CPT205 Ass2 Code Report

Student Name: Tonghui Wu **Student ID:** 2251076 **Date:** December 2024

Section1: Introduction

This report provides an overview of the key features, architecture, and interactive controls implemented in the OpenGL-based scene. The environment includes a furnished house interior, an embedded Snake mini-game, a Ferris wheel, a roller coaster track, and dynamic weather transitions. Each component demonstrates modular functions, lighting effects, texture mapping, and hierarchical modeling. Additionally, user inputs—both keyboard and mouse—allow for camera manipulation, character navigation, and smooth transitions between different gameplay modes.

Section 2: Keyboard and Mouse Interaction (Readme)

2.1 Keyboard Interactions

Function Used:

keyboard_input()	Managing keyboard controls for main and snake game scenes
mouse_input()	Handling mouse interactions for camera orientation

Instructions:

- **'H'/'h':** Activate spotlights focused on the game console, enhancing its visibility.
- **'W'/'w', 'S'/'s', 'A'/'a', 'D'/'d':** Move and rotate the character. **Hold *Shift* to run faster**, enabling quicker traversal of the scene.

  Please Walk Close To the Gameconsole(Rightside in house)
- **'T'/'t':** If **close enough to the console**, toggle between main scene and Snake game.
- ↑ ↓ ← →: Use **Up** ↑, **Down** ↓, **Left** ←, and **Right** → arrow keys to control snake's movement direction during the game, enabling control and navigation.
- **'R'/'r':** In Snake game mode, restart the game if currently in a gameover state.

  Press 'T' Again to Return to the main scene, enabling quick mode switching.
- **Left-Click & Drag:** Adjust the camera angle around the character.
- **'M'/'m':** Open/close door (behind man's initial start point, on the back wall).
- **'Spacebar':** Jump if not currently in mid-jump or falling, providing short aerial movement.
- **'O'/'o', 'P'/'p':** Zoom camera in or out, adjusting viewing distance for better scene.

  Hint: The following instructions You Need to Go Outside House:
  Please Close to the Ferris Wheel(Rightside in the whole scene)
- **'Y'/'y':** Turn on the Ferris wheel lights.

  Now, Please Close to the Flying Chair(On the left side of whole scene)
- **'X'/'x':** If **within the flying chair's range**, man will ride flying chair, and change to first-person view.
- **'B'/'b':** Gradually fade the sky from sunny to rainy conditions, also triggering rain and lightning effects.

Note: If the character **steps outside the ground boundary**, the scene simulates a **cliff fall**, causing the camera to **shake** briefly before resetting the character's position and restoring stability.

Section 3: Feature Design and Description

2.1 House and Internal Elements

2.1.1 Interior Furnishings

Functions Used:

renderSofa()	Renders a sofa using hierarchical modeling with cushions, armrests, and decorative elements.
renderCupboard()	Creates textured cupboard using texture mapping and smooth geometry for realism.
renderFloorLamp()	A floor lamp with **cylindrical components**, leveraging **smooth shading and lighting** techniques.
renderDesk()	Generates a **textured desk** with detailed supports and **additional objects** like a laptop, showcasing object placement and scaling.
drawPhoto()	A textured photo with **a wooden frame**, demonstrating a texture binding picture **StarryNight**.

Design Features:

- **Realistic Models:** Furniture pieces are modeled with attention to proportions and details, enhancing realism.
- **Lighting Effects:** Lamps emit light using emissive materials and blending, contributing to the ambiance.
- **Texture Mapping:** Appropriate textures are applied to furniture surfaces to simulate materials like wood and fabric.

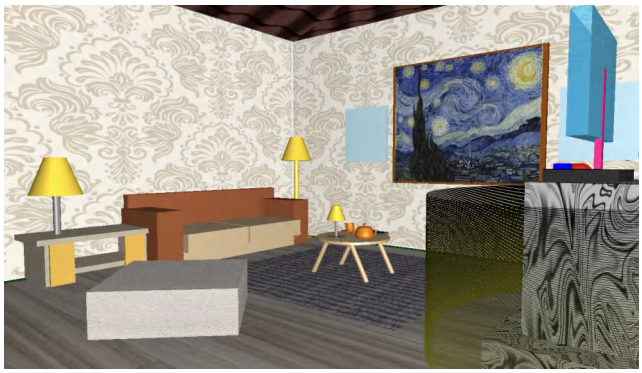

Fig.1: Room interior with detailed furniture.

