Group Members: Mohanish Sheth, Ken Hwang

Game topic: Racing Game

Basic Description:

We are designing a 2-player racing game, where each player controls a car that can move through a racetrack.  The racetrack will be designed by us, and will include the main road, as well as off-road conditons that are visible to the users.  We will also include obstacles to make the race track more difficult.  The car will display as either a normal car or in a "crashed" condition.  This means that if a user runs into an obstacle or another user, then the crash will be visible.  The goal of the game is to reach the finish line first, and your racing time will be recorded in the high-scores if it is fast enough.

Main Menu: New Game, View High Scores, Exit Game

The main menu will include an introductory graphic, and give the player some options to advance.  "New Game" will begin the actual gameplay, "View High Scores" will display the high scores stored in a SQL database, and "Exit Game" will terminate the game.

Game Contents:

The window will be placed in a JFrame component.  The introductory graphic will be placed on a custom class extending JPanel.  When a new game is started, a new custom panel will contain BufferedImages for the background, cars, and obstacles.  The boundaries for the race-track vs. off-road states will be stored as xy-coordinates in a data field and will affect the car’s movement if the car intersects in these regions.  Impact between cars and obstacles will refresh the car’s image to a “crashed” version of the car, and collision detection will take place by examining for intersection among xy-coordinates of the objects on the racetrack.  Cars will also be prevented from moving off the custom panel, by completely reducing movement if they reach the edge.  The custom panel will implement Runnable and ActionListener for a thread and user interface, respectively.

Our custom panel has an overridden paint() method that appropriately rotates (using AffineTransform) the car graphic according to the direction the car is facing in polar space.  Repaint() is called in our thread to continuously update the custom panel of the car location based on underlying variables containing the car’s speed and direction.  Additionally, frictional forces (either natural or enhanced due to off-road terrain) are consistently applied and updated to the car’s speed in our run() method for the Thread interface.

Key inputs will control the car’s ability to accelerate, decelerate, turn left, and turn right.  Acceleration and deceleration will affect the car’s speed, either increasing or decreasing its velocity.  Turning left or right will rotate the car counter-clockwise and clockwise, respectively, in polar space.   Afterwards, polar coordinates are converted to x- and y-coordinates for image placement on the panel.  Player 1 will use the Left, Right, Up, and Down arrow keys.  Player 2 will use the W, A, S, and D arrow keys.  Each user’s keys will appropriately affect their respective car.

Upon crossing the finish line, the duration of each car’s race time is recorded and applied to high scores if applicable.

Milestones:

1) Basic window display with car image  -- Complete (Mohanish/Ken)

2) Car movement and appropriate acceleration/turning physics-- Complete (Ken)

3) Runnable Thread with appropriate car location update and display-- Complete (Mohanish/Ken)

4) Environment boundaries, display, and application of frictional force (Mohanish)

5) Include obstacle detection and appropriate collision detection

6) Multiplayer extension

7) Timing finish time and finished race event

8) High score SQL database

9) Game finalization and deployment