

# ISOM

## Vector Rally: Final Project



av

**Tor-Jarle Sagen**  
[tor@itu.dk](mailto:tor@itu.dk)

**Lars-Martin Tollefsen**  
[tollar@itu.dk](mailto:tollar@itu.dk)

*link to project source code is*  
<http://itu.dk/people/tor/isom/VectorRally4.zip>



**The IT University**  
of Copenhagen

**Kandidatutdanning – SWU**

**Autumn 2004**

**Pages : 31**

# Table of Contents

Vector Rally Implementation.....	2
General Description.....	2
Functional Description.....	3
Zoom effect.....	3
Requirement 0.....	3
Installing a .sis file on Nokia 6600.....	3
Requirement 1.....	4
Naming Conventions.....	4
Requirement 2.....	5
Car and Obstacle classes.....	5
Requirement 3.....	5
Key input.....	5
Requirement 4.....	6
Reset menu.....	6
Requirement 5.....	6
Move counter.....	6
Requirement 6.....	6
Finishing the race.....	6
Description of design.....	7
The game board.....	7
The oversized game board .....	7
The complete path.....	7
Car movement and collision detection.....	7
Principle of the MoveCar method.....	7
Zoom effect.....	8
Vector Rally Classdiagram.....	9
Vector Rally Sourcecode.....	10
CAR.H.....	10
CAR.CPP.....	11
OBSTACLE.H.....	15
OBSTACLE.CPP.....	15
HELLOWORLDPLUSAPPLICATION.H.....	18
HELLOWORLDPLUSAPPLICATION.CPP.....	19
HELLOWORLDPLUSAPPUI.H.....	19
HELLOWORLDPLUSAPPUI.CPP.....	20
HELLOWORLDPLUSAPPVIEW.H.....	21
HELLOWORLDPLUSAPPVIEW.CPP.....	23
HELLOWORLDPLUSDOCUMENT.H.....	25
HELLOWORLDPLUSDOCUMENT.CPP.....	26
HELLOWORLDPLUS.CPP.....	27
HELLOWORLDPLUS.PREF.....	27
HELLOWORLDPLUS.RESOURCES.....	27
HELLOWORLDPLUS.MMP.....	28
HELLOWORLDPLUS.RSS.....	28
HELLOWORLDPLUS_CAPTION.RSS.....	30
HELLOWORLDPLUS.PKG.....	30

# Vector Rally Implementation

## General Description

Vector Rally is a mathematical simulation game where the challenge is to drive the car as fast as possibly through the course with many turns and obstacles. The simulation models the position and speed of the car with acceleration and slowing down. In the illustration bellow the basics of Vector Rally is explained. The Car starts at position  $(0,0)$  and speed vector  $(0,0)$ . For each iteration the car get a new position by adding its speed vector to its position vector with option of adjusting by  $\pm 1$  in each dimension. In the illustration we see that the car has made a bad start and is bound to hit the obstacle in front of it.

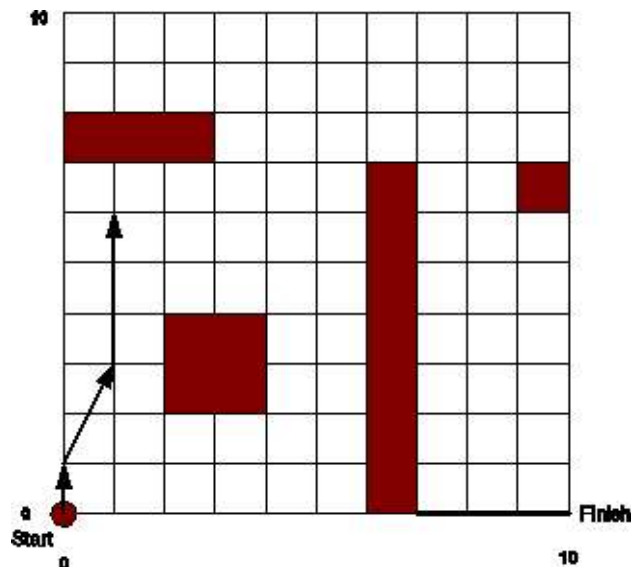
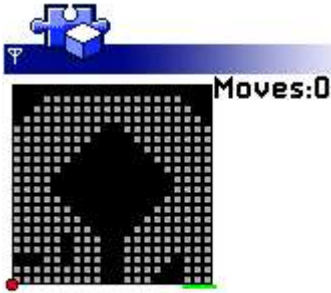


Illustration 1 Vector Rally grid

We used the HelloWorldPlus template from the CodeWarrior's stationary wizard. With this code framework we modified the code and incorporated our own code into it. The consequence of this it the HelloWorldPlus file names and classes.

## Functional Description

### Zoom effect

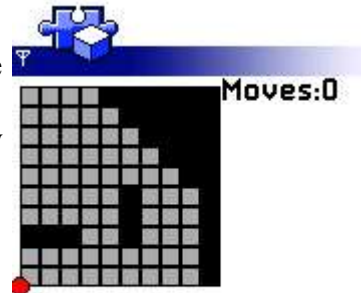


Pos: 0, 0 Speed: 0, 0  
Options

Illustration 2 Zoom out effect

We chose to implement the zoom effect, where the motivation was based on the following.

When the player plays on a large track, the screen of the phone becomes too small. With zooming functionality we can zoom out and in. In this way the player can zoom out and get a full overview of the total course. In the illustration to the left we see the zoomed out course. To the right we see the normal play view of the game, when it's zoomed in.



Pos: 0, 0 Speed: 0, 0  
Options Exit

Illustration 3 Zoomed in effect



Credits  
Zoom in  
Zoom out  
Reset game  
Exit

Select Cancel

Illustration 4 Zoom out selected

The player can invoke the zoom anytime in the game by pressing the Options button and choose Zoom Out.

To the left we see how the menu is chosen. When pressing the Select button, an info box will appear with "Zooming out!" and the whole course is displayed. When the player is in zoom out mode, it is not possible to move the car. To get back to play mode, the player must choose the Zoom In on the menu. The normal view will be back, and the car can be moved again.

## Requirement 0

### Installing a .sis file on Nokia 6600

We compiled with the ARMI UREL in CodeWarrior, and after building the armi release version we needed to package up the application components into an installation package (.sis) file. From the command prompt where the HelloWorldPlus.pkg is located (sis folder). We typed :

```
makesis HelloWorldPlus.pkg
```

After the successful build, we transferred the sis file with bluetooth to the Nokia 6600. We then installed the game to the main memory of the phone. The VR4 icon then could be found in the standard menu on the mobile. The game was played and tested on the phone and it worked in the same way as it did on the EPOC emulator.

## Requirement 1

### ***Naming Conventions***

We have followed the correct Symbian variable naming as stated in the “Symbian OS C++ for Mobile Phones” text book. The naming conventions we have used is the ones listed below:

- i-prefix for member variables.
- a-prefix for method parameters.
- C-prefix for classes that should be constructed on the heap.
- T-prefix for classes that can fit on the stack.
- L-postfix for methods that can leave.

