Final Project

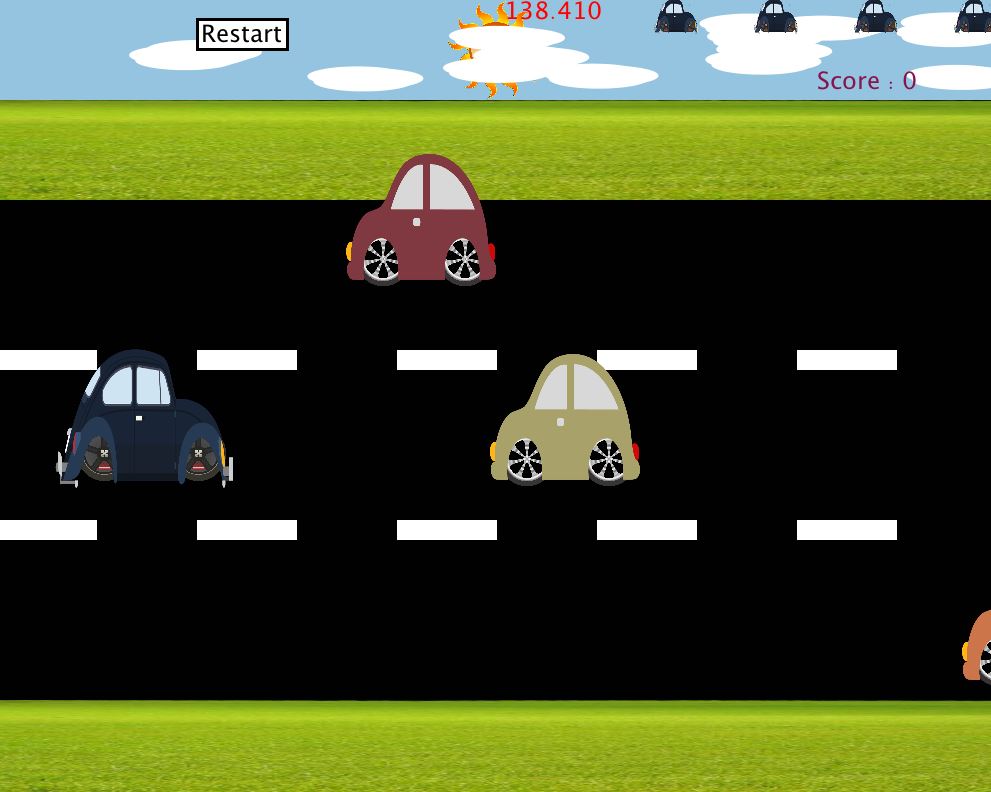
Programming Foundations  
Final Design Document  
**Game-**Vehicle Traffic Game   
By Aman Wangde  
Northeastern University

Prof: Velda Shaby  
June 12, 2017

# Concept of the game

The concept of this game is relative to the vehicle traffic game that can be found online but the main idea here is that the car or bike would tackle around the obstacles such as other cars to be alive. The game will have 4 lives where the user can play 4 times. The speed of the car will increase as the car moves on but if hit the car or any other obstacle, the user will lose its life. The game will also have the level up concept where the user can increase its life when passed through it. The start of the game will have user control brief up so that the user will be able to know on how to play the game. The music at the background which will make the user more focused towards the game. The user has to survive throughout the end if the user loses it lives than the game will get over with score he made on display. This game will also have an object which would boost the score of the game. This game is the simple arcade game with the scores dynamically on display. The traffic in this game will be inversed with objects mentioned above.

This game will use all the processing concepts needed for the project.



# How it works

* This game will have the option to choose night or day background before starting of the game.
* The clouds will be moving statically in the upper screen with respect to x and y axis.
* There will be a timer at the top of the screen. The game will only run about 160 millis.
* The car will move on the screen on the basis of arrow keys up, down, right and left.
* The score of the game will have a boost of 10 points on collection the coins which will be in between the game.
* The obstacles which car can lose it lives are bomb, stones, inverse cars.
* The functionality of the game that executes this game. It consists mainly loops and its mainly classes like Car, Background, Clouds, ReverseCars, HittingObstacles, ScoringObjects, Health.