2.2 Game Console and Snake Game

2.2.1 Game Console

Functions Used:

draw Game-Console()	Renders game console with textured surface, **semi-transparent screen**, **neon light** effects, and buttons using OpenGL primitives.
setup Spot-lights()	Rendering spotlights with **beam rendering** and **specular highlight**, focusing on console and **activated by 'H' key**, using **lighting techniques (cutoff angles and directional vectors)**.
draw Cool-Circle()	Creates a dynamic, **multi-layered circular glow** beneath the console, featuring **pulsating transparency and brightness** through time-based transformations.

Design Features:

- **Detailed Console Design:** Console with textured surface and modeled button to improve realism.
- **Emissive Screen:** The screen uses emissive materials to appear lit, enhancing authenticity.
- **Multi-Layered Circular Glow:** A pulsating circular glow beneath the console, featuring dynamic transparency and brightness.

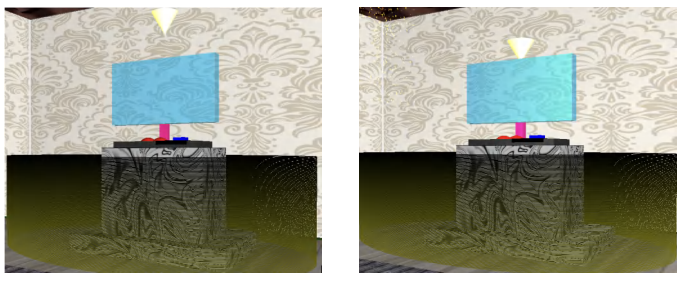

Fig.2: Game console with (right) and without spotlights (left)

2.2.2 Snake Game

Functions Used:(partial)

snake_display()	Renders game with **dynamic lighting**, **anti-aliasing**, and 3D transformations (grid, snake, food, particle).
snake_timer()	Updates **game logic**, including **snake movement**, collision detection, and particle effects for food consumption.
snake_draw_fireworks()	Renders fireworks with a **particle system**, **featuring transparency** and fading.
snake_draw_text_3d()	Creates **3D text** with **extrusion effects** for game status messages like "GAME OVER."

Design Features:

- **Embedded Mini-game:**Snake game is playable on the console within the scene, providing interactivity.
- **Particle Effects:** Eating food triggers fireworks with colorful, dynamic particle effects.
- **User Controls:** Keyboard inputs are mapped to control the snake, enhancing engagement.
- **3D Text Prompts:** Displays status messages like "GAME OVER" using extruded 3D text.

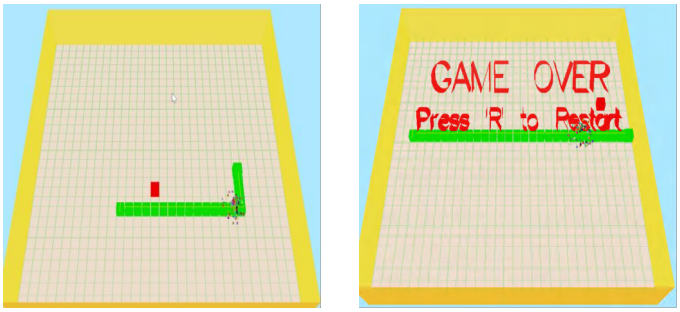

Fig.4: Snake game with grid, particle effects, 3Dtext prompts.

2.3 Ferris Wheel

2.3.1 Supporting Structure

Functions Used:

drawSupport Column()	Draws **textured cylindrical columns** with disks along **Y-axis**.
drawFound-ation()	Renders the **concrete foundation** using scalable cubes with **material** and optional **textures**.
draw Sup-ports()	Combines **support columns** and **foundation** into cohesive structure, ensuring **alignment** and **realism**.
drawSpokes()	Draws connecting **cylinders (spokes)** between the center and rim with **smooth shading** and optional **textures**.
drawInner Rings()	Renders **inner metallic rings** as textured toruses to enhance **visual detail** and **support integrity**.

Design Features:

- **Structural Realism:** The supporting structure is

designed to reflect real-world Ferris wheels, with detailed beams and joints.

- **Rotational Mechanics:** The wheel rotates smoothly around its axis, implemented with **transformation matrices**.
- **Gradient column colors:** Column are rendered with **HSV-based colors**, enhancing visual realism.
- **Material Properties:** Components utilize **ambient, diffuse, and specular** materials to interact with lighting realistically.

Fig.1: Gradient-colored Ferris wheel with realistic support structures..

