# IX5110001 Programming 2
# Assignment 1 –Breakout

| | |
|---|---|
| Study Block 3: | 2022 |
| Due date to be handed in: | **Week 7 Tuesday 7th June 9am** |
| Due date for Design: | Week 6 Tuesday 31th May 9am |
| Submission to Moodle: | Design Documentation, Code and Report |
| Contact Lecturer: | Tariq Khan |
| Total Marks | 100 |
| Weighting/Contribution: | 30% to final marks |
| Learning outcomes covered: | 1, 2 |
| Version: | 2.0 |

## Assignment Brief

For this assignment, you will create the arcade game "Breakout", where the user attempts to destroy layers of bricks at the top of the screen with a bouncing ball. The user can move the paddle back and forth across the bottom of the screen with the mouse and bounce the ball off the paddle. If the ball misses the paddle and goes off the bottom of the screen, the game ends. The ball bounces off the sides of the screen.

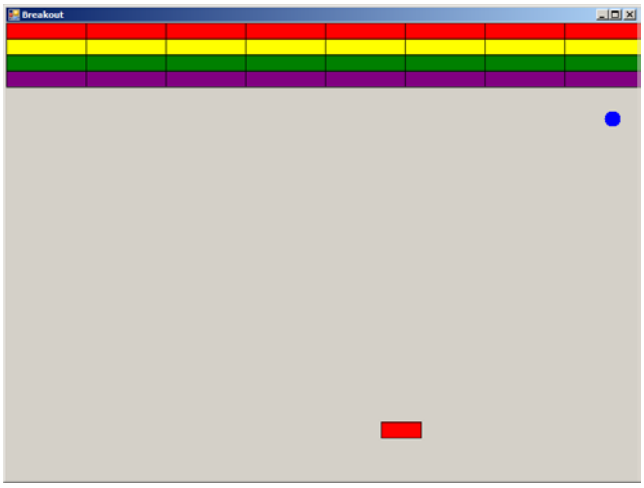
## Learning Outcomes

At the successful completion of this course, students will be able to:

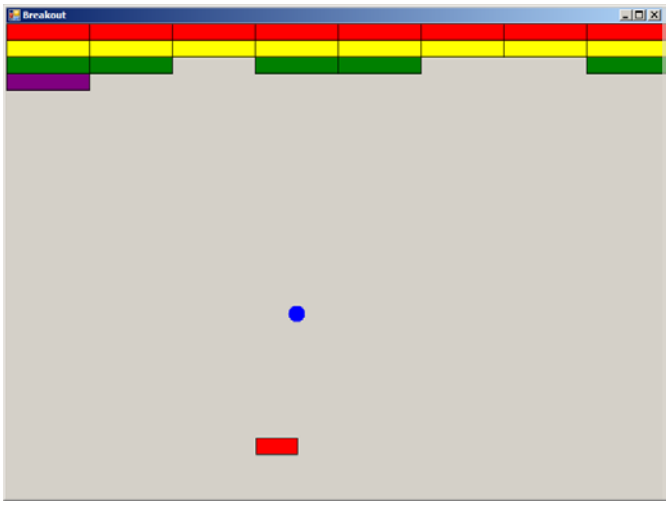
1. Build interactive, event-driven GUI applications using pre-built components.
2. Declare and implement user-defined classes using encapsulation, inheritance and polymorphism.

## Details

The screen shot below shows an example of the simplified version of Breakout required for this assignment. It contains a red paddle at the bottom of the screen, the ball and four layers of bricks at the top of the screen.




At the beginning of the game




During the game

## Functional Requirements

1. Your solution must be Object-Oriented.
2. Your game should be driven by a single Timer.
3. Your game must provide a ball, a paddle and a bank of bricks at the top of the screen.
4. The ball must bounce off the edges of the screen. The ball must also bounce off any brick at the top of the screen, causing the brick to disappear (or explode). The ball can bounce off the paddle at the bottom of the screen. If the ball misses the paddle and falls off the bottom of the screen, the game ends.
5. The paddle is controlled by the mouse (or the arrow keys). As the mouse is moved to the left or right, the paddle must follow the mouse's horizontal location. The mouse cursor must be inside playing field to steer the paddle.
6. The bricks should be managed within a list by the Manager class.
7. Each time a brick is hit, 10 points should be added to the score and clearly displayed and labelled on the screen.
8. When the game is lost (the ball falls off the bottom of the screen), the user must be given appropriate feedback.
9. When the game is won (all bricks have been destroyed), the user must be given appropriate feedback.
10. Your game must stop, when either the win or loss conditions are met.
11. Your game does **not** need to provide New Game or Quit buttons. The game should begin when the program is executed. Users can employ the Windows close box to exit the program.
12. Your game must be aesthetically pleasing, (i.e. have attractive screen layout and colour scheme).
13. You must use double buffering to control the flickering when objects are drawn on the form.