Here is a few snippet examples showing name conventions in our code.

```
// i-prefix for member variable.
if(iZoom) ...

void Ccar::CDrawBoard(CWindowGc& aGC) const { ...
// a-prefix for method parameter and a C-prefix for class on the heap.

// l-postfix for methodes that can leave.
Ccar * Ccar::NewL() {
    Ccar * self = new (ELeave) Ccar();
    CleanupStack::PushL(self);
    ...
}
```

## Requirement 2

### Car and Obstacle classes

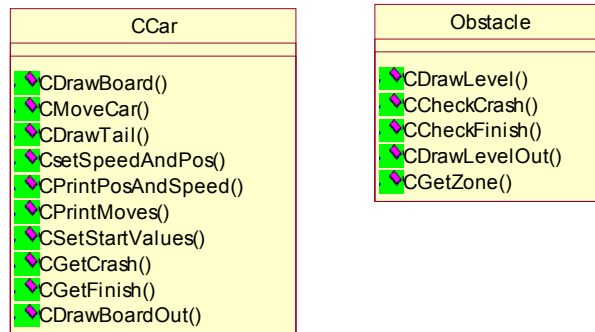


Illustration 5 Car and Obstacle class

We made a car and obstacle class, where the booth classes get created using NewL. We obtained pointers to both classes and they get put on the CleanupStack for protection against memory leaks. The NewL for the obstacle class is shown below.

```
Cobstacle * Cobstacle::NewL()
{
    Cobstacle * self = new (ELeave) Cobstacle();
    CleanupStack::PushL(self);
    self->ConstructL();
    CleanupStack::Pop(self);
    return self;
}
```

## Requirement 3

### Key input



Illustration 6 Keypad for car movement

In Vector Rally the car is controlled with the numeric keys 1-9 on the mobile phone. Pressing 2 will make the car go upwards, and add 1 to the speed. Pressing 5 will move the car in the last moved direction without increasing the speed. To select the menu for reset, zooming etc, you press the usual menu button under the Options or Exit on the screen.

## Requirement 4

### Reset menu

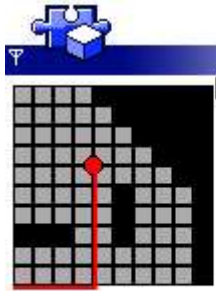
In the menu we have added a Reset game menu item, that when selected will reset the game back to its initial state. We have also made a Credits item that shows the creators of the game in an info box that pops up. The Zoom in/out is the zooming function added to the game.



Illustration 7 Menu items

## Requirement 5

### Move counter



In the game we have a move counter that shows how many moves the car have moved. To the left we see that the car has a moves counter in the upper right corner of the game. We see that the car has moved 10 steps. One of the challenges of the game is to make the path from start to finish in a few moves as possibly.

Pos: 4, 6 Speed: 0, 1  
Options Exit

Illustration 8 Move counter

## Requirement 6

### Finishing the race

When the car touch or pass the finish line the game is over, and the player gets notified with a "YEAH! YOU MADE IT". Then the keyboard is locked, and the player must choose Reset Game from the Options menu to play again. The finish line is marked as a green line at the bottom right corner of the course. Also the "Moves" status show how many steps the player used from start to the finish line.

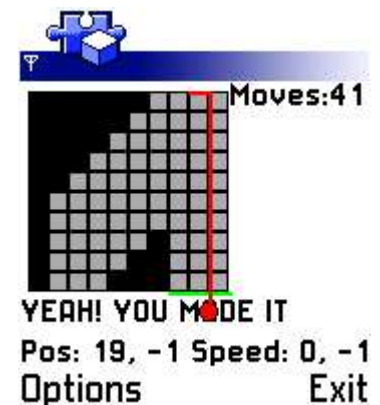


Illustration 9 Passing Finish line

## Description of design

### The game board

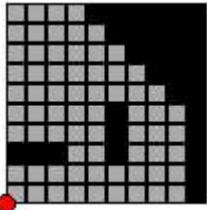


Illustration  
10Game board

The game board is made up of a two-dimensional array of integers. Where 0 represents a possible route and 1 are obstacles. When printing the game board to screen, the array is iterated and obstacles are printed as black squares and the possible route is printed as blank squares. Using an array make it very easy to later implement different levels in the game, where each level has its own array. This approach makes the solution more dynamic to make new levels.

### The oversized game board

In this final version of the Vector Rally game we have chosen to implement a game board that is four times bigger than what is shown on the screen. The game board is divided into 4 zones. It is the position vectors of the car who decides which zone that is printed to the screen.

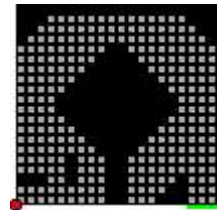


Illustration  
11Oversized game board

### The complete path

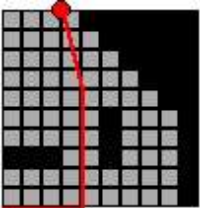


Illustration  
12Complete path

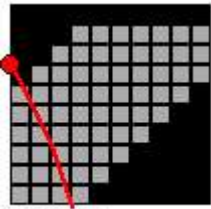
As stated in the requirements the complete path of the car is printed to the screen. Between each move the position of the car is saved in an array of TPoints. A TPoint consists of an x and y point which make up an exact position on the screen. The array is iterated after the car has reached its new position, and a line is drawn between the TPoints. The use of an oversized game board represented a challenge when drawing the complete path. We only want to see the part of the path from the zone we are in. This was solved by saving the zone the car is in when saving a TPoint. When the array of TPoints is iterated we only print to the screen those belonging to the zone the car is in.

## Car movement and collision detection

The car obtains a new position by the player pushing key 1 to 9 on the phone (or emulator). Then it is the MoveCar method in the car class that does the actual movement of the car on the screen.

### Principle of the MoveCar method

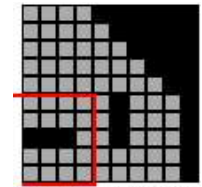
When the MoveCar method is called we have information of to points on the screen, the cars previous position (iPrevX and iPrevY) and the cars new position (iX and iY). It's vital to the accuracy of the game that the move is done step by step and not in one big jump (which would be very easy to implement). This to make sure that obstacles is not jumped over. To do this we use a do-while loop that runs until the cars previous position reaches the cars new position. After each step we check if the car has bumped into any obstacles. This is done by a call to the CheckCrash method in the obstacles class. The CheckCrash method does a lookup in the game board array to see if the car has crashed in an



**CRASH!**

*Illustration 13 Car crashed*

obstacle. If there is a crash the keypad is locked from further input, and the text “CRASH!” is shown, as the illustration to the left shows. The player must press the “Reset game” menu item to start the game over. There is also a check to see if the car has moved out of the board, it then shows “Out of Bounds!” as shown to the right. We also check to see if the car has reached the finish line. Finally we have some code that prevents the player from going back into a previous visited zone.



**Out of Bounds!**

*Illustration 14 Out of bounds*

## **Zoom effect**

The game has two zoom levels, zoom in and zoom out, which the player chooses from by pressing the respective menu items. The game starts initially at the zoom in level. In the zoom in mode,  $\frac{1}{4}$  of the total game board is shown on the screen. The player can then move the car around the board. When the car is at the edge of a zone the player doesn't see what obstacles may be in front of him. The player can then take a chance and step over the edge, or use the zoom out function. The zoom out function shows the complete game board and the cars position on the board. The player is prevented from moving the car when the game is in zoom out mode (this would be cheating). The zoom out mode is only meant to give the player an overview, so the mode needs to be changed back to zoom in mode to perform the actual step.