2.3.2 Cabins

Functions Used:

drawCabin()	Renders **cabins** with a **swinging animation**, including main body, roof, and windows. Incorporates customizable colors and textures.
HSVtoRGB()	**Converts HSV values to RGB** to create **gradient colors** for cabins, enhancing visual diversity.
drawFerris Wheel()	**Positions cabins** around wheel, aligning them with spokes, managing their **rotational placement**.

Design Features:

- **Equidistant Placement:** Cabins are evenly distributed around the wheel, ensuring balance.
- **Gradient Coloring:** Each cabin features a unique color gradient, enhancing visual appeal and diversity.

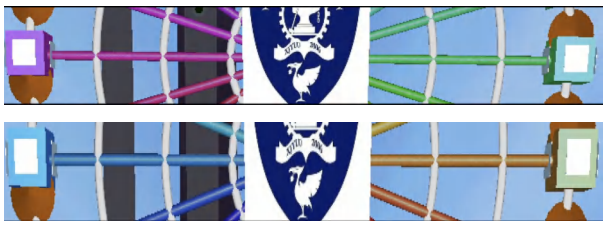

Fig.5: Cabin rotating along ferrywheel with gradient color

2.4 Flying Chairs

2.4.1 Rotating Structure

Functions Used (partial):

draw Turntable()	Draws turntable as a disk using texture mapping and **geometric transformations**, forming **base for rotating structure**.
drawCable()	Constructs cables as cylinders with **vector calculations** and **rotation** for realistic placement.
flyingChair()	Integrates all elements with **multi-axis rotations**, **oscillation**, and **dynamic animations**, showcasing animation techniques for rotating.

Design Features:

- **Rotational Motion:** The entire structure rotates around its **central axis**, causing chairs to swing outward, with combined **vertical oscillation** and **radial motion** for dynamic realism.
- **Multi-Axis Animation:** Combines **vertical oscillation**, **radial motion**, and **rotation** for a complex dynamic effect.

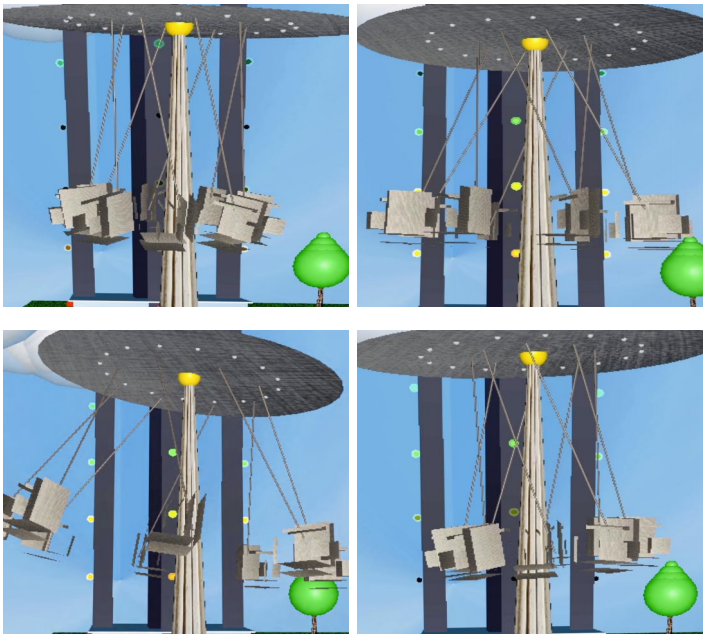

Fig.6: A series of rotations showing flying chair structure.

2.4.2 Chairs

Functions Used:

Chair()	Constructs chair using geometric transformations **(scaling, translation, rotation)**, **texture mapping** including **textured base**, **backrest**, **armrests**.

Design Features:

- **Swing Mechanics:** Chairs swing outward and sway due to the rotation and tilt of the ride.
- **Chain Simulation:** Chains are modeled to connect chairs to turntable, adding to realism.

2.5 Roller Coaster

Functions Used:

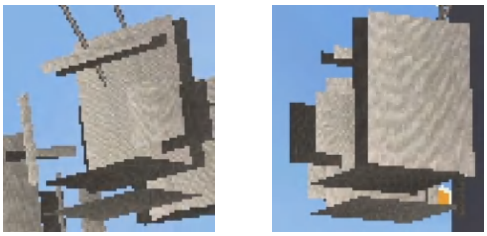

Fig.7: Structure of rotating chair.

initialize TrackPoints()	**Defines track controlling points**, organizing diverse segments like spirals,undulations using **linear transformation**.
catmull RomSpline()	Implements **Catmull-Rom spline interpolation** to generate curves between control points, maintaining track continuity.
draw CoasterTrack()	Renders the track using **quad strips**, applying **normals for lighting** and **texture mapping** to enhance detail and realism.

