Dumbo Driver

Team “Dumbo”

<https://dumbo.codeplex.com/>

AUTHORS:

Dimitrina Zlatkova

Veronika Georgieva

Kristyan Boikov

Penka Kamenova

Martin Veshev

Version 1.0

20.03.2013

Table of Contents

[Dumbo Driver 1](#_Toc351471005)

[Purpose 3](#_Toc351471016)

[Technologies 3](#_Toc351471017)

[User Interface 3](#_Toc351471018)

[Functionality 3](#_Toc351471019)

[Variables 3](#_Toc351471020)

[Objects 3](#_Toc351471021)

[User Input 4](#_Toc351471022)

[Game Over 4](#_Toc351471023)

[Structure 4](#_Toc351471024)

[GameObject Class 4](#_Toc351471025)

[GameInfo Class 5](#_Toc351471026)

[Field Class 5](#_Toc351471027)

[Lane enumeration 5](#_Toc351471028)

[InvalidSpeedOperationException Class 5](#_Toc351471029)

[ConsoleRenderer Class 5](#_Toc351471030)

[KeyboardInterface Class 6](#_Toc351471031)

[RandomObjectGenerator Class 6](#_Toc351471032)

[CollisionDetector Class 6](#_Toc351471033)

[Timer Class 6](#_Toc351471034)

[Engine Class 7](#_Toc351471035)

[Methods 7](#_Toc351471036)

[Run method 7](#_Toc351471037)

[DumboDriverMain Class 7](#_Toc351471038)

[Singleton Design Pattern 7](#_Toc351471039)

# Purpose

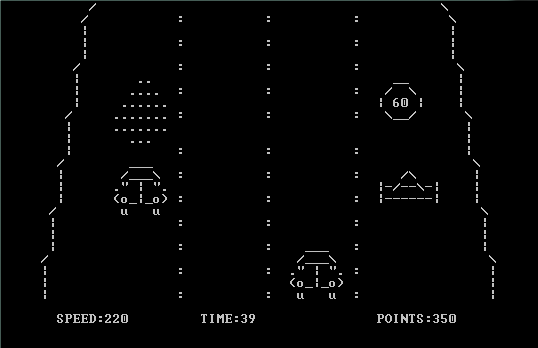
Since it is a game, the main purpose of the project is to entertain the users. It is a racing game, which means it can be played many times until the user achieves desired high score.

# Technologies

The project was made using these technologies:

* Visual Studio 2012
* .NET Framework 4.5
* C# Console Application

# User Interface



# Functionality

## Variables

* Speed
  + initial is 50
  + can vary in the range [0 - 300]
* Time
  + initial is 30
  + decreases every 1 second
  + can vary in the range [0 - infinity]
* Points
  + initial is 0
  + can vary in the range [0 - infinity]

## Objects

* Dumbo Car
* horizontal position is controlled by user input;
* steady vertical position;
* collides with other objects.
* Other Car
  + same design as Dumbo Car;
  + steady horizontal position;
  + vertical position controlled by game engine;
  + responds to collision with Dumbo Car by reducing Speed with 140.
* Cone
  + steady horizontal position;
  + vertical position controlled by game engine;
  + responds to collision with Dumbo Car by reducing Speed with 80.
* Spill
  + steady horizontal position;
  + vertical position controlled by game engine;
  + responds to collision with Dumbo Car by reducing Speed with 100.
* Reward
  + steady horizontal position;
  + vertical position controlled by game engine;
  + responds to collision with Dumbo Car by increasing Points with a randomized value in the range [10 - 90].
* Time Bonus
  + steady horizontal position;
  + vertical position controlled by game engine;
  + responds to collision with Dumbo Car by increasing Time with 10.

## User Input

The user interacts with the application by pressing Left and Right keyboard keys.

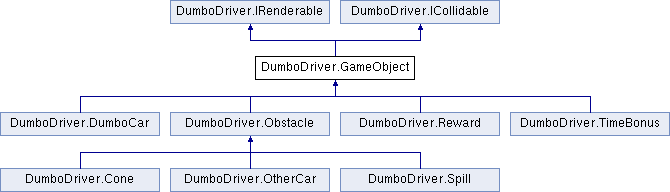
## Game Over

The game is over when the Time value is 1.