*Illustration 15 Game in Zoom mode*

## Vector Rally Classdiagram

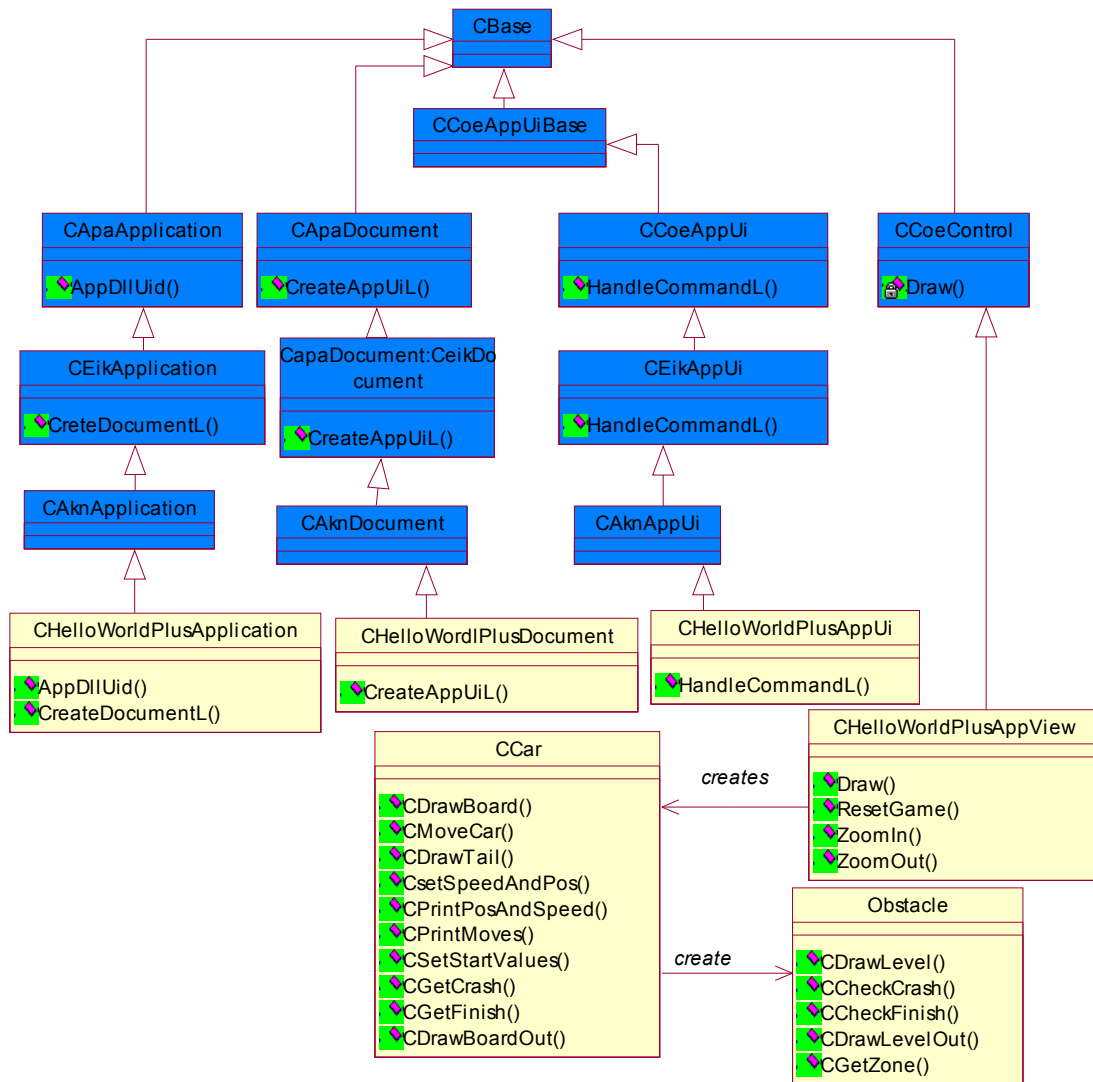


Illustration 16 Class diagram for Vector Rally

This diagram shows the classes implemented by the HelloWorldPlus example, and which files implement those classes. It also shows the class hierarchy of the classes that a standard application uses.

All the classes are derived from CBase. CBase has a number of useful features: it initialises all member data to zero, it has a virtual destructor, and it implements support for the Symbian OS cleanup stack.

I have shown the functions that we have implemented in the Car, Obstacle and ChelloWorldPlusAppView classes.

# Vector Rally Sourcecode

## CAR.H

```
#ifndef __CAR_H__
#define __CAR_H__

#include <e32base.h>
#include <e32std.h>
#include <w32std.h>
#include "obstacle.h"

class Ccar : public CBase
{

public:
    void CMoveCar(CWindowGc& aGC) const;
    void CDrawBoard(CWindowGc& aGC) const;
    void CDrawBoardOut(CWindowGc& aGC) const;
    void CsetSpeedAndPos(TInt step);
    void CPrintPosAndSpeed(CWindowGc& aGC) const;
    void CPrintMoves(CWindowGc& aGC) const;
    void CSetStartValues();
    void CDrawTail(CWindowGc& aGC) const;
    TBool CGetCrash() const;
    TBool CGetFinish()const;

    /*!
    @function NewL
    @discussion Create a car object
    @result a pointer to the created instance of car
    */

    static Ccar* NewL();

    /*!
    @function ~CCar
    @discussion Destroy the object and release all memory objects
    */
    ~Ccar();

    /*!
    @function CCar
    @discussion Constructs this object
    */

    Ccar();

private:
    Cobstacle* myObstacle;

    /*!
    @function ConstructL
    @discussion Performs second phase construction of this object
    */
    void ConstructL();

private:
    // Member variables
    mutable TInt8 iX, iY, iDX, iDY;
    mutable TInt8 iPrevY, iPrevX;
    mutable TInt iCounter;
    mutable TBool iCrash, iFinish;
    mutable TInt8 iCarZone;

```

```

mutable TPoint iTail[200];
mutable TInt iTailCounter;

mutable TInt iZoneTracker[200];
mutable TInt iTrackerCounter;

};

#endif // __CAR_H__

```

## **CAR.CPP**

```

#include "car.h"

Ccar * Ccar::NewL()
{
    Ccar * self = new (ELeave) Ccar();

    CleanupStack::PushL(self);

    self->ConstructL();

    CleanupStack::Pop(self);

    return self;
}

Ccar::Ccar()
{
    iX = iY = iDX = iDY = 0;
    iPrevX = iPrevY = 0;
    iCounter = 0;
    iCrash = iFinish = false;
    iCarZone = 1;
    iTailCounter = 0;
    iTrackerCounter = 0;
}

Ccar::~Ccar()
{
    delete myObstacle;
    myObstacle = NULL;
}

void Ccar::ConstructL()
{
    //create the obstacle object
    myObstacle = Cobstacle::NewL();
}

//when zoom in
void Ccar::CDrawBoard(CWindowGc& aGC) const {
    //draw the game level found in the obstacle class
    myObstacle->CDrawLevel(aGC, iY, iX);
}

//when zoom out
void Ccar::CDrawBoardOut(CWindowGc& aGC) const {
    //draw the game level found in the obstacle class
    myObstacle->CDrawLevelOut(aGC, iY, iX);
}

//move the car on the board step by step - draw the car and detect collisions
void Ccar::CMoveCar(CWindowGc& aGC) const {

    //max number of moves is 200
    if(iCounter >= 200) {
        aGC.DrawText(_L("Out of gas!"), TPoint(2,120));
        iFinish = true;
    }
}