Design Features:

- **Diverse Track Structure:** Track includes **ascending climb** for height buildup, **spiral descent** using polar coordinates with twists, **high-speed undulating section** with fluctuations,**spiral ascent** with increasing height and twists, and **S-curve return** with lateral shifts, all designed with unique **geometric logic** for dynamic motion.

Fig.8: Truck structure of roller coaster.

2.6 Drop Tower

2.6.1 Tower Structure

Functions Used (partial):

drawFirst DropTower()	Renders droptower with base, central pillar, support pillars, and seating using **gradient coloring** and **textured materials**.

Design Features:

- **Animated Drop Sequence:** Simulates the free-fall motion with acceleration and deceleration phases.
- **Structural Realism:** Tower includes detailed components like base, frame, and mechanical elements.
- **Multiple Towers:** A second tower is implemented with variations to add diversity to the scene.

2.6.2 Decorative Lighting and Particles

Functions Used:

updateTower Lights()	Dynamically updates light using **sine wave** and height-based **HSV-to-RGB color transitions**.
drawTower Lights()	Renders glowing light using **additive blending** for realism.
updateTower Particles()	Manages the particle system, updating positions, velocities, and transparency with **gravity simulation**.
drawTower Particles()	Renders particles with **billboard techniques** and **alpha blending**, creating dynamic visual effects.

Design Features:

- **Dynamic Lighting Effects:** Lights flash and change colors, employing **sine wave functions** to produce smooth **flickering effects**.
- **Glow Effect:** Uses **blending** and **alpha transparency** to create a glowing aura around lights.
- **Synchronized Effects:** Lighting and particles **synchronize with tower's drop sequence** for immersive animations.

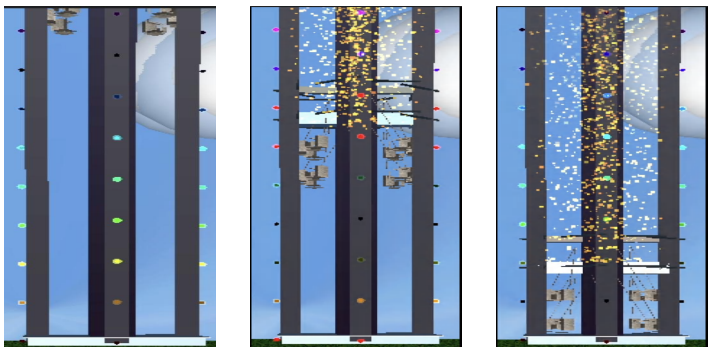

Fig.9 Lighting/particle effects during droptower movement.

2.7 Sky, cloud and Sky Transition

2.7.1 Sky

Functions Used:

drawSky Sphere()	Creates a seamless background by rendering **large textured mapping sphere**.

Design Features:

- **Camera-Centered:** The sky sphere is centered on the camera to prevent movement relative to the scene.

2.7.1.1 Clouds (Sunny clouds and Rainy cloud)

Functions Used:

initClouds()	Randomly generate cloud positions,shape **using multi-spheres**.
drawCloud()	Renders a cloud as **blended spheres** with dynamic **transparency**, adding **depth**.
drawSky Clouds()	Renders all clouds with **color** and **transparency** adjustments based on **weather conditions**.

Design Features:

- **Weather Adaptation:** Cloud **color** and **transparency** adjust based on **weather conditions**.
- **Layered Appearance:** Clouds are composed of overlapping **spheres** to create **fluffy shapes**.
- **Randomized Attributes:** Each cloud varies in **size** and **position** for **natural diversity**.

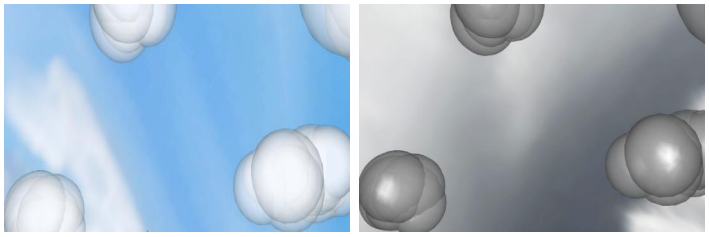

Fig.10: Sky and clouds transition based on weather condition.

