Documentation

Group/Game Name: Shachō Samurai

Brief description of implementation:

Shachō Samurai is a first-person action game built using OpenGL for rendering and PhysX for physics simulations. The game features detailed 3D level and enemy models, interactive environments, and advanced gameplay mechanics. Players navigate through various levels, combatting enemies with a katana and interacting with the environment objects. The game is designed to maintain a minimum of 60 FPS, ensuring smooth gameplay and framerate independence. Key features include GPU Vertex Skinning Animation, dynamic lighting, realistic physics interactions, and a responsive control system.

Additional libraries:

- PhysX Physics Engine
- Assimp (Open Asset Import Library)
- STB Image
- IrrKlang

Gameplay:

Mandatory:

- 3D Geometry:
  - **Player's Katana**: A model of a katana used by the player character.
  - **Enemy Models**: Enemy character models with different animations. Character rigs and animation were sourced from Mixamo.
  - **3D Levels**: 3 different levels providing a rich environment for gameplay.
  - **Boxes, Tables, Lamps, Cigarette Packs, Burgers, Beer Bottles:** Static objects within the environment.
  - **Noodle Boxes, Balls, Coffee Mugs:** Various items scattered throughout the levels that players can interact with, thanks to physics integration.
- Playable:
  The game ensures a smooth playing experience by implementing a responsive control system. Players can easily navigate through the game using intuitive controls.
- Advanced Gameplay:
  Advanced gameplay mechanics include enemy AI that attacks and follows the player when in proximity and combat systems requiring skill and timing. The player and enemies have different fight attacks with different animations. The player can also block the enemy attacks.
- Min 60 FPS and Framerate Independence:
  The game reads configuration settings for window width, height, and refresh rate from a configuration file and sets the refresh rate to 60 Hz using GLFW's function, ensuring the display refreshes at 60 FPS. Fixed timestep for physics

calculations and interpolation for rendering are used, to ensure framerate independence. This approach allows the game to maintain consistent gameplay mechanics regardless of variations in frame rendering times.

- Win/Lose Condition:
  The game features clear win and lose conditions. Players win by defeating all enemies in a level. Losing occurs when the player's health reaches zero or falls to death.
- Intuitive controls:
  - WASD: Movement of Agito
  - Mouse: Camera view direction
  - LMB: Light Katana attack
  - Hold LMB: Heavy Katana attack
  - RMB: Block
  - Space: Jump
  - ESC: Quit
  - F1: Toggle wire-frame mode
  - F2: Toggle back-face culling
- Intuitive Camera:
  The game uses first-person camera system. The camera dynamically adjusts to the player's movements and actions to provide the best view for interaction and combat.
- Illumination model:
  The game uses an illumination model utilizing point lights.
- Textures:
  Free textures were sourced from Poly Haven and BlenderKit.
- Moving Objects:
  Moving objects in the game are handled through PhysX ensuring responsive interactions. Dynamic objects such as noodle boxes, balls and coffee mugs can be interacted with by the player, causing them to move and fall realistically.
- Documentation:
  The game includes an easy to understand in-game tutorial ensuring players understand controls, objectives, and gameplay mechanics.

Optional:

- Collision Detection (Basic Physics):
  Utilized PhysX for basic collision detection, allowing for realistic interactions between the player, enemies, and environmental objects. Player collides with floors, walls and enemy characters.
- Advanced Physics: Simulation of the physics of the player character (Jumping). Physics simulation for the enemy models upon being hit by the player's weapon. Player can interract with the objects placed in the level (noodle boxes, cups, balls), the objects fall, slide and bounce.

Effects:

Lighting:

- Omnidirectional Shadow Map:
  Implemented to provide realistic shadowing from light sources.

Animation:

- GPU Vertex Skinning:
  GPU Vertex Skinning was implemented to provide smooth and efficient skeletal animation for the enemy models and the player's weapon.

Texturing:

- Specular Map:
  Specular mapping was used to enhance the realism of the materials in the game. By using a specular map, we control the shininess and reflective properties of different parts of a surface, making materials look more lifelike.

Shading:

- Simple Normal Mapping:
  Normal mapping was used to add surface details to 3D models

Sound:

- Implemented attack, block, heavy attack, enemy damage, enemy death, enemy attacking, enemy hitting a block and player death sounds using irrKlang library.

Effects Implmentation:

The effects in the game were implemented using OpenGL, guided by tutorials from Learn OpenGL and ogldev.org. These tutorials provided detailed instructions and examples for various effects:

- **GPU Vertex Skinning**: Implemented using tutorials from Learn OpenGL and ogldev.org.
- **Simple Normal Mapping**: Implemented using tutorials from Learn OpenGL.
- **Specular Map**: Implemented using tutorials from Learn OpenGL.
- **Omnidirectional Shadow Map**: Implemented using tutorials from Learn OpenGL.

Walk-through:

1. **Starting the Game**: Launch the game and select 'New Game' from the main menu.
2. **Basic Controls**: Use WASD to move, mouse to look around, and left-click to use a fast attack with the katana, hold left click to use a heavy attack. Use right click to block and space to jump.
3. **First Encounter**: Approach the first enemy and practice combat mechanics.
4. **Exploring the Environment**: Interact with objects like boxes, tables, balls, noodle boxes and coffee mugs.
5. **Winning the Game**: Defeat all enemies to win. Wait 5 seconds for the level to change. Win all 3 levels to win the game.
6. **Losing the Game**: Avoid taking too much damage from enemies or falling down. If health reaches zero, the game will end.

Files:

GPU Vertex Skinning:

- Animation.h
- texture.vert

Simple Normal Mapping:

- load_model_meshes.h
- texture.frag

Specular Map:

- load_model_meshes.h
- texture.frag

Omnidirectional Shadow Map:

- Shader.h
- ShadowMapping.h
- shadow.frag
- shadow.vert
- shadow.geom

Advanced Physics & Collision Detection:

- Enemies.h
- First_Person_Camera.h

Model Sources:

- Enemy: Mixamo
- Burger: https://www.turbosquid.com/3d-models/hamburgers-1362306
- Beer Bottle: https://free3d.com/3d-model/-oz-beer-bottle-v2--198578.html
- Cigarettes: https://www.turbosquid.com/3d-models/morley-cigarettes-pack-1981546
- Hanging Lamp: https://www.turbosquid.com/3d-models/hanging-lamp-1178062
- Standing Lamp: https://www.turbosquid.com/3d-models/traditional-japanese-lantern-1697205
- Table: https://www.cgtrader.com/free-3d-models/interior/living-room/low-table-japanese
- Tori Gate: https://www.turbosquid.com/3d-models/torii-japanese-arch-1348816

Sound Sources:

- https://pixabay.com/de/sound-effects/slash-21834/
- https://pixabay.com/de/sound-effects/sword-slash-and-swing-185432/
- https://pixabay.com/de/sound-effects/manx27s-cry-122258/
- https://pixabay.com/de/sound-effects/male-hurt7-48124/
- https://pixabay.com/de/sound-effects/hard-punch-90179/
- https://pixabay.com/de/sound-effects/block-6839/
- https://pixabay.com/de/sound-effects/draw-sword1-44724/
- https://pixabay.com/de/sound-effects/dumbfounded-pain-102130/
- https://pixabay.com/de/sound-effects/glass-break-2-80964/
- https://www.tryparrotai.com/ai-voice-generator/donald-trump [narration voice, note: if it's not appropriate we can take it out before the upload of the game on the webpage :)]