**The detailed description of the classes is as follows:**

##### **Car - Class**

The purpose of this object is to display the car object to the screen based on the road positions. This will be accessible as the user can access the movements of the car. The car will have the following properties which will be used to handle the car as follows:

* **CarBody ()** –This function will create the car body of the class. The horizontal look of the main car which will be functioned by the user.
* **CarWheels ()** - This function will create the wheels to car which will be able to rotate based on movements.
* **Display () –** This function will be able to display the complete image of the car. The code for example will be:

**Display ()**

**{**

**CarBody () //** This will display the body of the Car.

**CarWheels () //** This will create the wheels of the Car.

**}**The above whole code gives the basic understanding to display the Car as an object to the screen.

* **CarWheelsRotate () –** This will help in building car rotating movements.
* **CarRun () –** This function will control the car movements with the help of code below

**CarRun ();**

**if (key==CODED) {**

**if (keyCode==RIGHT) {**

**CarWheelsRotate (); //** Rotate the wheels.

**} else if (keyCode==UP) {**

**CarWheelsRotate ();**

**} else if (keyCode==DOWN) {**

**CarWheelsRotate ();**

**}**

**}**

* **Move () –** This function will be able to control the movements of the car. It contains the control of the running movement of the car based on its subset function above. The code for example will be:

**Move ()**

**{**

**CarRun () //** Control the car Movements.

**}**

##### **Background - Class**

The purpose of this class is to display the background. The background will carry the multiple weather condition properties, Road and the grass with trees on it. The properties of the background are as follows

* **Summer Display ()** - This will display the summer display i.e. clear weather with sun on the top of the screen.
* **WinterDisplay ()** – This will display the snowflakes when selected.
* **Road ()**- This will create the road using rectangles with the divider that will divide the car. The idea is to show the road in motion when the car will be in motion. The code will be as follows:

**Road (int x)**

**{**

**Running speed will be assigned globally in the background class**

**x=some value**

**if (key==CODED) {**

**if (keyCode==RIGHT) {**

**x=(x+runningSpeed);**

**} else if (keyCode==UP) {**

**x=(x+runningSpeed);**

**} else if (keyCode==DOWN) {**

**x=(x+runningSpeed);**

**} else {**

**x=some value;**

**}**

**}**

**}**

* **GrassUp () –** This function will help in grass display with trees on it on the upper side of the road.
* **GrassDown () –** This function will help to display the Grass on the lower side of the Road.
* **Snow Up () –** This function will help in snow display on the upper side of the road.
* **SnowDown () –** This function will help to display the Snow on the lower side of the Road.
* **SnowParticles () –** This will create snow particles.

##### **Clouds - Class**

This will be the first one to display on the screen. This can be created using Array objects and are displayed in the motion based on constant speed flow. The clouds will be display randomly based on the width of the screen and height allocated statically.  
  
The properties of the clouds are as follows:

* **Display () –** This will display the ellipses used to display the clouds based on random x and y positions.
* **Move () –** This will make the clouds move in the random position but based on the width. The code will be as follows:

**If (x>width)**

**{**

**x = width-somevalue.}**

##### **Reversecars- Class : There will be same classes for 3 cars in opposite direction**

This class will generate the cars in reverse order to that of the main car. This will show the cars in reverse traffic order. The properties will be the same as the above car has but it will be different in variations and the color of the car. The properties are as follows

* **ReverseCarBody ()** –This function will create the car body of the class. The horizontal look of the main car which will be functioned by the user.
* **ReverseCarWheels ()** - This function will create the wheels to car which will be able to rotate based on movements.
* **Display1 () –** This function will be able to display the complete image of the car.  
  This will be displayed randomly. The code for example will be:

**There will be three display of different cars.**The above whole code gives the basic understanding to display the Car as an object to the screen.

