Reflection

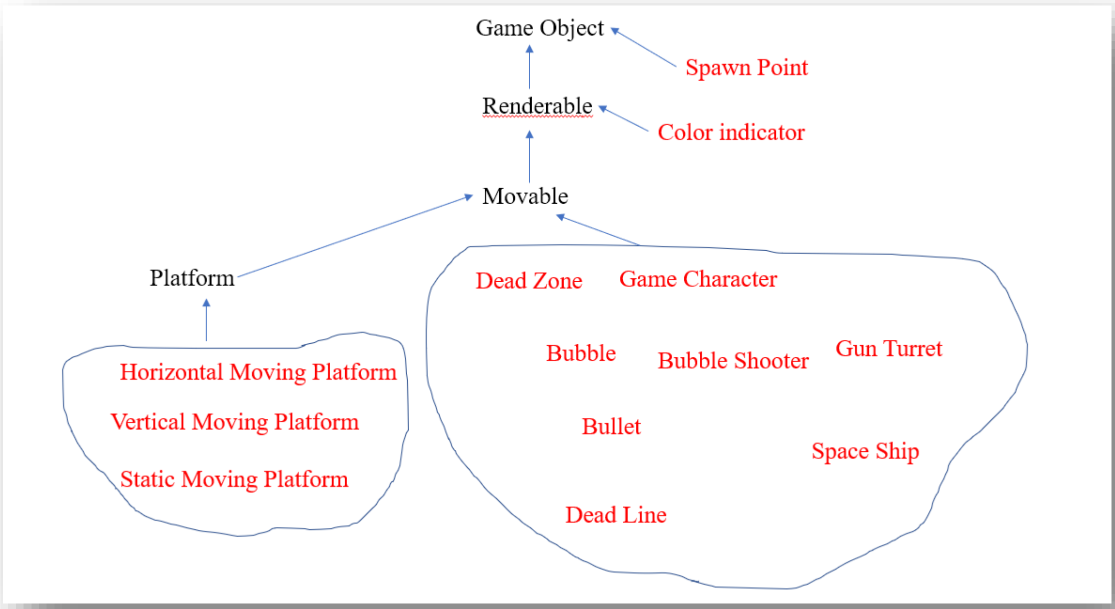
I have implemented the following components for my game engine:

- Game object system
- Network system
- Event system
- Time system
- Script system
- Replay system

I'll explain them in details:

- Game object system

My game object system is fairly successful. It allows me to create new types of game object without much effort. I implemented my game object model using inheritance. And the structure of my game object model looks like this:



In this picture, the black items are interfaces or abstract classes and red items are game objects. All the game objects used in all the three games I have made are included in this picture.

- Network system.

In my network system, server is designed to be able to accept incoming connections with clients asynchronously. Also, it communicates with established clients in asynchronized manner. To achieve this, when a server is set up, it will create a main thread for main game loop and a thread for accepting incoming network connections. Every time a client is set up, the client itself will create a main thread for handling data periodically, a thread for sending data out to server and a thread for receiving data from server. Meanwhile, the server will correspondingly set up a thread for sending data to this newly created client and a thread for receiving data from this client.

In this architecture, clients are dummy renderers whereas server handles all the logic and data in the game. Server sends the whole game world (in my design, this means a collection of game objects) to clients periodically and clients only send data to server when it's necessary (i.e. inputs from keyboard or mouse is generated). And this part is changed as I implemented the second game.

Originally, what clients send to server is A SINGLE integer number. For example, when a user press "space button", int 1 will be sent and when "UP button" is pressed, int 2 will be sent, etc. In this way client will only be able to send exactly one integer data to the server at a time. This is fine for the platform game. However, things get complicated when I was implementing the second game "Bubble Shooter". In this case, what I wish client to send is a pair of integers, x and y, indicating the mouse position. To solve this, I change what clients send to server into an object. This object can contains multiple information the clients wish to send. And on the server side, it need to decode the information in this object. In this design, clients can send whatever data they would like and get rid of the constraint of only one integer. This works well and no improvement need to be made for my third game.

- Time system

My time system works totally fine and no change need to be made for the second and third game.

For this part, I create a time system totally independent of other parts of game engine. I define a timeline interface which supports functions like: start, pause, un-pause, accelerate. "GlobalTimeline" class and "LocalTimeline" class implement this basic timeline interface. "GlobalTimeline" class is a timeline defined to be anchored to the real time (i.e. they are anchored to the time calculated by CPU cycles). And "LocalTimeline" class is a timeline anchored to another timeline.

In the implementation of all the games, I use these 2 kinds of timelines. In the setup process of the server, it first creates a Global Timeline anchored to the real time. Then, the server creates a Game World timeline anchored to the previously created Global timeline. All the game objects and event system are based on the Game World timeline.

In order to make the animation move accord to the timeline, game objects move according to the real tic number of a frame. And to get the tic number of a frame, I set a list in the frame loop. This list always maintains the lasted ten frame tics. And then we can speculate the tics of this frame by averaging the numbers in this list. This is a "guess strategy".

- Event system

My event systems works amazingly well and no change is need to be made for implementing the second and third game.

My event system has the following functionality:

- event representation
- event registration
- event scheduling
- event dispatch
- event handling

more detailed description can be seen in the write up in my homework 3

And in order to integrate a new event into the game, 4 steps need to be done:

- create a new event class (i.e. a new class implements the "event" interface)
- register the event in the set-up session in the server

- write the event handling code in the corresponding objects
- raise the event at appropriate time

- Replay system

My replay system is designed specifically for the platform game and it’s not applicable to other game. Fortunately, we don’t need to incorporate this functionality into the second and third game (confirmed by professor David in class). So, I didn’t take effort to change this one 😉.

Essentially, replay system is using the functionality in time system and event system.

- Script system

Script system is luckily and successfully built in a general way. No need to change this part when I was implementing the second and third game.

My script system has the following functionality:

- bind arguments
- load scripts
- execute scripts

And all the code given by professor David for this part is referenced.

Summary:

My game engine is built successfully at least for this course’s homework. No big change is made when I was implementing the second and third game.