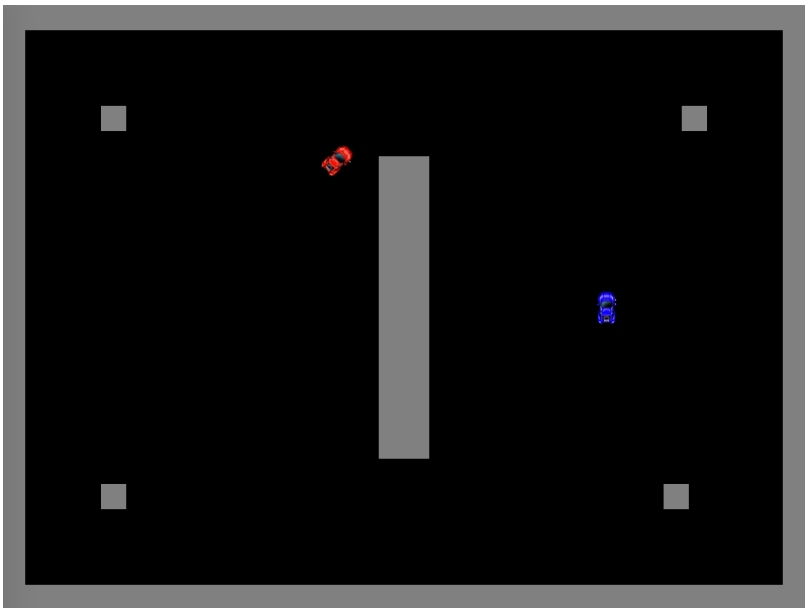
P1: **Project 1: Steering Behaviors**

The goal of this project is to learn how to implement **Steering Behaviors**, which are very commonly used in computer games for controlling cars, motorbikes, spaceships, etc.

In this project you will work with a very simple game engine written in Java that basically simulates cars moving around in a two-dimensional plane. You can see the source code from here: Steering-Java.zip.



In this "game", there is a scenario that contains a number of walls, and a number of cars. The goal is to make some controllers to make some of the cars behave in specific ways, using steering behaviors. Each car can be controller by controlling "steer", "throttle", and "brake" (like a regular car).

Specific Project Tasks:

- See the source code from here Steering-Java.zip
- Set the project in your favorite Java IDE (Netbeans, Eclipse, IDEA, command line, etc. up to you) (please do not ask the instructor for help with this, choosing your development environment is up to you).
- To test that you have it properly setup, you can try to run the file test.KeyboardExample. If you see a window launch with a red car that you can control with the arrow keys, you are good.
- Familiarize yourself with the code. Look at the engine.Game, engine.GameObject, and controllers.Controller classes, which are the ones you will be mostly handling.
- **Task 1, Seek**: locate the class test.SeekScenario. If you run it, you will see two cars, you can control one with the keyboard, but the other one does not do anything. Your task is to create a new controller (create a class "controllers.SeekController") that receives a GameObject, "target", in the constructor, and that controls the car to

continuously "Seek" the "target" using the "Seek" steering behavior. Apply this controller to "car2" (the blue car).

- **Task 2, Arrive**: locate the class test.ArriveScenario. If you run it, you will see one car and a green dot. Your task is to create a new controller (create a class "controllers.ArriveController") that receives a GameObject, "target", in the constructor, and that controls the car to "Arrive" to the "target" using the "Arrive" steering behavior. Apply this controller to "car1".
- **Task 3, Wall Avoidance**: locate the class test.WallAvoidanceSeekScenario. This time you need to do a Seek controller, but that also includes wall avoidance (as seen in class). The goal is to make the blue car "seek" the red car, but without bumping into walls.
- Create a short 3 minute video demoing your project, and send it to the instructor (do not send the video file! please just send a link (e.g., DropBox link, Youtube link, etc.). You will have to present this video in class.
- Turn in the source-code of your project, plus a 1-2 page description of what you did before midnight the due date. Submissions will be done via Blackboard

Notes:

- The goal of the project is to learn steering behaviors (plus output filtering), do not try to achieve the desired behaviors using any other approach than steering behaviors!
- To implement a new controller, just create a new class that extends the "controllers.Controller" class, and implement the "update" method. This method simply has to set the values of the three control variables (steer, throttle, brake). For example, if your steering behavior has decided that the car has to turn left, but only turning the steering wheel 50% of the way, accelerate full throttle and not brake, then it has to set the values like this:
  controlVariables[VARIABLE_STEERING] = -0.5;
  controlVariables[VARIABLE_THROTTLE] = 1.0;
  controlVariables[VARIABLE_BRAKE] = 0;
- If you need to perform any collision detection operation, notice that both GameObject and Game have a method that can check for collisions between GameObjects, or between GameObject and "RotatedRectangle"s. A RotatedRectangle is determined by its center coordinates, it width and height, and its rotation.
- Steering behaviors require heavy use of vectors. You might have to define your own vector class (implementing multiplication, addition, normalization, dot product, etc.).
- You will have to implement an "output filtering" class as we saw in class that transforms the output of the steering behaviors into the steer/throttle/brake commands for the cars.
- For the last task, you will have to implement your own "raycasting" method. Notice that this is actually quite easy given the functionality already provided by the engine. Specifically, notice that all game objects have a "collision" method implemented, and that the class "RotatedRectangle" can be used to test for collision with an arbitrary rectangle. So, to do a raycast, a simple way is to create a RotatedRectangle of the size of the car, rotate in the same orientation as the ray you want to cast, and then do a for loop moving the rectangle at uniform increments along the ray up to the length of the ray you want to cast. At each step you can check for collision with the obstacles in the game or with the other car.

Best!!