Questions and Exercises

These questions and exercises is an opportunity to see what you've learnt from the lecture as well as practice the new things we've been talking about. In other words, these questions and exercises are completely optional but it's recomended to do them. In the end of the document you will find the answers to the questions as well as possible solutions to the exercises, note that one can solve an exercise in different ways. There will also be some suggestions about what one could code if one want to continue with some more advanced things. These suggestions will not come with a possible solution and might include things that haven't been covered in the lecture.

Question 1
How does one check if the mouse is within a specific rectangle in the interface?

Question 2
If we want to synchronice data for an interface we can use the methods: addCraftingToCrafters, updateProgressBar and detectAndSendChanges. What's the idea behind these? How do we use them?

Question 3
When we used the mouseClickMove method to do things when the mouse moved we noticed a problem with our packets. What was that problem? What solutions are there to this problem?

Exercise
Continue with the exercise from the last lecture. Give the block a small buffer. If the buffer is empty, consume a cake to refill it completely. Draw this buffer in the interface. Instead of just placing a cake when the button is pressed, the cake on top should be regenerated once every third second. Display the timer in the interface(the timer should be controlled by the server though). The buffer should be used to regenerate the cake (so if half the cake is missing, replace it with a full one and drain half a cake from the buffer).

If you want to you can download a texture at the link below. Oveserve that the cake meter border has been moved compared to the last exercise.
https://dl.dropboxusercontent.com/u/46486053/CakeBoxTexturesLecture3.zip

Further explorations

Continue with the code from the exercise. Allow the user to set the generation speed from a few pre-defined values. Allow the user to set how big the buffer should be as well as allowing the generation process to go backwards(consume the cake to refill the buffer).

Answers and solutions

Answer to Question 1
To check if a specific point is in a rectangle one should check if it's to the right of the left edge, to the left of the right edge, below the top edge and above the bottom edge. Or in other words, inbetween the edges. This can be done with the following code

return x <= mouseX && mouseX <= x + w && y <= mouseY && mouseY <= y + h;

The mouse position itself is usually sent along in the parameters in the method of the interface. Whe can do so when we draw the background, draw the foreground or get a mouse click. However, you might need to think about guiLeft and guiTop of the interface. The mouse coordinate contains that value so if your rectangle don't, you'll have to keep that in mind. If your rectangle do contain the guiLeft and guiTop you don't have to bother about it.

Answer to Question 2
The idea behind synchronizing data through the container is to do so only to the clients that actually need the information, the clients with the interface open. The addCraftingToCrafters is used to send all the information to the client in the beginning, this is done when a player opens the interface. The detectAndSendChanges is used to continuously synchronizing data, this should however only be done when the data has actually been changed. To do this we'll have to store the old data to compare with, you must save that in the container. The updateProgressBar is used on the client side to receive the synchronized data.

Answer to Question 3
Sending the information from the client to the server and then back to the client takes a few ticks (can take longer depending on the connection). Therefore if we send a packet depending on a state we're changing, it might be sent multiple times (since the client won't receive the infromation to stop right away). To solve this we can either take a short-cut and set the information on the client side right away(we'll still have to tell the server though). This prevents us from sending more packets than needed, however if multiple players click at the same time we still might have an issue. The other solution is to send more information to the server, for instance which direction we want the value to change. If this is the case the server won't have a problem with receiving multiple packets, even if they are from different players. One can also combine these two solutions.

Possible solution to Exercise
https://dl.dropboxusercontent.com/u/46486053/CakeBoxSolutionLecture3.zip