Assignment 3 – Write up

CSC591 – Game Engine Foundations
Rachit Shrivastava

Part 1
For the purpose of timeline, I used help from the book and designed a TimeLine class which accepts a tick value and has a singleStep function in order to update time as per the tick size. This update method is called upon every draw call made by the PApplet, in order to constantly keep updating timestamp. The timeline starts using the System.nanoTime() as soon as a client is up and running.

Part 2
For the purpose of implementing an event management system, I had to add UserInput, Physics, CollisionDetection as some of my new components. In earlier assignment all of these with merely a function in GameClient class and were handled by just calling them of every frame. Having these components now will be useful to carry out similar functionality while implementing other games in Assignment 4.

Event Management System consists of an EventHandler interface which must be implemented by any component in order to handle an event.

Events here are registered to the EventManager during the constructor call to all the components.

EventType is an enum, which has mention of all the events in order of their priority. The list is huge, consisting of CollisionEvent, UpdationEvent, JumpEvent, InputEvent, SpawnEvent, DeathEvent, RecordEvent, ReplayEvent, etc. Having an enum in order to separate events would be a good choice for representation in order to avoid mix up by using a common oneventhandler function with hierarchies.

EventManager class here maintains a HashMap of all the registered events and their corresponding EventHandler classes and upon any event trigger, it maps a particular event as required. At any instant an event is triggered using the following code:

```
this.eventManager.trigger(new Event(EventType.UPDATE_POSITION_EVENT,
new ArrayList<Object>() {{
                     add(tl.getTimeCycles());
                     add(playerid);
                     add("update_position");
                }}));
```

Event argument consists of an arraylist with trigger timestamp and client ID, in order to keep a track of event. Any triggered event is then inserted into a priority queue which compares the priority firstly based on timestamp secondly based on the priority flag.

Part 3
Replay took a while for implementation. Buttons description has been mentioned in the PApplet, there are no actual buttons on screen but keyboard key press buttons.

- To start recording – key ‘a’
- To stop recording – key ‘s’
- To replay at 1x normal speed – key ‘d’
- To replay at 2x normal speed – key ‘f’
- To replay at 0.5x normal speed – key ‘g’

At the start of a replay in order to replicate the same timeline, I have anchored a new timeline from the point where the replay recording had begun. At the same time, there’s a backup of the GameObjects on screen created for that instance, in order to bring back the objects to place while replaying.

The replay recording stops when stop button is hit, and the user is allowed to move player to any desired location. Recording again may or may not work from now, sometimes you might get exceptions.

At any time when the player decides to replay, the game timeline is paused as soon user hits the replay 1x button ‘d’ and the replay timeline starts incrementing at every frame.

I observed an **issue for some scenarios** with my replay implementation, as discussed with Dr. Roberts after his lecture. Sometimes the replay directional jump behaves abnormally especially when playing in fast forward replay, the jump is incomplete and the player never makes it to the next platform, thereby ruining the entire sequences of replay events.

- The player movement happens at a constant rate per frame when user presses left and right key and continue to happen till the key is released.
- The program notes the keyPressed and keyReleased events with their respective timestamps for replay, but during replay at times it fails to behave as long as it was in actual gameplay.

The replay works perfectly fine on one client. Replay state would be mentioned at the top left corner of the game window.

Part 4
In my previous assignment code itself, I had used server merely as a medium to broadcast information to all the clients. All the positions of the player were being manipulated and calculated at each client and sent across the server.

I had thought that this design decision would be helpful in completing assignment 3, since now we had to send Events instead of positions in order to achieve Event Synchronization. In order to make this work, I had modified my code to send and receive events instead of player positions in the form of GameObjects. Somehow the event serialization didn't go through, as it depends indirectly on the EventManager which can't be made serializable. And I wasn't able to complete this task within time.
I've added my code for part4 as well, but it isn't functional.