SE 3XA3: Module Interface Specification
GrateBox

Team 8, Grate
Kelvin Lin (linkk4)
Eric Chaput (chaputem)
Jin Liu (liu456)

December 7, 2016

Table 1: **Revision History**

Date	**Version**	**Notes**
Nov 10	1.0	Created MIS Document
Nov 12	1.1	Updated MIS to reflect new modules
Nov 13	1.2	Added the car module to MIS
Nov 13	1.3	Added title page and revision history
Dec 7	1.4	Updated MIS to reflect new decomposition of program

Module Interface Specification: GrateBox.js

Members

CAM_SPEED

This variable keeps track of the default speed of the camera in the x-direction.

Source: GrateBox.js, line 72

CAM_X_TRANSLATION

This variable keeps track of the default shift factor in the movement of the camera in the x-direction.

Source: GrateBox.js, line 66

camerax

This variable keeps track of the horizontal velocity of the camera.

Source: GrateBox.js, line 101

cameray

This variable keeps track of the vertical velocity of the camera.

Source: GrateBox.js, line 107

car

This variable holds the car model

Source: GrateBox.js, line 95

carsArray

This variable array contains the cars in the population cars.

Source: GrateBox.js, line 153

currentGeneration

This is the current generation that the simulation is in.

Source: GrateBox.js, line 173

currentMember

This variable integer indicates the current member of the group of cars.

Source: GrateBox.js, line 163

Contents

DEFAULT_CAM_X

This variable keeps track of the default shift factor for the camera in the x-direction.

Source: GrateBox.js, line 60

diffx

This variable keeps track of the change in the horizontal displacement of the camera.

Source: GrateBox.js, line 112

diffy

This variable keeps track of the change in the vertical displacement of the camera.

Source: GrateBox.js, line 117

DRAW_SCALE

This variable keeps track of the scaling factor of the display of the simulation.

Source: GrateBox.js, line 16

FILL_ALPHA

This variable keeps track of the alpha value of the display of the simulation.

Source: GrateBox.js, line 21

frameRate

This is the frame rate for the simulation.

Source: GrateBox.js, line 168

GRAVITY

This variable keeps track of the acceleration of gravity in the simulation.

Source: GrateBox.js, line 6

INTERVAL_RATE

This variable keeps track of how often the simulation updates.

Source: GrateBox.js, line 49

LINE_THICKNESS

This variable keeps track of the line thickness.

Source: GrateBox.js, line 26

MIN_NUMBER_OF_CARS

This variable keeps track of the minimum number of cars allowed in the simulation.

Source: GrateBox.js, line 78

MOVEMENT_THRESHOLD

This variable keeps track of the minimum amount that a car has to move per iteration in order for it to be considered moving.

Source: GrateBox.js, line 44

mutationRate

This variable indicates the rate at which mutations occur.

Source: GrateBox.js, line 148

NUMBER_OF_GENES

This variable keeps track of the number of genes that each car has.

Source: GrateBox.js, line 83

parentPool

This variable indicates the size of the pool from which parents creat offspring.

Source: GrateBox.js, line 143

paused

This variable represents whether the user has paused the simulation.

Source: GrateBox.js, line 179

points

This variable keeps track of points on a car

Source: GrateBox.js, line 90

populationSize

This variable indicates the size of the initial population of cars.

Source: GrateBox.js, line 138

POSITION_ITERATION

This variable keeps track of the timestep used to update the position in the simulation.

Source: GrateBox.js, line 38

proc1

This variable keeps track of the game loop thread.

Source: GrateBox.js, line 122

proc2

This variable keeps track of updateCar thread.

Source: GrateBox.js, line 127

TIMEOUT_RATE

This variable keeps track of the maximum lifespan of the car.

Source: GrateBox.js, line 54

topCars

This variable array contains the highest performing cars for the purpose of creating the next generation.

Source: GrateBox.js, line 158

VELOCITY_ITERATION

This variable keeps track of the timestep used to update velocity in the simulation.

Source: GrateBox.js, line 32

WORLD_SCALE

This variable keeps track of the scaling factor for the objects in the simulation.

Source: GrateBox.js, line 11

Methods

cameraPos()

This method sets the camera position to the position of the car.

Source: GrateBox.js, line 337

crossOverOffsprings(cars, topCars) → {Array.<Cars>}

This method crosses over the chromosomes of the offspring cars.

