CSC 581 Homework 2 Writeup

Jae Jimmy Wong

1 Introduction

In this homework, I implemented two versions of the platformer game. The single-player version utilized multi-threading to handle object movements and game timelines. The multi-player version used not only multi-threading but also the ZeroMQ library to communicate between the server and clients. The game can handle an arbitrary number of clients and have synchronized a game environment between all clients.

2 Multithreaded Loop Architecture

As a prototype of a 2D platformer, the game has two distinct regular tasks: platform movement and character handling. By separating these tasks from the main rendering loop, the loop may iterate faster and create a smoother game-play experience. Putting platform movement into threads is pretty straightforward. The reference of the platform is passed as the argument of the threaded function. Since there is only one thread that updates the properties of the platform, data races between the main thread and moving platform can self-recover without explicit recovery code. This enables a lock-free structure of the moving platform and results in better rendering performance. Character handling has the same self-recovery property but it is much more complex in information passing. Keyboard inputs are tied to game windows and windows have to connect to the underlying operating system. The result is that reading keyboard inputs inside the threads directly can cause the game to crash. To avoid the problem, I created a data structure that stores pressed keys and shares them between threads with global variables. The character-handling thread controls the character based on this structure.

3 Measuring and Representing Time

To give `Timeline` the ability to represent real time, I set up a system clock as a static class member. When `Timeline` is initialized without an anchor, it returns the real time that comes from the system clock. This design is convenient for timeline users because real time and local time have similar syntax on initialization. The `Timeline` has two main features: pausing and speed changing. For the first one, I added one member variable `pause_time` to record the anchor's time while it is paused. Compared to the method of two variables, it is

more efficient on memory resources but slightly less intuitive. For the second one, I have the reference time updated each time the speed is changed to make sure there are no gaps in the timeline. These two features do interact with each other so planning is required.

4 Networking Basics

The setup needs two ZeroMQ sockets: one subscriber to receive join requests and one publisher to broadcast iteration messages. While receiving a join request, the server pushes a zero into the iteration list to represent the iteration count of the newest client. The list is then used to generate the iteration message, and the message is delivered to all clients in each server loop.

5 Putting it all together

The most difficult part in this section is the synchronization of the server and all clients. I found a simple approach to solve this problem, that is, the server regularly broadcasts the information about all objects on the screen. This approach is an application of the principle: single source of truth, since the server holds the state of all objects including the moving platform and characters. When a character is directed to move, the client will send the move command to the server and the server will calculate the new position of the character. Though this approach completely prevents synchronization-related problems, it makes the game less responsive in bad network conditions.

6 Asynchronicity!

The timeline class from part two is used in this section to calculate delta time. While paused, the delta time is always zero so the character will not move even if the player presses the movement keys. Also, the rendering steps are skipped on paused clients, we do not want the positions of the moving platform and other characters to be updated in the situation. When the client speed changes, the delta time changes accordingly thus the movements of the character speed up or down as expected.