## Class Structure

Your program must be Object-Oriented. You must, therefore, declare the necessary Classes and instances of these Classes. You will need objects to represent the Ball, the Brick, the Manager and the Paddle. Note that marks will be allocated for the Object-Oriented correctness of your implementation.

Your Classes will need Fields, Properties and Methods. For example, all objects need to know where they are on the screen, their size and their colour. They all need to how to know, how to

draw themselves, and so on. The ball and paddle also need to know how to move themselves. The ball will need to know how to determine if it has collided with another game entity. The Manager will need to know his score, and how to correctly implement his animation.

You need to identify all fields, properties and methods when you pseudocode your application.

## Getting Input from the Mouse

When a user moves the mouse, a MouseMove event is generated. For the Form's MouseMove event, the event method signature is:

```
private void Form1_MouseMove(object sender, MouseEventArgs e)
```

You need to write a handler for this Form event. The argument you are interested in is `MouseEventArgs e`, which contains `e.Location`, the current position of the mouse.

## Getting Input from the Keyboard (Arrow keys)

When a user presses a key on the keyboard, a KeyDown event is generated. For the Form's KeyDown event, the event method signature is:

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
```

You need to write a handler for this Form event. The argument you are interested in is `KeyEventArgs e`, which contains the value of the pressed key. The arrow key values are `Keys.Left, Keys.Right, Keys.Up` and `Keys.Down` for the corresponding arrow key. When the user presses the Down arrow key, for example, the system generates a `Form1_KeyDown` event, and passes in the argument `e`, whose value is `Keys.Down`. In your `Form1_KeyDown` event handler, you can have statements like:

```
switch (e.KeyCode)
{
    case Keys.Left:
        //do something in response to the left arrow key
        break;
 ……..
```

The Form's `KeyPreview` property must be set to True. Otherwise, it won't be able to respond to the `KeyDown` event.

## Optional Extra Credit Functionality (examples)

1. Your game is scalable to larger sets, for example to add more bricks, extra balls to the game without rewriting code.
2. Keep a record of the user's best score (the number of bricks destroyed before the ball falls off the bottom of the screen) and display this value on the screen. The best score is calculated for the current execution of the application.
3. Provide Bonus points, for example, extra balls are available at 2000 points, or points are allocated for the time the ball is kept bouncing.
4. Provide a Pause button to pause the game.
5. Provide game levels.
6. Allow user to play again.

There are many versions of Breakout available online, for example:

https://elgoog.im/breakout/

https://playpager.com/play-brickout/index.html

## Submission

1. Submit your design documentation with UML class diagrams and pseudo-code showing the complete design of your application on **Wednesday 24th June 8pm**

2. For your project must submit your application through Moodle submission link. **Compress the Visual Studio Project folder containing all of the project files and you also need to submit Microsoft Word or pdf file for user guide.**

   I will highly recommend to use version control system (VCS) for this project and gave me access for you repository. Your repository must be private. For VCS Explore :
   - Bitbucket https://bitbucket.org/
   - Github https://github.com/
   - Gitlab https://about.gitlab.com/

## Marking Schedule Assignment 1:

| Feature | Comment | Percentage of Total Mark |
|---|---|---|
| Design documentation | Top down deconstruction of the problem<br>Form design<br>UML class design<br>Iterative refinement of methods<br>Design of algorithms (eg collision detection) | 10 |
| Code commenting | Block comments at beginning of program,<br>for every class, and<br>for every method.<br>In-line comments as required. | 5 |
| Basic Functionality | Functionality as described above.<br>Your program must compile. | 45 |
| Code elegance | Class design<br>Modularity<br>Constants<br>Enumerations<br>Algorithmic elegance<br>Error handling | 25 |
| Product aesthetics | Attractive screen layout, colour scheme etc.<br>The user should immediately see how to play.<br>Any additional instructions should be included.<br>Sound<br>Double buffered | 5 |
| Bonus points | Unique extensions or additional features that improve the game's functionality, such as:<br>Ability for user to change colours, shapes, number of bricks….<br>Multiple forms<br>Bonus points<br>Pause button<br>Levels<br>Play again option | 5 |
| User guide | Develop a user guide with screenshots of your application | 5 |