Parameters:

Name	Type	Description
cars	Array.<Cars>	The array of cars to crossover
topCars	Integer	The number of cars in the surviving parent generation

Source: GrateBox.js, line 377

Returns:

An array of the crossed-over cars

Type
Array.<Cars>

drawworld(world, context)

This method draws the world on the screen, before it is updated.

Parameters:

Name	Type	Description
world	b2World	The world to draw on
context	Canvas	The canvas to draw the world on

Source: GrateBox.js, line 323

getRandomArbitrary(min, max) → {Float}

This method generates a random floating point number between min and max, exclusive.

Parameters:

Name	Type	Description
min	Integer	The lower bound
max	Integer	The upper bound

Source: GrateBox.js, line 597

Returns:

A floating point number between min and max.

Type
Float

getRandomArbitraryInteger(min, max) → {Integer}

This method generates a random integer between min and max, exclusive.

Parameters:

Name	Type	Description
min	Integer	The lower bound
max	Integer	The upper bound

Source: GrateBox.js, line 584

Returns:

A random number between min and max

Type
Integer

init()

This method initializes the Box2D environment, and any objects within the Box2D world.

Source: GrateBox.js, line 190

Returns:

The created Box2D world

mutateOffsprings(cars, numberOfParents, mutationFactor) → {Array.<Cars>}

This method mutates the genes in the offspring's chromosomes.

Parameters:

Name	Type	Description
cars	Array.<Cars>	The array of cars to crossover
numberOfParents	Integer	The number of parents in the cars array
mutationFactor	Float	The likelihood of mutation

Source: GrateBox.js, line 418

Returns:

An array of the mutated cars

Type
Array.<Cars>

nextCar()

This method selects the next car to be simulated.

Source: GrateBox.js, line 263

partition(items, left, right)

This method partitions the array into two sets based on a pivot. The following code was modified from: https://www.nczonline.net/blog/2012/11/27/computer-science-in-javascript-quicksort/

Parameters:

Name	Type	Description
items	Array.<Cars>	An array of cars
left	Integer	The left index of the pivot
right	Integer	The right index of the pivot

Source: GrateBox.js, line 555

Returns:

The left index of the partitioned array

quicksort(cars, left, right)

This method preforms quicksort on an array of cars according to fitness value.
The following code was modified from:
https://www.nczonline.net/blog/2012/11/27/computer-science-in-javascript-quicksort/

Parameters:

Name	Type	Description
cars	Array.<Cars>	The array of cars to sort
left	Integer	The left index
right	Integer	The right index

Source: GrateBox.js, line 513

Returns:

The sorted cars array

resetCamera(world, context)

This method resets the camera for the next simulation.

Parameters:

Name	Type	Description
world	b2World	The world on which the camera is reset.
context	Canvas	The canvas to draw the world on

Source: GrateBox.js, line 308

resetWorld(world)

This method resets the world for the next simulation.

Parameters:

Name	Type	Description
world	b2World	The world to be reset.

Source: GrateBox.js, line 296

selectNextGeneration(cars, n) → {Array.<Cars>}

This method selects for the next generation of cars.

Parameters:

Name	Type	Description
cars	Array.<Cars>	The array of cars to choose from.
n	Integer	The number of cars to select for.

Source: GrateBox.js, line 356

Returns:

An array of the top n cars.

Type
Array.<Cars>

swap(items, firstIndex, secondIndex)

This method swaps 2 items in an array. The following code was obtained from: https://www.nczonline.net/blog/2012/11/27/computer-science-in-javascript-quicksort/

Parameters:

Name	Type	Description
items	Array.<Cars>	An array of cars
firstIndex	Integer	The index of the first car to swap
secondIndex	Integer	The index of the second car to swap

Source: GrateBox.js, line 539

update()

This method updates the screen.