* **ReverseCarWheelsRotate () –** This will help in building car rotating movements.
* **ReverseCarRun () –** This function will control the car movements.
* **Intersect ()-** This function will help in understanding if the car object hits the object or not. The outer cover of the Car will have transparent rectangles. The following code will help as follows:

**Boolean intersect (Box rect1, Box rect2) {**

**if ((rect1.topLeftX > rect2.bottomRightX)**

**|| (rect1.bottomRightX < rect2.topLeftX)**

**|| (rect1.topLeftY > rect2.bottomRightY)**

**|| (rect1.bottomRightY < rect2.topLeftY)) {**

**return false;**

**}**

**else {**

**return true;**

**}**

**}**

* **Move () –** This function will be able to control the movements of the car. It contains the control of the running movement of the car based on its subset function above. This function will carry constant speed for the reverse car and will be appeared randomly.  
  The code for example will be:

**Move ()**

**{**

**ReverseCarRun () //** Control the car Movements.

**}**

##### **Scoringobjects- Classes: There will be 3 classes with respect to three lanes**

The purpose of this class is to display and move the scoring objects in reverse order. This help user score more points than normally.

* **CoinDisplay ()** – This function will display the coin like object.
* **Display ()** – This function will display the one up which will increase lives of the user
* **Move ()** – This function will help moving it with the speed of x in reverse order and when hit by car +10 points scored.
* **Intersect ()-** This function will help in understanding if the car object hits the object or not. The following code will help as follows:

**Boolean intersect (Box rect1, Box rect2) {**

**if ((rect1.topLeftX > rect2.bottomRightX)**

**|| (rect1.bottomRightX < rect2.topLeftX)**

**|| (rect1.topLeftY > rect2.bottomRightY)**

**|| (rect1.bottomRightY < rect2.topLeftY)) {**

**return false;**

**}**

**else {**

**return true;**

**}**

**}**

##### **Health- Class**

The purpose of this class is to display the health of the car as every time hit or scored. The main idea is if the life is 4 and even if you scored 1up the health will still be five only increases when the live isn’t full.

**Display () –** This function display the shapes and the colors use to display the lives in the LHS of the screen.

##### **Button- Class**

This class will be used generally for the buttons everywhere in the project. This class will be generic and can be called when required. The properties are as follows:

* **isClicked () –** Checks whether the button is clicked or not. If clicked, then the Boolean function returns true.
* **Display () –** It is used to display the button.

##### **Screen- Class**

This will have the screen class and display the screens within the loop of the game. This will be the main as the most of the draw function would be handled.

The properties are:

* **startScreen () –** This will display the screen at the start of the display.
* **nextScreen () –** This will display the Instructions for the game.
* **nextScreen1 () –** This will display the Hint for the game.
* **theSummerDisplay () –** This will add summer effects to the play.
* **theWinterDisplay () –** Thus will add snowy effect to the play.
* **winScreen () –** This will display the screen on winning of the screen.
* **LoseScreen () –** This will display the screen after losing.

// Intersect Code for ReverseCar1

boolean intersect(Car C, ReverseCar1 RC1) {

if ((C.box.topLeftX > RC1.b1.bottomRightX)

|| (C.box.bottomRightX < RC1.b1.topLeftX)

|| (C.box.topLeftY > RC1.b1.bottomRightY)

|| (C.box.bottomRightY < RC1.b1.topLeftY)) {

return false;

} else {

return true;

}

}

// Intersect Code for ReverseCar2

boolean intersect(Car C1, ReverseCar2 RC2) {

if ((C1.box.topLeftX > RC2.b2.bottomRightX)

|| (C1.box.bottomRightX < RC2.b2.topLeftX)

|| (C1.box.topLeftY > RC2.b2.bottomRightY)

|| (C1.box.bottomRightY < RC2.b2.topLeftY)) {

return false;

} else {

return true;

}

}

// Intersect Code for ReverseCar2

boolean intersect(Car C2, ReverseCar3 RC3) {

if ((C2.box.topLeftX > RC3.b3.bottomRightX)

|| (C2.box.bottomRightX < RC3.b3.topLeftX)

|| (C2.box.topLeftY > RC3.b3.bottomRightY)

|| (C2.box.bottomRightY < RC3.b3.topLeftY)) {

return false;