```

```

aGC.SetPenColor(KRgbBlack);
aGC.SetBrushStyle(CWindowGc::ESolidBrush);

//some temp variables to help make the move
TInt moveX = iPrevX;
TInt moveY = iPrevY;
TInt flipY;

    //a loop to make the move
    do {

        //must do the move step by step to avoid jumping obstacles
        if(moveX < iX) moveX ++;
        if(moveX > iX) moveX --;
        if(moveY < iY) moveY ++;
        if(moveY > iY) moveY --;

        //must adjust y to place the car correct in the array
        flipY = (19 - moveY);//flips the y value

        //check for collisions
        TBool crash = myObstacle->CCheckCrash(flipY, moveX);
        if(crash){
            aGC.DrawText( _L("CRASH!"), TPoint(2,120));
            iCrash = true;
        }

        //check to see if the car has reached the finish line
        iCarZone = myObstacle->CGetZone();
        if(iCarZone == 4 && moveX >= 17 && moveY <= 0) {
            aGC.DrawText( _L("FINISH"), TPoint(2,120));
            iFinish = true;
        }

        //check to see if the car is outside the board
        else if(moveX < 0 || moveX > 20 || moveY < 0 || moveY > 20) {
            aGC.DrawText( _L("Out of Bounds!"), TPoint(2,120));
            iCrash = true;
        }
    }while (moveX != iX || moveY != iY);

    //need to keep track of the car's last zone
    iCarZone = myObstacle->CGetZone();
    iZoneTracker[iTrackerCounter] = iCarZone;

    //to make sure you dont go back to a prev. zone
    TInt currentZone = iZoneTracker[iTrackerCounter];
    for (int i = 0; i < iTrackerCounter; i++) {
        if(iZoneTracker[i] > currentZone) {
            iX = iPrevX;
            iY = iPrevY;
            myObstacle->CDrawLevel(aGC, iY, iX);
            aGC.DrawText( _L("MOVE FORWARD!"), TPoint(2,120));
        }
    }

    //because of the oversized board we need to adjust the car's position when going from
    //one board to another
    TInt8 carX, carY;
    if(iX > 10) carX = iX - 10;
    else carX = iX;

    if(iY > 10) carY = iY - 10;
    else carY = iY;

    //adds the TPoint to the array so the complete path can be drawn
    iTail[iTailCounter] = TPoint((carX * 10) + 5 , ((10 - carY) * 10) + 5);

    iTailCounter++;
    iTrackerCounter++;

```

```

        //draw the car's new position
        aGC.SetBrushColor(KRgbRed);
        aGC.SetPenSize(TSize(1,1));
        //TPoint is the upper left corner for iX and iY
        aGC.DrawEllipse(TRect(TPoint(carX * 10 , (10 - carY) * 10),TSize(10,10)));

} //ends the MoveCar function

//draws the tail of the car depending of which zone the car is in
void Ccar::CDrawTail(CWindowGc& aGC) const {

    aGC.SetPenColor(KRgbRed);
    aGC.SetPenSize(TSize(2,2));
    iCarZone = myObstacle->CGetZone();

    //iterate the array to draw the tail
    for (int i = 0; i < iTailCounter - 1; i++) {
        //gets the first point
        TInt tx1 = iTail[i].iX;
        TInt ty1 = iTail[i].iY;

        //gets the first zone
        TInt z1 = iZoneTracker[i];

        //gets the next point
        TInt tx2 = iTail[i+1].iX;
        TInt ty2 = iTail[i+1].iY;

        //gets the next zone
        TInt z2 = iZoneTracker[i+1];

        //make sure we only draw the tail for the current zone
        if(iCarZone == 1) {
            if(z1 == 1 && z2 == 1){
                aGC.DrawLine(TPoint(tx1, ty1), TPoint(tx2, ty2));
            }
        }
        if(iCarZone == 2) {
            if(z1 == 2 && z2 == 2){
                aGC.DrawLine(TPoint(tx1, ty1), TPoint(tx2, ty2));
            }
            else if(z2 == 2) {
                aGC.DrawLine(TPoint(tx1, 105), TPoint(tx2, ty2));
            }
        }
        if(iCarZone == 3) {
            if(z1 == 3 && z2 == 3){
                aGC.DrawLine(TPoint(tx1, ty1), TPoint(tx2, ty2));
            }
            else if(z2 == 3) {
                aGC.DrawLine(TPoint(5, ty1), TPoint(tx2, ty2));
            }
        }
        if(iCarZone == 4) {
            if(z1 == 4 && z2 == 4){
                aGC.DrawLine(TPoint(tx1, ty1), TPoint(tx2, ty2));
            }
            else if(z2 == 4) {
                aGC.DrawLine(TPoint(tx1, 5), TPoint(tx2, ty2));
            }
        }
    }

    //ends the loop
} //ends the DrawTail function

//sets new speed and position vectors

```

```

void Ccar::CsetSpeedAndPos(TInt step) {

    iPrevX = iX;//keep track of previous x and y
    iPrevY = iY;//so we can do the move step by step

    iCounter ++;

    //speed vectors
    switch(step) {
    case 1:
        iDX += -1; iDY += 1; break;
    case 2:
        iDX += 0; iDY += 1; break;
    case 3:
        iDX += 1; iDY += 1; break;
    case 4:
        iDX += -1; iDY += 0; break;
    case 5:
        iDX += 0; iDY += 0; break;
    case 6:
        iDX += 1; iDY += 0; break;
    case 7:
        iDX += -1; iDY += -1; break;
    case 8:
        iDX += 0; iDY += -1; break;
    case 9:
        iDX += 1; iDY += -1;break;
    default:
        break;
    }

    //In each iteration the car obtains a new position by adding its speed vector
    //to its position vector with the option of adjusting by +/- 1 in each dimension.

    iX += (iDX); iY += (iDY); //sets the new iX and iY values
}

//prints position and speed
void Ccar::CPrintPosAndSpeed(CWindowGc& aGC) const {
    TBuf16<50> posAndSpeedBuffer;
    _LIT (posAndSpeedText, "Pos: %d, %d Speed: %d, %d");
    posAndSpeedBuffer.Format(posAndSpeedText, iX, iY, iDX, iDY);
    aGC.SetPenColor(KRgbBlack);
    aGC.DrawText(posAndSpeedBuffer, TPoint(2,140));
}

//prints the numbers of moves made
void Ccar::CPrintMoves(CWindowGc& aGC) const {
    TBuf16<50> counterBuffer;
    _LIT (counterText, "Moves:%d");
    counterBuffer.Format(counterText, iCounter);
    aGC.SetPenColor(KRgbBlack);
    aGC.DrawText(counterBuffer, TPoint(106,12));
}

//resets the game
void Ccar::CSetStartValues() {
    iX = iY = iDX = iDY = 0;
    iPrevX = iPrevY = 0;
    iCounter = 0;
    iCrash = iFinish = false;
    iTailCounter = 0;
    iTrackerCounter = 0;
}

TBool Ccar::CGetCrash()const{
return iCrash;
}

TBool Ccar::CGetFinish()const{
return iFinish;
}

