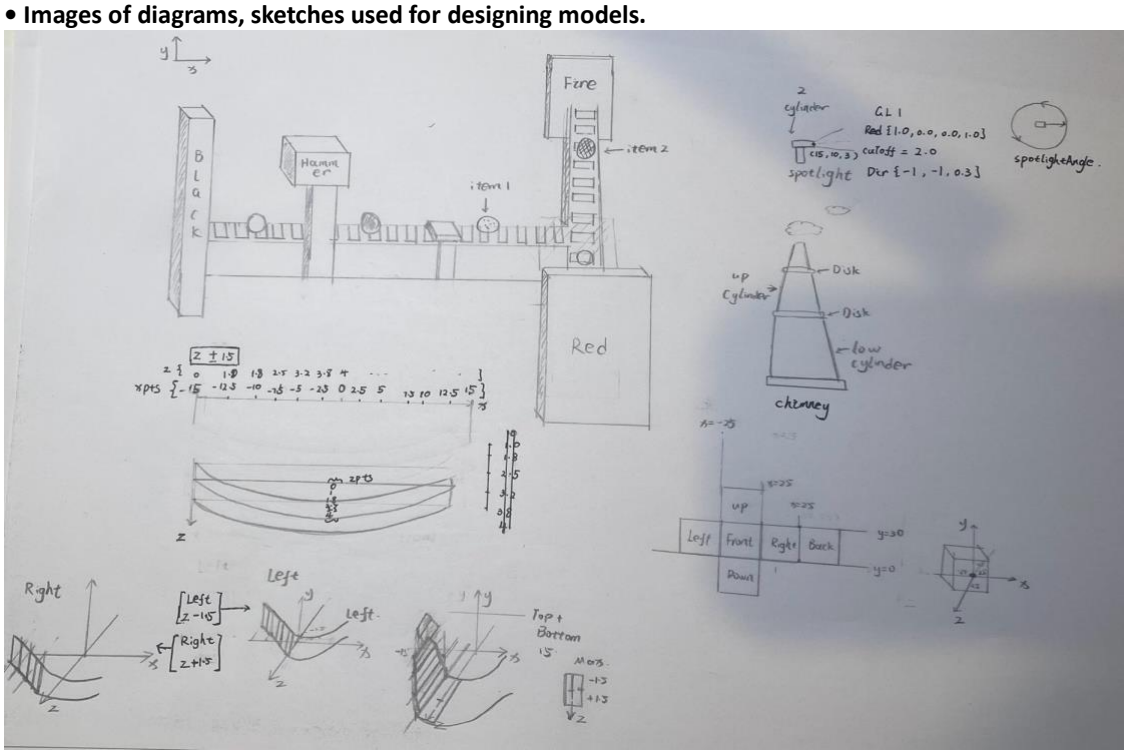
3.6 A custom-built sweep surface

The Non-linear conveyor, `drawConveyorBeltBottom()`*.The Chimney function ,* `drawChimney()`*.*

3.9 Particle systems

`drawParticle(),newParticle(),updateQueue() and timer().`

• Images of diagrams, sketches used for designing models.



•Surfaces Equations:

`normal(int i)`

Normal vector equation:

glNormal3f(0., -(zdiff1 + zdiff2), (xdiff1 + xdiff2)); N = (0 , -(z2-z1+z3-z2) , (x2-x1+x3-x2))

`drawConveyorBeltBottom()`

Bottom:

x = xpts[i], y = 0, z = zpts[i] - width /2 x = xpts[i], y = 0, z = zpts[i] + width / 2

Top :

x = xpts[i], y = height, z = zpts[i] – width/2 x = xpts[i], y = height, z = zpts[i] + width/2

Left side:

x = xpts[i] , y = 0, z = zpts[i] - width / 2 x = xpts[i+1] , y = 0, z = zpts[i+1] - width / 2

x = xpts[i] , y = height, z = zpts[i] - width / 2 x = xpts[i+1] , y = height, z = zpts[i+1] - width / 2