Source: GrateBox.js, line 238

Module Interface Specification: CarObject.js

Members

CAR_FITNESS

This variable keeps track of the car's fitness

Source: CarObject.js, line 11

CAR_HEALTH

This variable keeps track of the car's health

Source: CarObject.js, line 6

MAX_WHEEL_RADIUS

This variable keeps track of the maximum radius a wheel can have

Source: CarObject.js, line 26

MIN_WHEEL_RADIUS

This variable keeps track of the minimum radius a wheel can have

Source: CarObject.js, line 21

NUMBER_OF_VECTORS

This variable keeps track of the number of vectors that a car has

Source: CarObject.js, line 16

NUMBER_OF_WHEELS

This variable keeps track of the number of wheels a car can have

Source: CarObject.js, line 31

VECT_X_LOC

This variable keeps track of the starting location of the X-vector on the car's chromosome.

Source: CarObject.js, line 37

VECT_Y_LOC

This variable keeps track of the starting location of the Y-vector on the car's chromosome.

Source: CarObject.js, line 43

Contents

WHEEL_POS_LOC

This variable keeps track of the starting location of the wheel position array on the car's chromosome.

Source: CarObject.js, line 49

WHEEL_RADIUS_LOC

This variable keeps track of the starting location of the wheel radi on the car's chromosome.

Source: CarObject.js, line 55

Methods

generateNewCar()

Method that generates new car randomly.

Source: CarObject.js, line 74

getCarDef() → {BodyDef}

Method that retrieves the definition of a car.

Source: CarObject.js, line 262

Returns:

The car's body definition

Type
BodyDef

getChromosome() → {Array.<Integer>}

Method that retrieves the specific chromosome of a car.

Source: CarObject.js, line 246

Returns:

The car's chromosome

Type
Array.<Integer>

getFitness() → {Integer}

Method that retrieves the fitness of a car.

Source: CarObject.js, line 270

Returns:

The car's fitness

Type
Integer

getHealth() → {Integer}

Method that retrieves the health of a car.

Source: CarObject.js, line 254

Returns:

The health of the car

Type
Integer

getVertexXArray() → {Array.<Integer>}

Method that retrieves the array of horizontal vertices of a car.

Source: CarObject.js, line 214

Returns:

The x-vertex array

Type
Array.<Integer>

getVertexYArray() → {Array.<Integer>}

Method that retrieves the array of vertical vertices of a car.

Source: CarObject.js, line 222

Returns:

The y-vertex array

Type
Array.<Integer>

getWheelPosArray() → {Array.<Integer>}

Method that retrieves the array of wheel positions of a car.

Source: CarObject.js, line 230

Returns:

The wheel position array

Type
Array.<Integer>

getWheelRadiusArray() → {Array.<Integer>}

Method that retrieves the array of wheel radiuses of a car.

Source: CarObject.js, line 238

Returns:

The wheel radius array

Type
Array.<Integer>

increaseFitness()

Method that increases the fitness value of a car by 1.

Source: CarObject.js, line 104

removeHealth()

Method that reduces the health value of a car by 1.

Source: CarObject.js, line 111

setCarDef(carDef)

Method that sets the definition of a car.

Parameters:

Name	Type	Description
carDef	Float	The set value of the ar's definition.

Source: CarObject.js, line 179

setChromosome(chromosome)

Method that sets the chromosome of a car

Parameters:

Name	Type	Description
chromosome	Array.<Chromosome>	The chromosome to be altered.

Source: CarObject.js, line 187

setFitness(fitness)

Method that sets the fitness of a car

Parameters:

Name	Type	Description
fitness	Integer	The fitness to be altered.

Source: CarObject.js, line 206

setHealth(health)

Method that sets the health of a car

Parameters:

Name	Type	Description
health	Integer	The health to be altered.

Source: CarObject.js, line 198

setVertexX(vertexXArray, i)

Method that sets a specific vertex in an array of the horizontal vertices to a specific value.

Parameters:

Name	Type	Description
vertexXArray	Array. <vertexXArray>	The array of vertices where the vertex is present.
i	Integer	The identify of the specific vertex in the array that is to be altered.

Source: CarObject.js, line 128

setVertexXArray(vertexXArray)

Method that sets the array of vertices in the horizontal.

Parameters:

Name	Type	Description
vertexXArray	Array.<vertexXArray>	The array of vertices to be set.

Source: CarObject.js, line 119

setVertexY(vertexYArray, i)

Method that sets a specific vertex in an array of the vertical vertices to a specific value.

Parameters:

Name	Type	Description
vertexYArray	Array. <vertexXArray>	The array of vertices where the vertex is present.
i	Integer	The identify of the specific vertex in the array that is to be altered.