```

## **OBSTACLE.H**

```
/* Copyright (c) 2004, Nokia Mobile Phones. All rights reserved */

#ifndef __OBSTACLE_H__
#define __OBSTACLE_H__

#include <e32base.h>
#include <e32std.h>
#include <w32std.h>

class Cobstacle : public CBase
{
public:

void CDrawLevel(CWindowGc& aGC, TInt8 aY, TInt8 aX) const;
void CDrawLevelOut(CWindowGc& aGC, TInt8 aY, TInt8 aX) const;

TBool CCheckCrash(TInt8 aFlipY, TInt8 aMoveX) const;
TInt8 CGetZone() const;

/*!
 @function NewL
 @discussion Create a obstacle object
 @result a pointer to the created instance of obstacle
 */
 static Cobstacle* NewL();

/*!
 @function ~obstacle
 @discussion Destroy the object and release all memory objects
 */
 ~Cobstacle();

private:

/*!
 @function obstacle
 @discussion Constructs this object
 */
 Cobstacle();

/*!
 @function ConstructL
 @discussion Performs second phase construction of this object
 */
 void ConstructL();

private:
 // Member variables
 mutable TInt8 iZone;

};

#endif // __OBSTACLE_H__
```

## **OBSTACLE.CPP**

```
#include "obstacle.h"

Cobstacle * Cobstacle::NewL()
{
 Cobstacle * self = new (ELeave) Cobstacle();

 CleanupStack::PushL(self);

 self->ConstructL();

 CleanupStack::Pop(self);
}
```

```

        return self;
    }

Cobstacle::Cobstacle()
{
    iZone = 1;
}

Cobstacle::~Cobstacle()
{
}

void Cobstacle::ConstructL()
{
}

//the array that represents the game board
TInt const level2[21][21] =
{{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,9},
{1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,9},
{1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,9},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,9},
{0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,9},
{0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,9},
{0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,9},
{0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,9},
{0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,9},
{0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,9},
{0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,9},
{0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,9},
{0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,9},
{0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,9},
{0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,9},
{0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,9},
{0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,9},
{0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,9},
{0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,9},
{0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,9},
{0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,9},
{0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,9},
{9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9}};

//draw the gameboard
void Cobstacle::CDrawLevel(CWindowGc& aGC, TInt8 aY, TInt8 aX) const {

    aGC.SetPenColor(KRgbBlack);
    aGC.SetPenSize(TSize(1,1));
    aGC.SetBrushStyle(CWindowGc::ESolidBrush);

    //variables to help iterate the array
    TInt8 y;
    TInt8 x;
    TInt8 drawY = -1;
    TInt8 drawX;
    TInt8 limitX;
    TInt8 limitY;

    //The x and y coordinates decides which part of the array we print

    //upper left part
    if(aY > 10 && aX <11) {
        y=0;
        x=0;
        limitX = 10;
        limitY = 10;
        iZone = 2;
    }
    //upper right part
    else if(aY > 10 && aX > 10) {
        y=0;

```

```

        x=10;
        limitX = 20;
        limitY = 10;
        iZone = 3;
    }
    //lower right
    else if(aY < 11 && aX > 10) {
        y=10;
        x=10;
        limitX = 20;
        limitY = 20;
        iZone = 4;
    }

    //lower left (startposition)
    else {
        y=10;
        x=0;
        limitX = 10;
        limitY = 20;
        iZone = 1;
    }

    //iterate the selected part of the two-dimensional array
    for( ; y<limitY; y++) { //y-aksen
        drawY++;
        drawX = -1;

        for( ; x<limitX; x++) { //x-aksen
            drawX++;
            if(level2[y][x] == 1) aGC.SetBrushColor(KRgbBlack);
            else aGC.SetBrushColor(KRgbGray);
            //draws the actual part of the array to the screen
            aGC.DrawRect(TRect((drawX * 10) + 5, (drawY * 10) + 5, (drawX * 10) + 15,
(drawY * 10) + 15));
        }

        x -= 10; //resets the x value
    }

    //draw the finish line if the car is in zone 4
    if(iZone == 4) {
        aGC.SetPenColor(KRgbGreen);
        aGC.SetPenSize(TSize(2,2));
        aGC.DrawLine(TPoint(75, 105), TPoint(105, 105));
    }
}

//draw the board when zooming out
void CObstacle::CDrawLevelOut(CWindowGc& aGC, TInt8 aY, TInt8 aX) const {
    aGC.SetPenColor(KRgbBlack);
    aGC.SetPenSize(TSize(1,1));
    aGC.SetBrushStyle(CWindowGc::ESolidBrush);

    TInt8 t, i;
    //draw the complete gameboard to the screen

    for(t=0; t<20; t++) { //y-aksen
        for(i=0; i<20; i++) { //x-aksen
            if(level2[t][i] == 1) aGC.SetBrushColor(KRgbBlack);
            else aGC.SetBrushColor(KRgbGray);
            //draws the actual part of the array to the screen
            aGC.DrawRect(TRect((i * 5) + 5, (t * 5) + 5, (i * 5) + 10, (t * 5) + 10));
        }
    }

    //draw the car
    aGC.SetBrushColor(KRgbRed);

```

```

aGC.SetPenSize(TSize(1,1));
//TPoint is the upper left corner for iX and iY
aGC.DrawEllipse(TRect(TPoint(aX * 5 + 2 , (20 - aY) * 5 + 2),TSize(6,6)));
//aGC.DrawEllipse(TRect(TPoint((aX * 5) + 2 , ((10 - aY) * 10) + 2,TSize(6,6)));

//the green finish line
aGC.SetPenColor(KRgbGreen);
aGC.SetPenSize(TSize(2,2));
aGC.DrawLine(TPoint(90, 105), TPoint(105, 105));
}

```

```

//checks if the car crashes with a obstacle
TBool Cobstacle::CCheckCrash(TInt8 aFlipY, TInt8 aMoveX) const{
    TBool crash;
    if(level2[aFlipY][aMoveX] == 1) crash = true; //right crash
    else if(level2[aFlipY][aMoveX - 1] == 1) crash = true; //left crash
    else if(level2[aFlipY + 1][aMoveX] == 1) crash = true; //right down crash
    else if(level2[aFlipY + 1][aMoveX - 1] == 1) crash = true; //left down crash
    else crash = false;

    return crash;
}

//gets the zone the car is in
TInt8 Cobstacle::CGetZone() const{
    return iZone;
}

```

## **HELLOWORLDPLUSAPPLICATION.H**

```

#ifndef __HELLOWORLDPLUS_APPLICATION_H__
#define __HELLOWORLDPLUS_APPLICATION_H__

#include <aknapp.h>

/*!
 @class CHelloWorldPlusApplication

 @discussion An instance of CHelloWorldPlusApplication is the application part of the AVKON
 application framework for the HelloWorldPlus example application
 */
class CHelloWorldPlusApplication : public CAknApplication
{
public: // from CApaApplication

/*!
 @function AppDllUid

 @discussion Return the application DLL UID value
 @result the UID of this Application/Dll
 */
    TUid AppDllUid() const;

protected: // from CEikApplication
/*!
 @function CreateDocumentL

 @discussion Create a CApaDocument object and return a pointer to it
 @result a pointer to the created document
 */
    CApaDocument* CreateDocumentL();
};

#endif // __HELLOWORLDPLUS_APPLICATION_H__