Right side:

x = xpts[i] , y = 0, z = zpts[i] + width / 2 x = xpts[i+1] , y = 0, z = zpts[i+1] + width / 2

x = xpts[i] , y = height, z = zpts[i] + width / 2 x = xpts[i+1] , y = height, z = zpts[i+1] + width / 2

`drawChimney(float x, float y, float z, float scale)`

X-Rotation equation:

wx[i] = cos(angStep) * vx[i] + sin(angStep) * vz[i]; x' = x * cos(θ) + z * sin(θ)

Z-Rotation equation:

wz[i] = -sin(angStep) * vx[i] + cos(angStep) * vz[i]; -x * sin(θ) + z * cos(θ)

Normal-X-Rotation equation:

mx[i] = cos(angStep) * nx[i] + sin(angStep) * nz[i]; nx * cos(θ) + nz * sin(θ)

Normal-Y-Rotation equation:

mz[i] = -sin(angStep) * nx[i] + cos(angStep) * nz[i]; -nx * sin(θ) + nz * cos(θ)

•Animation Equations:

`timer(int value)`

Linear motion:

brickposition += item1Speed; brickposition += item2Speed

Conditional reset:
if (brickposition > 14.0) { brickposition = -15.0; } if (brickposition > 8.5) { brickposition = -9.9; }
Hammer motion:
if (hammerDown) { hammerPosition -= 0.1; } if (hammerPosition <= 2.5) { hammerPosition += 0.1; }
Spotlight rotation:
spotlightAngle += 1.0; if (spotlightAngle >= 360.0) { spotlightAngle -= 360.0; }

• Control functions (keyboard, mouse, special keys)
`special(int key, int x, int y)`
Special Keys:
Left arrow: Rotate view left Right arrow: Rotate view right
Up arrow: Move camera up Down arrow: Move camera down
`keyboard(unsigned char key, int x, int y)`
Keyboard:
'q' or 'Q': Toggle polygon mode (disabled lighting in wireframe mode)
'w' or 'W': Move camera forward 's' or 'S': Move camera backward

• Build commands or instructions for compiling and running the program on CSSE lab machines.

```
cmake_minimum_required(VERSION 2.8...3.5)
project(cosc363pro1)
add_executable(Factory.out Factory.cpp)
set(OpenGL_GL_PREFERENCE LEGACY)
find_package(OpenGL REQUIRED)
find_package(GLUT REQUIRED)
include_directories( ${OPENGL_INCLUDE_DIRS}  ${GLUT_INCLUDE_DIRS} )
target_link_libraries( Factory.out ${OPENGL_LIBRARIES} ${GLUT_LIBRARIES} )
```

• References
Skybox texture: *https://polyhaven.com/a/metro_vijzelgracht*
Photoshop for trim the photo
Green door Texture: *https://polyhaven.com/a/painted_concrete*
Rusty door Texture: *https://polyhaven.com/a/rusty_metal_03*
Chimney Texture: *https://polyhaven.com/a/rectangular_facade_tiles*
Conveyor Belt Texture: *https://polyhaven.com/a/japanese_sycamore*
Transfer texture photo format : *https://convertio.co/zh/png-tga/*
Lab code and resources been used.

Generative AI Usage

GPT-O4 :
Generating short code segments for reuse (1-3 lines):
In `interpolate` function, Use to generate the code in the if statement on z-position transit and for reasoning explanation.

```
float t = (x - xArray[i]) / (xArray[i + 1] - xArray[i]);
            return zArray[i] + t * (zArray[i + 1] - zArray[i]);
```

In keyboard interaction the direction vector from the camera's position to the look-at point `d_x` and `d_z` defined.

```
float d_x = look_x - eye_x;
float d_z = look_z - eye_z;
```

Explaining the usage of computer graphics functions, Reasoning problem:
Shadow color defined in OpenGL .
Light color modulation (Ambient light, Diffuse light).
GPT-O4 and Grammarly : Translate and gramma check for this report.