NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
SEMESTER EXAMINATION FOR
Semester 1 AY2012/2013

CS4344 Networked and Mobile Gaming

December 2012 Time Allowed 2 hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains FOUR (4) questions and comprises THREE (3) printed pages, including this page.
2. Answer **ALL** questions. State your assumptions, if any, clearly.
3. The total marks for this paper is A HUNDRED (100).
4. Answer all questions in the answer book provided.
5. This is an **OPEN BOOK** examination.

1. (15 points) When developing a multiplayer game, it is sometimes useful to wait within the game loop. For each of the scenarios below, explain why waiting is useful. Explain the factors that affect the calculation of appropriate waiting time for each case.
   (a) (5 points) In a client/server-based game, After receiving the event from the user input and sending the event information to the server, the client waits before simulating the game and updating the state locally.
   (b) (5 points) In a client/server-based game, After receiving the event from the client, the server waits before processing the event and simulating the game.
   (c) (5 points) In a point-to-point game, after receiving the event from other clients, a client waits before processing the event and simulating the game.

2. (15 points) Let $d(x, y)$ be the average network latency between two hosts x and y. We assume that:
$$\text{if } d(x, y) \approx 0, \text{then } d(x, z) \approx d(y, z)$$
That is, if two hosts are very close (in terms of latency) to each other, they are assumed to belong to the same network, and therefore their latency to a third host z is similar.
   Based on the assumptions above, and without using network coordinates, explain how you can reduce the number of RTT measurements (i.e., the number of probes) a client has to send in the game server discovery protocol. Pay attention to the actions of, and the interactions among, the master server, the game servers, and the clients.

3. (50 points) Consider the following multi-player car racing game called "Demand for Velocity". Each player controls a car. The objective of the game is to move the car from the starting line to the finishing line, along an open, curvy, road, in the shortest amount of time. Cars can collide. When this happens, the colliding cars move uncontrollably in the opposite direction of collision for a short period of time. Cars move slowly when it is off the road. The strategy of the game is thus to "bump" the opponent's car off the road to slow it down, while avoiding being "bumped" by the opponents, at the same time moving towards the finishing line as soon as possible, navigating through the curvy road.
   A player controls the car using two parameters: acceleration and the rotation of a steering wheel. The controls are sent to the server using the following messages:
   - `accelerate` α: where α is the current value of acceleration. A negative α decelerates the car.
   - `rotate` θ: where θ is the current rotation angle of the steering wheel.

   The current game design uses a "smart server, dump client" model, in which all simulation is done at the server. The server simulates the game and periodically (every 50ms) sends an `update` message to all the clients, containing the position of every car in the race. The client then renders the game based on the states received.
   Figure 1 depicts a rendered scene in the game, showing the view from a camera placed just behind, and following, the player's car. Cars are represented as boxes. The player's car is drawn with an arrow hovering above the car. The view always

shows the cars in front of the player, up to a certain distance, and the cars slight behind the player.

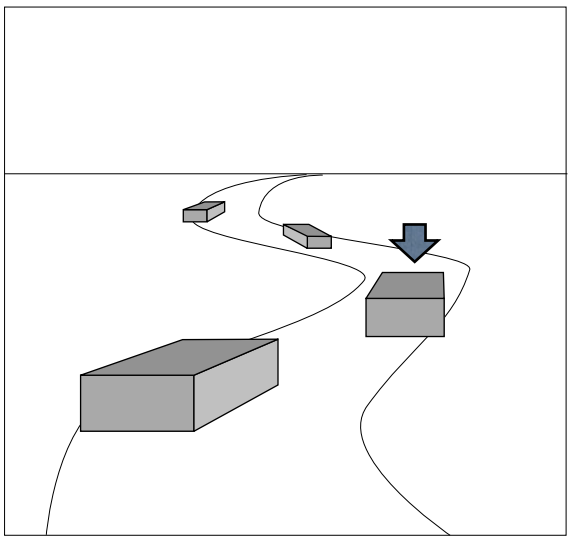

Figure 1: Screenshot of Demand for Velocity.

(a) (5 points) Do the messages in this game (`accelerate`, `rotate`, `update`) need to be sent reliably? Explain.

(b) (10 points) Suggest how you can reduce the size of `update` messages from server without incuring additional processing cost on the client and without affecting playability. Explain the additional overhead that your technique would incur on the server.

(c) (10 points) Explain how you can apply dead reckoning in this game with minimal effect on playability.

(d) (10 points) Describe the advantages and disadvantages of using dead reckoning as described in Part (c) above.

(e) (15 points) The client runs on a mobile device. Describe how you would change the implementation of the game so that the wireless network interface consumes less energy, with minimal effect on playabiity.

4. (20 points) For each of the following latency compensation technique, indicate if it can be used in cloud gaming to address synchronization/fairness issues caused by the latency between the thin client and the cloud. Justify your answer.

(a) (4 points) Local lag

(b) (4 points) Short circuiting

(c) (4 points) Artificial server delay

(d) (4 points) Time warp

(e) (4 points) Bucket synchronization

END OF PAPER