```

## HELLOWORLDPLUSAPPLICATION.CPP

```
#include "HelloWorldPlusDocument.h"
#include "HelloWorldPlusApplication.h"

// UID for the application; this should correspond to the uid defined in the mmp file
const TUid KUidHelloWorldPlusApp = {0x10005bcc};

CApaDocument* CHelloWorldPlusApplication::CreateDocumentL()
{
    // Create an HelloWorldPlus document, and return a pointer to it
    return (static_cast<CApaDocument*>(CHelloWorldPlusDocument::NewL(*this)));
}

TUid CHelloWorldPlusApplication::AppDllUid() const
{
    // Return the UID for the HelloWorldPlus application
    return KUidHelloWorldPlusApp;
}
```

## HELLOWORLDPLUSAPPUI.H

```
#ifndef __HELLOWORLDPLUS_APPUI_H__
#define __HELLOWORLDPLUS_APPUI_H__

#include <aknappui.h>

class CHelloWorldPlusAppView;

/*!
 * @class CHelloWorldPlusAppUi
 *
 * @discussion An instance of class CHelloWorldPlusAppUi is the UserInterface part of the AVKON
 * application framework for the HelloWorldPlus example application
 */
class CHelloWorldPlusAppUi : public CAknAppUi
{
public:
    /*!
     * @function ConstructL
     *
     * @discussion Perform the second phase construction of a CHelloWorldPlusAppUi object
     * this needs to be public due to the way the framework constructs the AppUi
     */
    void ConstructL();

    /*!
     * @function CHelloWorldPlusAppUi
     *
     * @discussion Perform the first phase of two phase construction.
     * This needs to be public due to the way the framework constructs the AppUi
     */
    CHelloWorldPlusAppUi();

    /*!
     * @function ~CHelloWorldPlusAppUi
     *
     * @discussion Destroy the object
     */
    ~CHelloWorldPlusAppUi();

public: // from CAknAppUi
    /*!
     * @function HandleCommandL
     *
     * @discussion Handle user menu selections
     * @param aCommand the enumerated code for the option selected
     */
    void HandleCommandL(TInt aCommand);

private:
    /*! @var iAppView The application view */

```

```

    CHelloWorldPlusAppView* iAppView;
};

#endif // __HELLOWORLDPLUS_APPUI_H__

```

## **HELLOWORLDPLUSAPPUI.CPP**

```

#include <aknnotewrappers.h>
#include <avkon.hrh>

#include <stringloader.h>
#include <helloworldplus.rsg>

#include "HelloWorldPlus.pan"
#include "HelloWorldPlusAppUi.h"
#include "HelloWorldPlusAppView.h"
#include "HelloWorldPlus.hrh"

_LIT(KHWPlusPanic, "HelloWorldPlus");

void CHelloWorldPlusAppUi::ConstructL()
{
    BaseConstructL(EAknEnableSkin);

    iAppView = CHelloWorldPlusAppView::NewL(ClientRect());

    AddToStackL(iAppView);
}

CHelloWorldPlusAppUi::CHelloWorldPlusAppUi()
{
    // No implementation required
}

CHelloWorldPlusAppUi::~CHelloWorldPlusAppUi()
{
    if (iAppView)
    {
        iEikonEnv->RemoveFromStack(iAppView);
        delete iAppView;
        iAppView = NULL;
    }
}

void CHelloWorldPlusAppUi::HandleCommandL(TInt aCommand)
{
    switch(aCommand)
    {
        case EHelloWorldPlusCommand:
        {
            // Load a string from the resource file and display it
            HBufC* textResource = StringLoader::LoadLC(R_NEW_STRING_RESOURCE);
            CAknInformationNote* informationNote;
            informationNote = new (ELeave) CAknInformationNote;
            informationNote->ExecuteLD(textResource->Des());
            CleanupStack::PopAndDestroy(textResource);
        }
        break;

        case EHelloWorldPlusDrawNow:
        {
            _LIT(message, "Zooming in");
            CAknInformationNote* informationNote = new (ELeave) CAknInformationNote;
            informationNote->ExecuteLD(message);
            iAppView->ZoomIn();
        }
        break;

        case EHelloWorldPlusUserDraw:
        {
            _LIT(message, "Zooming out");
            CAknInformationNote* informationNote = new (ELeave) CAknInformationNote;

```

```

        informationNote->ExecuteLD(message);

        iAppView->ZoomOut();

        }
        break;
    case EHelloWorldPlusCommand1:
    {
        iAppView->ResetGame();
    }
    break;

    case EEikCmdExit:
    case EAknSoftkeyExit:
        Exit();
        break;

    default:
        User::Panic(KHWPlusPanic, EHelloWorldPlusBasicUi);
        break;
}
}

```

## **HELLOWORLDPLUSAPPVIEW.H**

```

#ifndef __HELLOWORLDPLUSAPPVIEW_H__
#define __HELLOWORLDPLUSAPPVIEW_H__

#include <coecntrl.h>
// #include "obstacle.h"
#include "car.h"

/*!
 * @class CHelloWorldPlusAppView
 *
 * @discussion An instance of the Application View object for the HelloWorldPlus
 * example application
 */
class CHelloWorldPlusAppView : public CCoeControl
{
public:
    /*!
     * @function NewL
     *
     * @discussion Create a CHelloWorldPlusAppView object, which will draw itself to aRect
     * @param aRect the rectangle this view will be drawn to
     * @result a pointer to the created instance of CHelloWorldPlusAppView
     */
    static CHelloWorldPlusAppView* NewL(const TRect& aRect);

    /*!
     * @function NewLC
     *
     * @discussion Create a CHelloWorldPlusAppView object, which will draw itself to aRect
     * @param aRect the rectangle this view will be drawn to
     * @result a pointer to the created instance of CHelloWorldPlusAppView
     */
    static CHelloWorldPlusAppView* NewLC(const TRect& aRect);

    /*!
     * @function ~CHelloWorldPlusAppView
     *
     * @discussion Destroy the object
     */
    ~CHelloWorldPlusAppView();

    /*!
     * @function UserDraw
     */

```

```

    @discussion Draw this CHelloWorldAppView to the screen
    */
    void UserDraw() const;

/*function for reset the game*/
void ResetGame();

/*zoom functions*/
void ZoomOut();
void ZoomIn();

public: // from CCoeControl
/*!
@function Draw

@discussion Draw this CHelloWorldPlusAppView to the screen
@param aRect the rectangle of this view that needs updating
*/
void Draw(const TRect& aRect) const;

/*!
@function OfferKeyEventL

@discussion Handle any user keypresses
@param aKeyEvent holds the data for the event that occurred
@param aType holds the type of key event that occurred
@result a TKeyResponse indicating if the key was consumed or not
*/
TKeyResponse OfferKeyEventL(const TKeyEvent& aKeyEvent, TEventCode aType);

/*!
@function InputCapabilities

@discussion Return the capabilities of the OfferKeyEventL
method for this class
@result a TCoeInputCapabilities indicating the capabilities
for this class
*/
TCoeInputCapabilities InputCapabilities() const;

private:

//Create pointers to the objects
Ccar* myCar;
mutable TBool iZoom;

/*!
@function ConstructL

@discussion Perform the second phase construction of a CHelloWorldPlusAppView object
@param aRect the rectangle this view will be drawn to
*/
void ConstructL(const TRect& aRect);

/*!
@function CHelloWorldPlusAppView

@discussion Perform the first phase of two phase construction
*/
CHelloWorldPlusAppView();
};

#endif // __HELLOWORLDPLUSAPPVIEW_H__