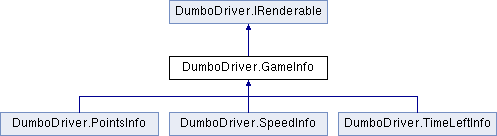
# Structure

The project was written according to the principles of Object-Oriented Programing. The main elements in the structure of the application are as follows:

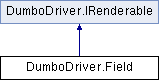
## GameObject Class



## GameInfo Class



## Field Class



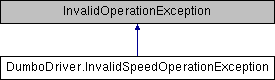
The field includes everything that is not a Game Object or Game Info.

## Lane enumeration

Determines which lane the object is in. Takes the following values:

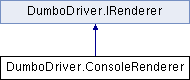
* Left
* MiddleLeft
* MiddleRight
* Right

## InvalidSpeedOperationException Class



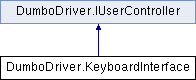
This exception is thrown when the program attempts to subtract a bigger value from Speed than it currently holds. When caught, the Speed value is set to 0.

## ConsoleRenderer Class



Its responsibilities are to render the field, the objects and the information about the game on the console.

## KeyboardInterface Class



The job of this class is to observe the events of user input and to handle them appropriately.

## RandomObjectGenerator Class

When the method GenerateObjects of this class is called, it does the following things in the following order:

* randomize number of objects produced within the range [1 - 3];
* randomize unoccupied\* lanes;
* randomize type of object - Time Bonus or other;

If the time left, which is displayed by the TimeInfo class, is less than 25, a chance is randomized within the range [1 - 100]. The chance is checked by the formula

chance < 20 – Points/100. If it returns true, a Time Bonus is generated, otherwise the generated object is something else.

* randomize other type of object – Spill, Other Car, Cone and Reward have equal chance of being generated.

## CollisionDetector Class

This class detects two types of collisions and has two methods respectively:

* HandleCollisions – handles collisions between Dumbo Car and another object by calling the RespondToCollision method of the object;
* HandleScreenExit – handles collisions between a moving object and the last row of the field by calling the ExitScreen method of the object.

## Timer Class

This is a static class, which holds variables, reached from the whole application.

* StartTime and EndTime – DateTime variables, which are responsible for the counting of the seconds passed;
* timeToReachCheckpoint – increases when Dumbo Car collides with Time Bonus, initial value is 30;
* timeLeft – initially equals timeToReachCheckpoint, later is calculated with the formula timeToReachCheckpoint - (EndTime - StartTime).Minutes\*60 - (EndTime - StartTime).Seconds;
* intervalCount – initially 0, increased by 1 every time the application reloads

## Engine Class

### Methods

* MovePlayerCarLeft()
* MovePlayerCarRight()
* Run()

### Run method

The Run method is vital and does the following things in the following order:

* sets Timer class default properties
* sets DumboCar class default properties
* starts a loop, which is terminated when the game is over and which does the following things:
  + renders everything in the renderer queue
  + sleeps
  + processes user input
  + clears the renderer queue
  + puts the Field and Dumbo Car in the queue
  + puts Speed Info, Time Info and Points Info in the queue
  + updates Dumbo Car and moving objects
  + handles collisions and exit screen
  + produces a list of objects if rows 0 to 7 are not occupied
  + calculates Timer class properties

## DumboDriverMain Class

This class executes when the application is run. It does the following things:

* sets default Console parameters
* starts a loop, which is terminated on user input and which does the following things:
  + makes instances of IRenderer, IUserController and Engine
  + renders everything in the renderer queue
  + runs the Engine
  + asks for user input about playing again
  + clears the Console

# Singleton Design Pattern

In the project the Singleton design pattern is applied to the DumboCar Class. Since the game is not multiplayer and requires only one Dumbo Car, it is appropriate to use the pattern in this case. The pattern is achieved by:

* creating a static member in DumboCar class
* making the constructor of the class private
* creating a static public method that returns a reference to the static member
* accessing the DumboCar class via the static public method, which is responsible for creating its class unique instance in case it is not created yet and to return that instance