} else {

return true;

}

}

// Intersect Code for Coin

boolean intersect(Car C2, ScoringObject1 S1) {

if (S1.b4!=null)

{

//println(S1.b4.bottomRightX);

if ((C2.box.topLeftX > S1.b4.bottomRightX)

|| (C2.box.bottomRightX < S1.b4.topLeftX)

|| (C2.box.topLeftY > S1.b4.bottomRightY)

|| (C2.box.bottomRightY < S1.b4.topLeftY)) {

return false;

} else {

return true;

}

}

return false;

}

// Intersect Code for Fuel Points

boolean intersect(Car C3, ScoringObject2 S2) {

if (S2.b5!=null)

{

//println(S2.b5.bottomRightX);

if ((C3.box.topLeftX > S2.b5.bottomRightX)

|| (C3.box.bottomRightX < S2.b5.topLeftX)

|| (C3.box.topLeftY > S2.b5.bottomRightY)

|| (C3.box.bottomRightY < S2.b5.topLeftY)) {

return false;

} else {

return true;

}

}

return false;

}

// Intersect Code for Coin

boolean intersect(Car C4, ScoringObject3 S3) {

if (S3.b6!=null)

{

//println(S2.b5.bottomRightX);

if ((C4.box.topLeftX > S3.b6.bottomRightX)

|| (C4.box.bottomRightX < S3.b6.topLeftX)

|| (C4.box.topLeftY > S3.b6.bottomRightY)

|| (C4.box.bottomRightY < S3.b6.topLeftY)) {

return false;

} else {

return true;

}

}

return false;

# Main game Loop

The main processing game loop the window, creates the necessary game objects, and controls the logic and timing of the game. The functionality is explained in detail as follows:

**Setup ():**

1. Sets the window size to 1000 x 800 pixels.
2. All the classes will be created as mentioned above (Car, Background, Clouds, ReverseCars, HittingObstacles, ScoringObjects, Health).
3. The ReverseCar Objects are instantiated and well functioned with different display.
4. The Array is created which holds up the cloud as well as the Health of the car at the top of the screen.
5. It imports the file for the music which will be used at the start of the screen before the game starts, during the game etc.
6. Creates the Font object that will be used to display the texture w.r.t the example game loop.

**Draw ():**

1. The draw function is use to test the functions will be created like **Win or Lose**. This functions will checked that the user has lost the game lost all his health or he survived the game within the given timer. If the player loses the game the screen will be appeared to replay or exit. And if the player wins the game with the score than the screen will appeared saying Your score do you want to play again or exit.
2. The game actually begins when the above functions like **Win or Lose** is false. The game will start as given below:

* It draws all the classes and the objects which is used to create the designs for the game.
* It will have functions like **pushmatrix () and popmatrix ()** which will be control the image positions in the screen.
* Based on the screen class the weather condition will be displayed on the main gaming screen.
* The car object will display on the screen with is properties like **move ()**
* The player will use the **keyPressed () function** to play as this will help moving the car positions in the game. The code explained is below:

**if(key==CODED) {**

**if(keyCode==UP) {**

**y = y- somevalue used to go up;**

**if (y<some value)**

**y=some number;**

**}**

**else {**

**y= y + some value to go down.;}**

* The timer will be created in the game as somewhere the game has to stop as it will run forever. This will control the fast movements of the car and other obstacles.
* The Array of Health is given with the specific value as it will carry only 4 lives.
* The Array of Clouds will have a loop to display the cloud randomly on the screen.
* The classes are instantiated now calling on the main vehicles in the direction opposite direction to the car like ReverseCar Objects with **display ()** and **move ()** properties, HittingsObstcales objects with again **display () and move ()** and same with the ScoringObjects Class.
* ReverseObject class will have cars with different colors too.
* Object like Car will have an intersection method this will be tested to understand if car hits any other obstacle and if hit with Scoring Objects than it will increase the score or lose it life.
* The scoring methodologies will be every point taken will have +10 points but if you hit the obstacles might lose -10 points too at the same time.
* The shorten () method will be used for decreasing the health of the car.