Programming in C#
COSC 223-0001
Programming Assignment 8
Due Date: Apr 21, 2009 (before class)

Instructions

The instructions below give important information about how the program should be submitted and written:

- Complete the programming assignment as given in the instructions below.
- All programs will be graded using the Visual Studio .NET 2008 IDE.
- Email your assignment to dar5p@mcs.uvawise.edu before class on the due date.
- Your project should be sent as a .zip file attached to the email message.
- Your name (Last Name, First Name) and "Programming Assignment #1" (or whichever program number this is) should appear on the "Subject:" line of your email message. Points might be deducted if this is not the case.
- Make sure to "Cc:" yourself on this mail message, then verify that what you sent is indeed a valid project file, and that it is the one you intended to send.
- Late programs will not be accepted.
- If you send more than one email, only the last one you send will be graded.
- No group work is permitted.
- Once you sit down at a computer to work on an assignment, you should be doing your own work. (Review the syllabus if necessary).

In addition, the following instructions give important information about how the program should be formatted:

- You should have identifying header-block comments (name, student number, course and section number, program number) along with comments describing the purpose of the program at the top of your source program.
- Each program should also have several internal comments to explain particular sections of code that otherwise might be somewhat confusing to whoever is reading and grading your program.
- If your program's indentation (or lack of it) makes your program too difficult to grade, points will be deducted.
- Similarly, if your choice of variable names makes your program too difficult to grade, points will be deducted.

- You must not change the order of the input, add extra input values, or otherwise modify the input stream in any manner.
- You must not change the output, or add any extra output to the program you turn in. While using many extra "cout" statements is an excellent way to debug a program, be sure that all the extra output statements are either removed or commented out before you turn in your program. Extra output can make your program much more difficult to grade, and anytime this happens points might be deducted.

Programming Assignment

For this assignment, you will again need to create a Windows Form Application. This assignment is the most complex yet and will require the use of most, if not all of the skills we have learned thus far this semester. The assignment has an extended due date of four weeks, and will count for four times as much as normal one week assignments.

Your assignment is to create a digital version of a board game that is closely based on an Avalon Hill copyright owned game called RoboRally. Our version is going to be a derivative work called RoboRampage. However, your code should not be distributed outside of this class due to copyright concerns.

In addition, we will be making use of code we wrote in our second and third programming projects concerning robot, and command card deck objects.

The application should create a game board that is a 12 x 12 grid representing a playing field that is navigated by 1 player controlled robot and three computer controlled robots. Each robot exists on a single grid block. Not all grid blocks are the same however. Some grid blocks are open space, some grid blocks are pits (which kills (causes them to lose a life) robots if they "fall in"), some grid blocks are turntables which rotate the robot 90 degrees to the left, some are turntables that rotate the robot 90 degrees to the right, some are treadmills that slide the robot either up/down/left/right 1 space, some are treadmills that slide the robot either up/down/left/right 2 spaces, some grid blocks have directional lasers (up/down/left/right), some grids have goal flags (there will be 3 on the board, numbered 1-3). The specific layout of the board is up to you, but be sure to exercise sound object-oriented design when defining the board and the elements on the board.

Each robot has a laser that points forward. The player robot also has five command registers, nine hit points, and three lives. The computer controlled robots are controlled by an algorithm of your choice but can only move according to the commands dealt to each robot from the single deck of command cards. They can be destroyed but are never fully eliminated from the game (see below).

The game progresses as follows: Robots begin in positions along the bottom row of the game board. The player is dealt nine command cards from the shuffled deck. He/she then has 1 minute to choose command cards and put them in registers 1 through 5 in the order that they should be executed by the robot. After 1 minute, unassigned registers will be randomly chosen from available cards.

At this point the 1^{st} game round is ready to begin. Each game round consists of 5 steps (coresponding to the five registers). The steps progress as follows:

1. In the n^{th} step the n^{th} register is executed by the player robot
2. The computer controlled robots then move (if they collide with the player robot then they push it across the board)
3. Treadmills are activated causing any robots on the treadmill grid to "slide" across the board
4. Turntables turn causing any robots on a turntable grid to rotate 90 degrees in the correct direction
5. All lasers (including each robot's laser) fire across the entire board (vertically or horizontally) until hitting something or exiting the playing grid, if a robot is hit then it loses a hit point.

Each of these five steps should be displayed to the user as discrete parts of the game appearing in sequence of object movement on the board.

After each step the player robot "reaches" a grid position. The goal of the game is for the player to reach each of the three goal flag grid blocks in order of their number (1 through 3), before running out of lives.

The first round is now over. After each round all robot registers are cleared with the following exception: for each hit taken by the player robot over four, one register is "locked" and its contents cannot be cleared. The first register to be locked is register five, followed by register four, etc. At the beginning of each successive round each player is again dealt command cards at random from the entire original deck but now with the restriction that each player only receives 9 minus the number of hits taken by the player's robot. For instance, if a player's robot has taken 6 hits then only 3 cards will be dealt to the player, the fourth and fifth register will be locked, so the 3 cards will be used to set the first 3 registers.

After cards have been dealt, the player will again have 1 minute to choose command cards for open registers before they are randomly chosen instead.

If the player robot receives nine hits then it is destroyed. Likewise, if a robot runs into a pit grid block, or runs off the playing field then it is destroyed. Once destroyed the robot loses a "life" (if it is the player robot) and starts over with fresh hit points at its initial position. However, even after being destroyed the previously visited goal flags still count. If the player robot loses all three lives then the game is over.

The game is won after the player robot has ended a step on each of the three goal flag grid blocks in order of their number.

How you create the game using Visual C# will be largely up to you. The game application should contain a visual playing grid, images to represent each of the game elements, a 1-minute timer, a way for command cards to be presented to the user and selected for each of the five registers, a way for the player to see have commands selected for the three computer-controlled robots revealed to them one at a time, a button to indicate that the registers are set, and a step button to indicate that the next step should be played.

As far as coding is concerned, you should of course make use of the Card and Deck classes from earlier as well as the Robot class. You should also use new derived classes and polymorphism to handle the board and other game elements.

The game should be as interactive, engaging, smooth and intuitive to use as possible. You can obtain extra credit by adding a game reset functionality and/or a help menu.

Your program will again be graded based on usability, a major part of which will be a consistent look-and-feel (make your program visually pleasing and interesting) as well as a straightforward organization of controls (the organization should be robust even assuming window resizing). You are allowed to use any and all resources available to you in order to create this Visual C# Windows Form application and may use any form controls you like where not specifically prescribed above.

As before, when you have completed the assignment you will need to compress the solution folder into a .zip file. This is the file that you need to e-mail to me (YourLastNameFirstNamePA7.zip).