Source: CarObject.js, line 145

setVertexYArray(vertexYArray)

Method that sets the array of vertices in the vertical.

Parameters:

Name	Type	Description
vertexYArray	Array.<vertexXArray>	The array of vertices to be set.

Source: CarObject.js, line 136

setWheelPos(wheelPos, i)

Method that sets the position of a specific wheel to a specific location.

Parameters:

Name	Type	Description
wheelPos	Array. <wheelPosArray>	The array that conatins the locations of the wheels.
i	Integer	The identify of the specific wheel position to be set in the array.

Source: CarObject.js, line 154

setWheelRadius(wheelRadius, i)

Method that sets the radius of a specific wheel.

Parameters:

Name	Type	Description
wheelRadius	Array. <wheelRadius>	The array that conatins the radiuses of the wheels.
i	Integer	The identify of the specific wheel radius to be set in the array.

Source: CarObject.js, line 163

setWheelRadiusArray(wheelRadiusArray)

Method that sets array of wheel radiuses to a specific array.

Parameters:

Name	Type	Description
wheelRadiusArray	Array. <wheelRadius>	The array that conatins the locations of the wheels.

Source: CarObject.js, line 171

Module Interface Specification: MakeCar.js

Contents

Members

JOINT_MAX_TORQUE

This variable keeps track of the maximum torque that a joint can support.

Source: MakeCar.js, line 56

JOINT_SPEED

This variable keeps track of the maximum speed that a joint can support.

Source: MakeCar.js, line 61

NUMBER_OF_VERTICES

This variable keeps track of the number of verticies needed to define a polygon.

Source: MakeCar.js, line 11

POLYGON_DENSITY

This variable keeps track of the density of a polygon.

Source: MakeCar.js, line 16

POLYGON_FILTER_GROUP_INDEX

This variable keeps track of the filter group applied to the polygon.

Source: MakeCar.js, line 26

POLYGON_FRICTION

This variable keeps track of the friction of a polygon.

Source: MakeCar.js, line 21

POLYGON_RESTITUTION

This variable keeps track of the restitution of the polygon.

Source: MakeCar.js, line 31

WHEEL_DENSITY

This variable keeps track of the density of a wheel.

Source: MakeCar.js, line 36

WHEEL_FILTER_GROUP_INDEX

This variable keeps track of the filter group applied to a wheel.

Source: MakeCar.js, line 46

WHEEL_FRICTION

This variable keeps track of the friction of a wheel.

Source: MakeCar.js, line 41

WHEEL_RESTITUTION

This variable keeps track of the restitution of a wheel.

Source: MakeCar.js, line 51

WORLD_SCALE

This variable keeps track of the scale of the world for the objects in the simulation.

Source: MakeCar.js, line 6

X_SCALE

This variable keeps track of the scaling factor in the x-direction.

Source: MakeCar.js, line 66

Y_SCALE

This variable keeps track of the scaling factor in the y-direction.

Source: MakeCar.js, line 71

Methods

drawCar(world, WORLD_SCALE, vertex1X, vertex1Y, vertex2X, vertex2Y, vertex3X, vertex3Y, vertex4X, vertex4Y, vertex5X, vertex5Y, vertex6X, vertex6Y, vertex7X, vertex7Y, vertex8X, vertex8Y, frontwheelPos, rearWheelPos) → {b2BodyDef}

This method creates a car to the screen.

Parameters:

Name	Type	Description
world	b2World	The Box2D world where the car will be placed in
WORLD_SCALE	Integer	The scaling factor for the Box2D world
vertex1X	Integer	The x-coordinate of the first vertex
vertex1Y	Integer	The y-coordinate of the first vertex

Name	Type	Description
vertex2X	Integer	The x-coordinate of the second vertex
vertex2Y	Integer	The y-coordinate of the second vertex
vertex3X	Integer	The x-coordinate of the third vertex
vertex3Y	Integer	The y-coordinate of the third vertex
vertex4X	Integer	The x-coordinate of the fourth vertex
vertex4Y	Integer	The y-coordinate of the fourth vertex
vertex5X	Integer	The x-coordinate of the fifth vertex
vertex5Y	Integer	The y-coordinate of the fifth vertex
vertex6X	Integer	The x-coordinate of the sixth vertex
vertex6Y	Integer	The y-coordinate of the sixth vertex
vertex7X	Integer	The x-coordinate of the seventh vertex
vertex7Y	Integer	The y-coordinate of the seventh vertex
vertex8X	Integer	The x-coordinate of the eighth vertex
vertex8Y	Integer	The y-coordinate of the eighth vertex
frontwheelPos	Integer	The vertex that the front wheel is attached to
rearWheelPos	Integer	The vertex that the back wheel is attached to

