Team 7 Game Milestone One: Skeletal Game

Team Members:

Shuoyi Ma	23162143	l0e0b
Xianchen Long	35526145	u7h0b
Dongan Liu	44846153	l1m0b
Sizhuang Liu	20091147	m4c0b
Guanting Li	36807155	n0k0b
Xin Shi	13151148	e7l0b

In this document we detail how the code aligns with the milestone requirements and the game proposal features. We have met all the milestone requirements and implemented the several creative parts. The development plan we made in the proposal was beyond the milestone requirements.

Game Proposal Features
Week: February 1 - Skeletal Game

[1 player, 1 map, movement, map boundary, movable enemy, minimal HP mana display.]

- Design -> assets start

  The design team finished designs for hero, enemy and projectiles. The map and grass texture were ready to use. The only thing we haven't designed is the icon for the game interface, which isn't required at this level of gameplay.

  Development plan in the proposal:

  - character design (hero & enemy)
  - map (single color)
  - icon
  - texture
  - projectiles

- Gameplay data start

  We have implemented all the player and enemy essential classes except for experience and cure, and we need more time to decide on skill classes. For skills and projectiles, we implemented fireballs and the player can shoot and kill enemies. However, We realized that skill class actually has more interaction with projectile class than what we originally thought. We are still discussing the relationship between them and thinking about how to link two classes together and keep the implementation consistent. For example, we need to think about if we should put damage and mana cost in Projectile class or Skill class. In addition, we can click "x" to close the game.

Development plan in the proposal:

- player enemy essential stats, classes(create, exp, move, kill, damage, destroy, cure).
- player basic control(move, attack)
- skill class with essential stats (damage, mana cost, cd, leveling up)
- projectiles(create, destroy, damage)
- Quit game

- Rendering & basic physics-> camera & collision start

  Our hero can walk around and the camera would follow the player. We implemented the collision detection. When our hero shoots fireball, the fireball will collide with the enemy and kill it. When enemy collides when our hero, our hero dies. We have yet to implement the map boundary. This week, our team put most of our focuses on world scaling, and on camera following the player. There were several setbacks when we realized our functions that we were studying were deprecated for OpenGL3.x, and we couldn't find a good way to access these legacy functions. We didn't implement "mesh" because we decided, for simplicity's sake in the time being (and to focus our efforts on cameras), to do mesh at a later time. The hero would still need mesh but, as we still have yet to implement our final hero design into our skeletal game, it does not make much sense to create the mesh early on either.

  Development plan in the proposal:

  - Camera focus on player & map boundary
  - Player at the center of the camera
  - Collision detection (monster & player, monster & monster, monster/player & boundary)
    - Gameplay? Could send sth over to gameplay side if something happens?
    - Reference salmon code for help
  - (no changing colors at the moment)
  - Changing camera size (zoom in/out)
  - Screen resolution will stay constant for this stage
  - Mesh: one for player, one for boss, one for all enemy
  - Reach: create brown and green background assets, figure out bounding box and zooming levels; momentum also a possibility

Milestone requirements

- Working application rendering code and shaders for background and sprite assets
  - Code renders sprites for enemy, hero, and fireball correctly, and shaders also compile for our program
- Loading and rendering of textured geometry with correct blending.
  - Enemy textures work properly in game
- Working basic 2D transformations.
  - Pressing "P" would zoom the screen in, and pressing "O" would zoom the screen out.

        - Our hero can walk around, and have the camera follow our player. The camera-following is smooth. Combined with our Creative Feature of zooming in/out, this makes for the backbone of our rendering.
- Keyboard/mouse control of one or more character sprites. This can include changes in set of rendered objects, object geometry, position, orientation, textures, colors, and other attributes.
    - pressing the W/S keys should make the hero swim move up and down and pressing the A/D keys should make the hero move left and right.
    - moving the mouse to the left/right rotates the hero clockwise/counterclockwise
    - Clicking the mouse to shoot fireballs
- Random or hard-coded action of (other) characters/assets.
    - Character: hero and enemy
    - Projectile: fireball
    - Environment design: grass tile
- Basic keyframe/state interpolation (smooth movement from point A to point B in Cartesian or angle space).
    - Currently the hero and projectiles moves in straight line and enemies constantly change their moving directions based on hero position. Straight lines can be considered as smooth movement. Since we don't have much animation at this stage, this requirement currently does not have a really obvious representation in the game.

- Stable game code supporting continuing execution and graceful termination.
    - The game can support continuing execution for more than 5 mins

Creative

- Projectiles(fireball) fired by hero at mouse direction.
- Kill enemies using fireballs.
    - Upon collision with enemy, both projectile and enemy is destroyed.
- Random respawn enemies on screen
- Zoom in and out using O and P key.
    - Applied transformations on "projection_2D", which resized the matrix appropriately.
- Background music changed