```

## HELLOWORLDPLUSAPPVIEW.CPP

```
#include <coemain.h>
#include <aknotewrappers.h>
#include <HelloWorldPlus.rsg>

#include "HelloWorldPlusAppView.h"

CHelloWorldPlusAppView* CHelloWorldPlusAppView::NewL(const TRect& aRect)
{
    CHelloWorldPlusAppView* self = CHelloWorldPlusAppView::NewLC(aRect);
    CleanupStack::Pop(self);
    return self;
}

CHelloWorldPlusAppView* CHelloWorldPlusAppView::NewLC(const TRect& aRect)
{
    CHelloWorldPlusAppView* self = new (ELeave) CHelloWorldPlusAppView;
    CleanupStack::PushL(self);
    self->ConstructL(aRect);
    return self;
}

void CHelloWorldPlusAppView::ConstructL(const TRect& aRect)
{
    // Create a window for this application view
    CreateWindowL();

    // Set the windows size
    SetRect(aRect);

    // Activate the window, which makes it ready to be drawn
    ActivateL();

    //create the car object
    myCar = Ccar::NewL();

}

CHelloWorldPlusAppView::CHelloWorldPlusAppView()
{
    iZoom = true;//the default is zooming in
    // No implementation required
}

CHelloWorldPlusAppView::~CHelloWorldPlusAppView()
{
    delete myCar;
    myCar = NULL;
}

void CHelloWorldPlusAppView::Draw(const TRect& /*aRect*/) const
{
    // Get the standard graphics context
    CWindowGc& gc = SystemGc();

    // Gets the control's extent
    TRect rect = Rect();

    // Clears the screen
    gc.Clear(rect);

    //sets the font
    gc.UseFont(iCoeEnv->NormalFont());

    if(iZoom) {

        //draw the game board
        myCar->CDrawBoard(gc);

        //move the car
        myCar->CMoveCar(gc);
    }
}
```

```

        //draw the tail
        myCar->CDrawTail(gc);
    }

    else {
        myCar->CDrawBoardOut(gc);
    }

    //print pos and speed
    myCar->CPrintPosAndSpeed(gc);

    //prints number of moves made
    myCar->CPrintMoves(gc);
}

TKeyResponse CHelloWorldPlusAppView::OfferKeyEventL(
    const TKeyEvent& aKeyEvent, TEventCode aType)
{
    if(!myCar->CGetCrash() && !myCar->CGetFinish() && iZoom) {
        // We only want the key press, not the key up/down event
        if (aType == EEventKey)
        {
            // Check if the 2 key was pressed
            if (aKeyEvent.iCode == '1')
            {
                //sets the car's new position and speed vector
                myCar->CsetSpeedAndPos(1);
                UserDraw();
                return EKeyWasConsumed;
            }
            if (aKeyEvent.iCode == '2')
            {
                myCar->CsetSpeedAndPos(2);
                UserDraw();
                return EKeyWasConsumed;
            }
            if (aKeyEvent.iCode == '3')
            {
                myCar->CsetSpeedAndPos(3);
                UserDraw();
                return EKeyWasConsumed;
            }
            if (aKeyEvent.iCode == '4')
            {
                myCar->CsetSpeedAndPos(4);
                UserDraw();
                return EKeyWasConsumed;
            }
            if (aKeyEvent.iCode == '5')
            {
                myCar->CsetSpeedAndPos(5);
                UserDraw();
                return EKeyWasConsumed;
            }
            if (aKeyEvent.iCode == '6')
            {
                myCar->CsetSpeedAndPos(6);
                UserDraw();
                return EKeyWasConsumed;
            }
            if (aKeyEvent.iCode == '7')
            {
                myCar->CsetSpeedAndPos(7);
                UserDraw();
                return EKeyWasConsumed;
            }
            if (aKeyEvent.iCode == '8')
            {
                myCar->CsetSpeedAndPos(8);
                UserDraw();
            }
        }
    }
}

```

```

        return EKeyWasConsumed;
    }
    if (aKeyEvent.iCode == '9')
    {
        myCar->CsetSpeedAndPos(9);
        UserDraw();
        return EKeyWasConsumed;
    }
}

} //end if
// Return the default functionality
return CCoeControl::OfferKeyEventL(aKeyEvent, aType);
}

TCoeInputCapabilities CHelloWorldPlusAppView::InputCapabilities() const
{
    // This class does not implement any 'unusual' input capabilities
    return TCoeInputCapabilities::ENone;
}

void CHelloWorldPlusAppView::UserDraw() const
{
    ActivateGc();
    Draw(Rect());
    DeactivateGc();
}

//resets the game if the user presses the menu item "Reset"
void CHelloWorldPlusAppView::ResetGame()
{
    myCar->CSetStartValues();
    iZoom = true;
    UserDraw();
}

void CHelloWorldPlusAppView::ZoomOut()
{
    iZoom = false;
    UserDraw();
}

void CHelloWorldPlusAppView::ZoomIn()
{
    iZoom = true;
    UserDraw();
}

```

## **HELLOWORLDPLUSDOCUMENT.H**

```

#ifndef __HELLOWORLDPLUS_DOCUMENT_H__
#define __HELLOWORLDPLUS_DOCUMENT_H__

#include <akndoc.h>

// Forward references
class CHelloWorldPlusAppUi;
class CEikApplication;

/*!
    @class CHelloWorldPlusDocument

    @discussion An instance of class CHelloWorldPlusDocument is the Document part of the AVKON
    application framework for the HelloWorldPlus example application
    */

```

```

class CHelloWorldPlusDocument : public CAknDocument
{
public:
    /*!
    @function NewL

    @discussion Construct a CHelloWorldPlusDocument for the AVKON application aApp
    using two phase construction, and return a pointer to the created object
    @param aApp application creating this document
    @result a pointer to the created instance of CHelloWorldPlusDocument
    */
    static CHelloWorldPlusDocument* NewL(CEikApplication& aApp);

    /*!
    @function NewLC

    @discussion Construct a CHelloWorldPlusDocument for the AVKON application aApp
    using two phase construction, and return a pointer to the created object
    @param aApp application creating this document
    @result a pointer to the created instance of CHelloWorldPlusDocument
    */
    static CHelloWorldPlusDocument* NewLC(CEikApplication& aApp);

    /*!
    @function ~CHelloWorldPlusDocument

    @discussion Destroy the object
    */
    ~CHelloWorldPlusDocument();

    /*!
    @function CreateAppUiL

    @discussion Create a CHelloWorldPlusAppUi object and return a pointer to it
    @result a pointer to the created instance of the AppUi created
    */
    CEikAppUi* CreateAppUiL();
private:
    /*!
    @function ConstructL

    @discussion Perform the second phase construction of a CHelloWorldPlusDocument object
    */
    void ConstructL();

    /*!
    @function CHelloWorldPlusDocument

    @discussion Perform the first phase of two phase construction
    @param aApp application creating this document
    */
    CHelloWorldPlusDocument(CEikApplication& aApp);

};

#endif // __HELLOWORLDPLUS_DOCUMENT_H__

