**Software Report Breakout Game**

Lynn Hacquebord 3761045  
Kevin Hutchinson 3741001  
Marijn Mengerink 3803627

**The Program was written in C# using XNA Framework**

**1. List of implemented features**The following features were mandatory to implement in the breakout game:  
1) Three types of balls (normal, electric, ghost)  
2) Three types of blocks (normal, bonus, transformer)  
3) A starting amount of balls  
4) A high score list with a total of three high scores, to be saved on quitting the game.

The high score list will be saved on modification rather than on quitting to avoid problems with irregular program termination. Aside from change, all of the above features along with the basic rules for breakout have been implemented.

On top of this the following features were also implemented:  
1) A starting menu with 2 buttons (play and quit)  
2) Level loading from files for easy extensibility

Controls:  
Buttons are interacted with using the mouse  
Spacebar is used to launch the ball at the start  
The paddle aligns with the mouse on the X-axis  
If the player managed to reach a high score he can type in his name (max 20 chars) using his keyboard.

**2. Unit testing**

Unit testing was preformed on 8 different classes:

|  |  |
| --- | --- |
| Class | Description |
| Ball | The ball in the game. This class handles its collision with other objects |
| Button | A button class for the menu |
| FileManager | For Splitting path names and creating files |
| HighScoreReader | For reading the high score file and editing it |
| InsertMenu | For inserting a new high score in the high score list |
| LevelLoader | For loading levels from files |
| Paddle | For handling paddle input |
| Transformer | For transforming the ball object |

These 8 classes have a total 18 unit tests for testing their non-trivial methods.

An object mock class has been created to mimic behavior of classes so that individual unit testing can be preformed

All other classes and methods in this project that were not unit tested are either trivial or a subsystem of already tested software. Note that XNA framework was used in this program and thus the content loading and drawing of that was not tested. We assume that the features of XNA are well tested as this framework has been used widely.

**3. Unit testing conclusion**

Having tested all untested and non-trivial classes and methods, one can only conclude that all unit testing up to this point is thoroughly completed and without additions and modification no further unit testing should be necessary if both the system test and the acceptance test require no individual changes to the classes. On class change however, the unit tests should be rerun and failed asserts should be corrected.