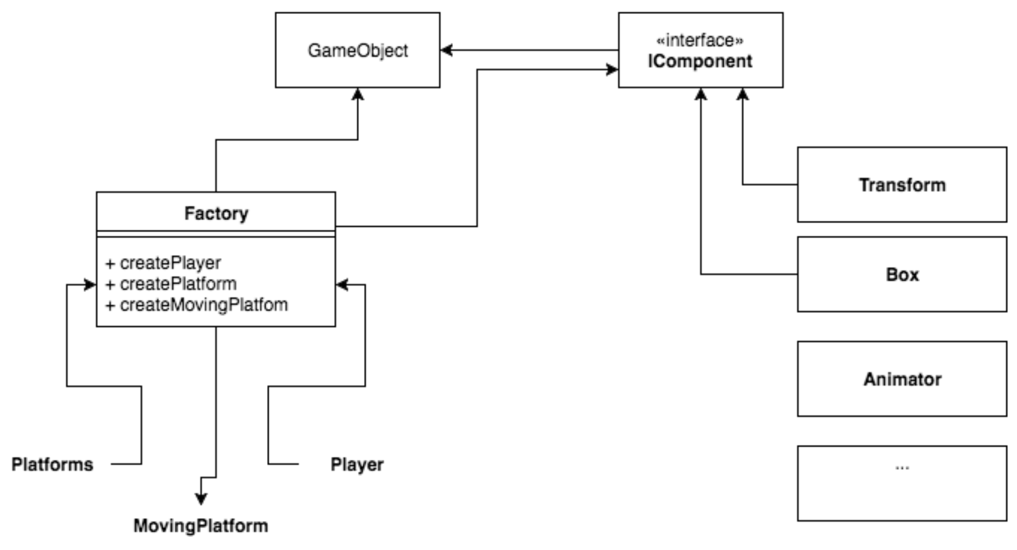
Assignment 2 – Write up

CSC591 – Game Engine Foundations
Rachit Shrivastava

Part 1: Game Object Model

- The model that I opted for software game object model is a generic component model. Since both platform and player GO here have many common components to be taken care about, so according to me keeping a list of components.
- At present I've created 3 components namely Transform, Box, Animate with the GUIDs corresponding to 1,2,3 respectively. They have some functions defined within them, which I would be updating as per the requirement for designing the Game Engine.
- IComponent is an interface to be implemented by every Component.
- The GameObject class contains a list of components associated with it. Components associated with the GameObject are stored in a HashMap.
- The Factory.java is where instances of GameObject are created and their corresponding components are assigned to them.
- Every time a client creates a GameObject I've kept a statically initialized counter to keep a track of game object created in a particular client. Any object will have a id like 1001, 2023, meaning 1001 is the first object of the first client and 2023 is the 23rd object of the second client.
- Components for a particular instance of GameObject can be obtained by knowing their key value from the HashMap which is equivalent to the enum that I have created for the component list.

Following can be considered as design of my engine's Game Object Model:



Note – this is just a sample of the component model, actual variables used in rest of the sections may be different.

Part 2: Multithreaded, Networked, Processing Sketch

- For part two, I'm sending GameObjects for communication between server and client.
- When the client connects, I've initialized the scene from server with a start position of the player (square) with a random color.
- Every client will have a different color so as to differentiate between them.
- This is where the single jump functionality works.
- When multiple client windows are open, keyboard events on one client will result same output on other screens.
- This is achieved by sending the current client's player gameObject to the server on key press, and the server in turn broadcasts this received gameObject to other clients.
- Closing of any client socket is not handled.
- Sending and setting the death zones, spawn points and the moving objects have been taken care in part three of the assignment. For this part only the player movement synchronization between clients was done along with setting up initial scene.

Part 3: A 2D Platformer

- The server here now sends the platform positions as well to every client connected along with the player init values.
- The "death zone" here is the ground. The scene has gravity pulling the square downwards.
- Collision detection is done for square and the platform, if the square sits on a platform its good. If there's no platform below the square it keeps on falling to the ground till it meets a platform.
- If it goes out of lower boundary, it is reset to its starting point "spawn point".
- As of now there's only one spawn and one death zone in the platformer.
- For later part I'm thinking of converting this death zones as a module using which I can make some of my enemy object as well as the floor boundary of
- The player can make a directional jump and switch directions while in air.
- Also there are no hard walls on the left and right side of the game window, the player comes out from the right if it exited from the left boundary.
- For accomplishing this, there was collision detection done at the client end between the platforms and the player.
- Spawn points and the death zone (since we have one of each type) are just the properties for the x-y coordinates of the player game object.
- I've a dynamic moving platform colored in red, which accepts the new positions from the server and then populate the same on every client.

- This platform has got a new Component called Animator which gives the Box for this GameObject a new X-coordinate when requested for. Animator component also has functions to change y coordinate of the corresponding Box component.
- This platform keeps on moving from left to right at a certain speed, but doesn't move the object along with it when it sits on top of the red platform.
- The platform moves at its regular pace, and if the player's square isn't moved it falls to the ground as soon as it isn't on the platform.
- Closing of any client socket is not handled. So if you close any client, you'll see a lot of exceptions on your console, just close all the clients and server and start over.

Part 4: Performance Comparison

- I wrote separate codes to check the performance of server client communication while sending strings and objects.
- I was quite amazed to see that object take a lot of time when compared to strings, I wonder is it because of the data structure or the memory. Or maybe strings are easy to be read byte by byte over network as compared to an object type.
- I've jotted down my readings in a tabular format mentioned below, it was quite cool to see that my MacBook successfully executed sending 500 gameObjects to 3 different clients over 1000 game loop iteration. Although it was heat up for a while but its fan made a loud noise making sending the object type bad for processor.
- But then again it was one system which ran 3 clients and 1 server over it for communication between them.

No. of Clients	String Communication (ms)	Object Size	Object Communication (ms)
2	857, 739		
		10	2702, 2577
		100	19166, 18751
		500	90042, 89369
3	758, 897, 939		
		10	2756, 2720, 2471
		100	23160, 23361, 22236
		500	109918, 110807, 109556 + noisy laptop