```

## **HELLOWORLDPLUSDOCUMENT.CPP**

```

#include "HelloWorldPlusAppUi.h"
#include "HelloWorldPlusDocument.h"

CHelloWorldPlusDocument* CHelloWorldPlusDocument::NewL(CEikApplication& aApp)
{
    CHelloWorldPlusDocument* self = NewLC(aApp);
    CleanupStack::Pop(self);
    return self;
}

CHelloWorldPlusDocument* CHelloWorldPlusDocument::NewLC(CEikApplication& aApp)

```

```

    {
        CHelloWorldPlusDocument* self = new (ELeave) CHelloWorldPlusDocument(aApp);
        CleanupStack::PushL(self);
        self->ConstructL();
        return self;
    }

void CHelloWorldPlusDocument::ConstructL()
{
    // No implementation required
}

CHelloWorldPlusDocument::CHelloWorldPlusDocument(CEikApplication& aApp) : CAknDocument(aApp)
{
    // No implementation required
}

CHelloWorldPlusDocument::~CHelloWorldPlusDocument()
{
    // No implementation required
}

CEikAppUi* CHelloWorldPlusDocument::CreateAppUiL()
{
    // Create the application user interface, and return a pointer to it
    return(static_cast<CEikAppUi*>(new (ELeave) CHelloWorldPlusAppUi));
}

```

## **HELLOWORLDPLUS.CPP**

```

#include "HelloWorldPlusApplication.h"

// DLL entry point, return that everything is ok
GLDEF_C TInt E32Dll(TDllReason /*aReason*/)
{
    return KErrNone;
}

// Create an application, and return a pointer to it
EXPORT_C CApaApplication* NewApplication()
{
    return (static_cast<CApaApplication*>(new CHelloWorldPlusApplication));
}

```

## **HELLOWORLDPLUS.PREF**

```

TARGET        HELLOWORLDPLUS.APP
TARGETTYPE    APP
UID           0x100039ce 0x10005bcc

```

## **HELLOWORLDPLUS.RESOURCES**

```

<resources>
  <rscc sourcefile = "GROUP\HELLOWORLDPLUS.RSS" targetpath = "Z\SYSTEM\APPS\HELLOWORLDPLUS" header =
"true" sourcepath = ".">
    <language id = "SC"/>
  </rscc>
  <rscc sourcefile = "GROUP\HELLOWORLDPLUS_CAPTION.RSS" targetpath = "Z\SYSTEM\APPS\HELLOWORLDPLUS"
header = "true" sourcepath = ".">
    <language id = "SC"/>
  </rscc>
</resources>

```

## HELLOWORLDPLUS.MMP

```
TARGET          HelloWorldPlus.app
TARGETTYPE      app

// Change the second number here to change the UID for this application
UID             0x100039CE 0x10005bcc
TARGETPATH      \system\apps\helloworldplus

SOURCEPATH      ..\src
SOURCE          HelloWorldPlus.cpp
SOURCE          HelloWorldPlusApplication.cpp
SOURCE          HelloWorldPlusAppView.cpp
SOURCE          HelloWorldPlusAppUi.cpp
SOURCE          HelloWorldPlusDocument.cpp
SOURCE          obstacle.cpp
SOURCE          car.cpp

SOURCEPATH      ..\group
RESOURCE        HelloWorldPlus.rss
RESOURCE        HelloWorldPlus_caption.rss

USERINCLUDE     ..\inc

SYSTEMINCLUDE   \epoc32\include

LIBRARY         euser.lib
LIBRARY         apparc.lib
LIBRARY         cone.lib
LIBRARY         eikcore.lib
LIBRARY         avkon.lib
LIBRARY         commonengine.lib
```

## HELLOWORLDPLUS.RSS

```
NAME HELL

#include <eikon.rh>
#include <avkon.rh>
#include <avkon.rsg>

#include "HelloWorldPlus.hrh"

// -----
//
//   Define the resource file signature
//   This resource should be empty.
//
// -----
//
RESOURCE RSS_SIGNATURE
{
}

// -----
//
//   Default Document Name
//
// -----
//
RESOURCE TBUF r_default_document_name
{
    buf="";
}

// -----
//
//   Define default menu and CBA key.
//
// -----
//
RESOURCE EIK_APP_INFO
{
```

```

menubar = r_helloworldplus_menubar;
cba = R_AVKON_SOFTKEYS_OPTIONS_EXIT;
}

// -----
//
// r_helloworldplus_menubar
// Menubar for HelloWorldPlus example
// -----
//
RESOURCE MENU_BAR r_helloworldplus_menubar
{
    titles =
    {
        MENU_TITLE
            {
                menu_pane = r_helloworldplus_menu;
            }
    };
}

// -----
//
// r_helloworldplus_menu
// Menu for "Options"
// -----
//
RESOURCE MENU_PANE r_helloworldplus_menu
{
    items =
    {
        // added the new Options menu command here
        MENU_ITEM
            {
                command = EHelloWorldPlusCommand;
                txt = "Credits";
            },
        MENU_ITEM
            {
                command = EHelloWorldPlusDrawNow;
                txt = "Zoom in";
            },
        MENU_ITEM
            {
                command = EHelloWorldPlusUserDraw;
                txt = "Zoom out";
            },
        MENU_ITEM
            {
                command = EHelloWorldPlusCommand1;
                txt = "Reset game";
            },
        MENU_ITEM
            {
                command = EAknSoftkeyExit;
                txt = "Exit";
            }
    };
}

// -----
//
// New string resource
// -----
//
RESOURCE TBUF32 r_new_string_resource
{
    buf="Developed by Lars and Tor";
}

```

## **HELLOWORLDPLUS\_CAPTION.RSS**

```
#include <apcaptionfile.rh>

RESOURCE CAPTION_DATA
{
    caption="VectorRally";
    shortcaption="VR4";
}
```

## **HELLOWORLDPLUS.PKG**

```
; HelloWorldPlus.pkg
;
;Language - standard language definitions
&EN

; standard SIS file header
#{ "HelloWorldPlus", (0x10000001), 1, 0, 0

; Supports Series 60 v2.0
(0x101F7960), 0, 0, 0, { "Series60ProductID" }

;
"C:\Symbian\7.0s\Series60_v21_CW\Epoc32\release\armi\urel\HelloWorldPlus.APP"-":\system\apps\HelloWorldPlus\HelloWorldPlus.app"
"C:\Symbian\7.0s\Series60_v21_CW\Epoc32\data\z\system\apps\HELLOWORLDPLUS\HelloWorldPlus.rsc"-":\system\apps\HelloWorldPlus\HelloWorldPlus.rsc"
"C:\Symbian\7.0s\Series60_v21_CW\Epoc32\data\z\system\apps\HELLOWORLDPLUS\HelloWorldPlus_caption.rsc"-":\system\apps\HelloWorldPlus\HelloWorldPlus_caption.rsc"
```