Distributed Programming MOD006128 2018-19 Sem 2
Coursework Assignment
A Distributed Multiplayer Racing Game

Element 010 (100% of module mark)

The ultimate goal of this assignment is to develop a go-kart racing game which allows one person to drive his or her go-kart on one workstation and the other person to drive his or her go-kart on a second workstation while both individuals can view both karts on their respective screens in real time, the communication being managed by a multithreaded server. This will be a 2D game – the go-karts essentially race around a racetrack.

This assignment should to be developed in **three stages**. You can optionally work on this assignment during prac class time provided you first attempt any class specific exercises set in relation to the current weeks' topic. The tutor will offer advice (formative feedback) which may help you better develop your final solution. As a rule of thumb, try to complete and demo part 1 by the end of week 6, part 2 by the end of week 8, and part 3 by the end of week 11 before submission.

Part 1: Design your go-karts

Using an off the shelf image editing tool of your choice (eg GIMP) create a JPEG image, 50 pixels by 50 pixels, of a go-kart viewed from above. It will be important to choose an image where it is easy to tell the front from the back of the vehicle. You should create 16 versions of the image rotated around its center, i.e. one for every 22.5 degrees around the circle. Hint: create each new image from the original and NOT from a previously rotated image, as multiple rotations will have an accumulative effect on errors in the image.

You will need <u>two</u> go-kart images (different colours?) with the same 16 views. Make sure you can distinguish between the two.

Write a Java application that spins one go-kart at a designated rate. To do this create one class inheriting from a JFrame, and add another class which inherits from a JPanel implementing ActionListener. The customized JPanel will effectively be the 'palette' containing the image of the go-kart. An animation can be created by rapidly displaying slightly different views of an array of images (a 'sprite'). Recent versions of Deitel & Deitel's book Java: How to Program for example include a section on 'Animating a Series of Images' (if you have access, the book is also available for free via the Digital Library – a Quick Search using the terms >java deitel< should bring up the resource). General web searches should also provide sources. The recommended approach is to make each image a type of ImageIcon so that the JPanel's paintComponent() method can render the image using method paintIcon(). The animation should be driven using a Swing Timer class (not to be confused with a Timer class from the java.util package) that can repeatedly delay execution for a designated number of milliseconds, automatically calling paintComponent() at the end of each cycle via the ActionListener whose actionPerformed method calls method repaint(). *NB Do <u>not</u> develop a threaded solution using the Java Thread class* – use of the Swing Timer class is all that is needed.

Part 2: Develop the race track

In the second phase, create a Java application for the race track where you can 'drive' your *two* gokarts around around a central reservation on *one* workstation. The racing arena is in a window 850 pixels by 650 pixels and uses the following code to create the arena;

```
Color c1 = Color.green;
g.setColor( c1 );
g.fillRect( 150, 200, 550, 300 ); grass

Color c2 = Color.black;
g.setColor( c2 );
g.drawRect(50, 100, 750, 500);  // outer edge
g.drawRect(150, 200, 550, 300); // inner edge

Color c3 = Color.yellow;
g.setColor( c3 );
g.drawRect( 100, 150, 650, 400 ); // mid-lane marker

Color c4 = Color.white;
g.setColor( c4 );
g.drawLine( 425, 500, 425, 600 ); // start line
```

Select four keys for each of the two players. Two keys are used to change the direction of each go-kart. A key press turns the kart 22.5 degrees left or right *by changing the current image – I recommend you do not arithmetically rotate a single image during execution as it will create latency issues and does not naturally follow from the part 1 development.* The other two keys are to change the speed of the kart. Since speed = distance/time, in this context the motion of the car will be represented by pixel displacement/screen refresh. Use a scheme whereby speed is represented on a scale from 0 to 100. One press of a key either increases the speed by 10 or decreases the speed by 10. I recommend you develop *one* JFrame containing *one* JPanel which itself contains *both* the track and the two gokart images, but you will also need a KeyListener to implement the keypresses.

Write a Java application that allows two people to drive the two karts around the central reservation by sharing a common keyboard with different sets of keys. You may add additional graphical features to make it visually more interesting.

Factors to consider include how a change in direction is implemented, how a change in speed is implemented, how you might implement *collision detection* between a go-kart and the inner edge of the track or the outer edge of the track. Ideally in the event of a collision with such a barrier the kart's speed becomes 0 whilst the other go-kart proceeds, unless they collide with each other, in which case the game is ended. You could even associate sound effects with these events (eg. http://simplythebest.net/sounds/ contains some free sound clip files).

Part 3: Race the gokarts on two workstations

The idea here is to create a client/server application, i.e. a server and two clients where each player drives one kart on each client and races against the other player whose kart is also displayed on the corresponding client.

