### IN613 C++ 2015 Assignment 1: Tetris

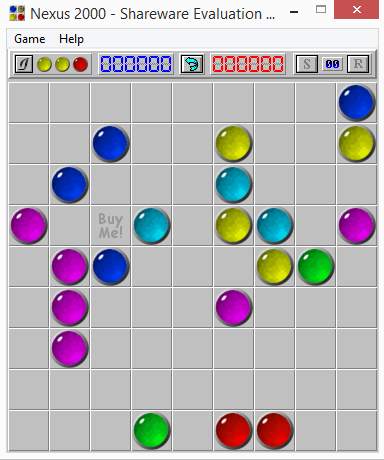
**Due Date: Thursday 15th October, 9.00 am**

**Group Size: This assignment is done individually.**

**Value: 30% of total course mark.**

**Definition:**

For this assignment you are to implement Nexus 2k provided. The game is played on a grid of 9 x 9 squares. The purpose of the game is to move balls of different colors and arrange them in a sequence of 5 or more. Doing so the sequence will be deleted and points will be earned. Every time a move is done (except when clearing a sequence) 3 random balls are being generated. The game ends when all the cells are filled with balls. The image below shows a typical Nexus game in progress:



**Minimum Functionality:**

Your implementation of Nexus must:

1. Be written using C++/CLI or Native C++
2. Have a Start button.
3. Provide at grid of a 9 X 9 grid of squares.
4. Have at least 6 different colored balls (provided)
5. Generate 3 new balls when a move is done.
6. Implement path finding for moving balls across the board
7. Implement sequence deletion and update score.
8. Implement undo functionality.
9. Implement high score.

**Extra Credit Functionality:**

Extra credit is available for any of the following. A maximum of 7% extra credit will be given. If you have other ideas, please check with me first.

|  |  |
| --- | --- |
|  | **Maximum Extra Credit** |
| Testing | 1% |
| Creative scoring algorithm | 1% |
| Sound (must be user-controlled) | 1% |
| Shortest path | 4% |

**Marking Schedule:**

|  |  |  |
| --- | --- | --- |
| **Component** | **Comment** | **Mark** |
| Project plan | Include class/modules structure sketches, user stories, development plan. | 10% |
| Code commenting | Function header block comments are required. In-line commenting as needed. Remember that comments serve to describe the logic, not to translate the commands. Please comment closing curly brackets clearly. | 10% |
| Architecture | Responsibilities should be assigned to each class/module in a sensible way. | 10% |
| Code elegance | Includes correct use of subroutines, algorithmic conciseness, meaningful variable naming, sensible flow of control, etc. | 25% |
| Functionality & Robustness | As described above. Includes error-checking, feedback and bug freeness. | 45% |