Source: MakeCar.js, line 309

Returns:

The completed car

Type
b2BodyDef

makeCarJoints(world, bodyA, bodyB, wheelPosX, wheelPosY) → {b2RevoluteJointDef}

This method creates joints used to connect the wheels to the car chassis.

Parameters:

Name	Type	Description
world	b2World	The Box2D world where the joint will be placed in
bodyA	b2BodyDef	The first object to connect the joint to
bodyB	b2BodyDef	The second object to connect the joint to
wheelPosX	Integer	The x-coordinate of the wheel
wheelPosY	Integer	The y-coordinate of the wheel

Source: MakeCar.js, line 256

Returns:

The joint connecting bodyA to bodyB

Type
b2RevoluteJointDef

makePolygon(num, vertex1X, vertex1Y, vertex2X, vertex2Y)

This method creates a polygon given the x and y coordinate of 2 vertices, and it joins them at the origin.

Parameters:

Name	Type	Description
num	Integer	The ID of the polygon
vertex1X	Integer	The x-coordinate of the first vector
vertex1Y	Integer	The y-coordinate of the first vector
vertex2X	Integer	The x-coordinate of the second vector
vertex2Y	Integer	The y-coordinate of the second vector

Source: MakeCar.js, line 83

makeWheelFixture(world, car, wheelbodyDef, wheelFixture) → {Body}

This method connects the wheel to the car chassis.

Parameters:

Name	Type	Description
world	b2World	The Box2D world where the wheel will be placed in
car	b2BodyDef	The car to connect the wheels to
wheelbodyDef	b2BodyDef	The body (physics) definition of the wheel
wheelFixture	b2FixtureDef	The shape definition of the wheel

Source: MakeCar.js, line 277

Returns:

The wheel

Type
Body

makeWheelShape(world, WORLD_SCALE, radius) → {b2FixtureDef}

This method creates the shape of a wheel for the car given its radius.

Parameters:

Name	Type	Description
world	b2World	The Box2D world where the wheel will be placed in
WORLD_SCALE	Integer	The scaling factor
radius	Float	The radius of the wheel

Source: MakeCar.js, line 235

Returns:

The shape of the wheel created

Type
b2FixtureDef

Module Interface Specification: Path.js

Members

HEIGHT

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 41

NUMBER_OF_VERTICES

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 6

ORIGIN_X

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 26

ORIGIN_Y

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 31

SLOPE1

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 46

SLOPE2

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 51

SLOPE3

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 56

Contents

SLOPE4

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 61

SLOPE5

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 66

TILE_DENSITY

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 11

TILE_FRICTION

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 16

TILE_RESTITUTION

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 21

TILES_PER_PART

This variable keeps track of the number of tiles per part of the road.

Source: Path.js, line 71

WIDTH

This variable keeps track of the scale of the world for the objects in the simulation.

Source: Path.js, line 36

Methods

connecttile()

This method connects the tiles to each other in a sequential fashion starting from the first tile at the origin. The road is split into 5 components, each with a different slope. An array of random numbers is used in order to create the illusion of randomness, while enforcing comparability between cars of different

generations.

Source: Path.js, line 120

createtile(point1X, point1Y, point2X, point2Y, point3X, point3Y, point4X, point4Y)

This method creates a tile for the road.

Parameters:

Name	Type	Description
point1X	Integer	The x-coordinate of the upper right hand vertex
point1Y	Integer	The y-coordinate of the upper right hand vertex
point2X	Integer	The x-coordinate of the lower right hand vertex
point2Y	Integer	The y-coordinate of the lower right hand vertex
point3X	Integer	The x coordinate of the lower left hand vertex
point3Y	Integer	The y-coordinate of the lower left hand vertex
point4X	Integer	The x-coordinate of the upper left hand vertex
point4Y	Integer	The y-coordinate of the upper left hand vertex

Source: Path.js, line 85