General issues:
At some stage you should test your application on a network; for example you could access other machines by simultaneously logging into different machines, either via multiple instances of your own login or with help from a colleagues' login. Load the server software on one machine and a copy each of the client software onto two other machines (if accessing via a colleagues' account, make sure you delete any files you have copied). Note it is entirely possible for testing purposes to undertake all development on one machine; for example run two clients and the server from three separate console windows, or perhaps with one of the instances from JGrasp.

Many issues must be considered when designing a client/server application; some *general* questions are listed below applicable to *any* client/server application (emboldened) followed by a comment specific to this project;

1. **What functionally is performed on the server? Does the server need to interact with a database or legacy software? Does it provide centralized services?**

   In this 3rd part you should use as much of part 2 code as possible. Therefore, the two clients perform most of the functionality. The server only provides the communication pipe between the two clients. Only the server needs to be threaded if you use a Swing Timer class in the client (which guarantees a regular screen update).

2. **What functionally is performed on the client? What user interface needs to be provided?**

   A user can only control his or her own go-kart with four keys. When one person drives his or her go-kart, the motion of the kart should appear on both workstations' screens. If the two go-karts collide, the game is over and both clients need to be notified. Allow either player to restart or exit the game from a menu.

3. **How many clients? And what categories of clients?**

   In this scenario we have two clients.

4. **What message protocols are needed? What does each type of message look like? What events cause a message? Does the application need a secure channel?**

   You will need to give careful thought on how to do this. Ignore the issue of security.

5. **How to get the application started? Is authentication of clients/users needed?**

   You will need to start the server first. Then allow either client to connect one after the other. After both clients have connected and the images of the go-karts are loaded, you will need a way to start the race.

6. **How to close the application down?**

   If a user exits in the middle of a race, the client should inform the server then the server can inform the other client to exit as well.

7. **How many applications can run simultaneously?**

   Here we assume only one game with two clients (2 go-karts racing) at a time.

8. **How to coordinate the server and the clients?**

   You can't have a client send a request and the server *not* respond. If you do, the application will lockup.

Assignment Submission

You should hand in;

1. Source and compiled code for all three parts in three separate folders on a removable media (CD or DVD or usb).
2. Your hard-copy documentation which should also be in three sections, each section reflecting the three parts of the assignment. Submit to the Cambridge iCenter (or via equivalent partner submission process) by *the last teaching day of Semester 2* (Friday 3rd May 2019). In each section present a brief description (about 300 words) evaluating any items of interest related to your development (eg go-kart graphics, arena/map detail, display update, implementation of networked version). Even if any part was not completed or unsuccessful the work should be critically appraised. Each section should also contain a print-out of the relevant code (landscape, courier new, 12pt, single line spaced). It is likely some of the code will be duplicated between parts 1, 2 and 3, however you do not need to duplicate the print out of such code (although it should be duplicated in the electronic on-disk versions). The brief descriptions for each section should overall sum to no more than 1000 words in total.
3. Instructions for set up and running each of the three applications (a couple of sentences in each case).

Reassessment

Should you be unfortunate enough to require a reassessment, please submit another attempt of this assignment by the standard resubmission date for Semester 2 2018-19 reassessments. You should consider developing alternative strategies from your first attempt to achieving a solution, if applicable.

Assessment and feedback

Personalized electronic feedback by email will summarize marks/grades using this table.

Component	Typical Criteria	Contribution	Mark
Part 1 Design the go-karts			
D	2-sets of 16 50x50pixel images created. No documentation.	20%	
C	Kart rotates at nominal speed, but noticeably fluctuates in size. Little design rationale. Instructions included.		
B	Kart rotates successfully. Clearly coded but with some code redundancy.		
A	Kart rotates smoothly. Excellent design rationale.		
Part 2 Develop the racing track arena			
D	Application displays track arena and gokarts but karts do not move. No instructions for running the program, No description of program limitations.	45%	
C	Application runs on one computer with one go-kart successfully manipulated with 1 set of 4- keys. Design rationale included.		
B	Application can be run on one computer with two go-karts manipulated with 2 sets of 4 keys. Good code layout with appropriate classes.		
A	Track enhancements, transparent background to go-kart images, collision strategies implemented (eg kart/kart; kart/central area; kart/outer area), sound effects, choice of karts/tracks. Description includes enhancements.		
Part 3 Race the cars on two workstations			
D	Server program developed but completely unfunctional. No rationale for design.	35%	
C	Two clients connect successfully to the server but unfunctional server threading with no communication exchange of go-kart data. No connection instructions. Well commented code.		
B	Application partly successful with 2 clients and server but with noticeable latency effects. User interface facilitates connectivity options.		
A	Application can be successfully run between three instances (2 clients and an efficiently threaded server). Excellent description/rationale of program.		
Comment:			
		Total / Overall Grade 010 =	