2.7.2 Rain and Flash lightning in Rainy weather

Functions Used:

initRain Particles()	Initializes rain particles with **randomized positions and speeds**.
updateRain Particles()	**Updates positions** of rain particles over time.
drawRain Particles()	Renders rain particles using blending to create transparence.
generate Lightning-Bolt()	Generates lightning path with **random zigzag points** between sky and ground.
draw LightningFlash()	Render full-screen lightning effect **using alpha blending** .

Design Features:

- **Smooth Transition:** Gradually fades between sky textures to simulate weather changes.
- **Lightning Flash Effect:** A full-screen flash is triggered randomly to simulate a lightning strike, brightening the scene.

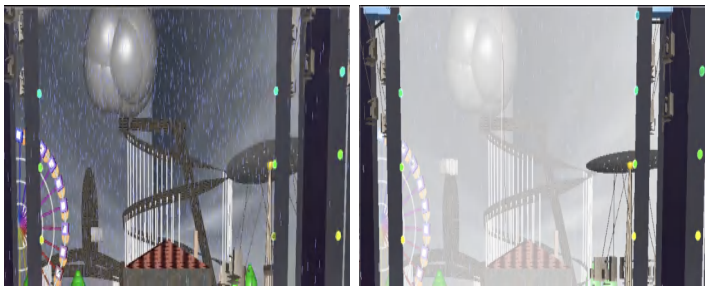

Fig.11: Rain particles and flash lighting in rainy weather.

2.8 Man

2.8.1 Structure of Man Body

Functions Used:

The **hierarchical modelling diagram** below is used to show the relevant functions of body parts of man

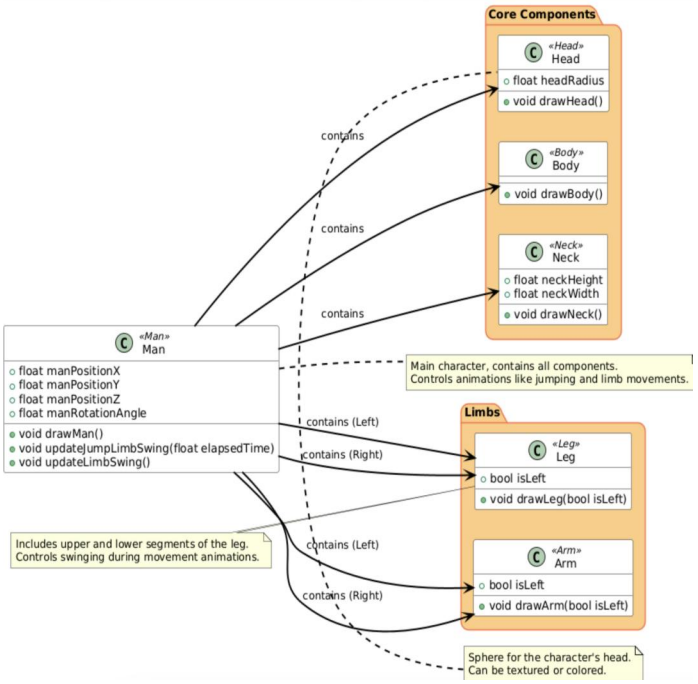

Fig.12: Hierarchical modelling diagram for body structure.

Design Features:

- **Hierarchical Structure:** The character's body is made of interconnected segments (**head**, **torso**, **limbs**), enabling coordinated movement.
- **Joint Articulation:** Each **limb** (**arms**, **legs**) rotates at joints, allowing for dynamic actions such as **walking**, **running**, and **jumping**.

2.8.2 Movement of Man

Functions Used:

update Limb-Swing()	Updates limb angles for **smooth walking animation** using **geometrical transformations** to simulate movement.
update JumpLimb-Swing()	Adjusts limb during **jumping animation**, relying on **kinematics and physics-based modeling**.

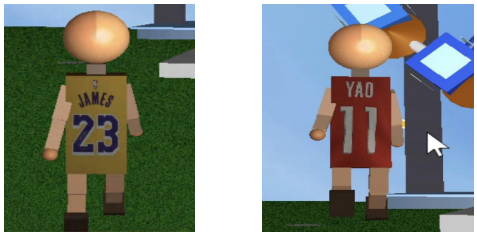

Fig.13: Walking/jumping movement of man in two clothes