Bachelor of Computing Systems		
Course No: ISCG 6421	**GUI Programming**	**Level: 6** **Credits: 15**

Student Name:	**Student ID:**
Assessment Type: Breakout Game	**Weighting: 20%**
Due Date and Time: 29th October 11:59pm **Checkpoint : Wednesday 15th October**	**Total Marks: 100**

Student declaration

I confirm that:

- This is an original assessment and is entirely my own work.
- The work I am submitting for this assessment is free of plagiarism. I have read and understood the Academic Integrity Procedure (including the key principles about using the use of Generative Artificial Intelligence (GenAI) listed in Section 3.2). I have also read and understood the Student Disciplinary Statue.
- Where I have used ideas, tables, diagrams etc of other writers, I have acknowledged the source in every case.

Students Signature:	**Date:**

Assessment Mapping

After completing this assessment, the student will have met the following learning outcomes:

Learning Outcome	Tasks
1. Demonstrate competency in the use of a GUI programming language and environment.	1,2,3,4
3. Demonstrate the testing, debugging and documentation of GUI programs	5, 6

Assessment information/guidelines:

- This is individual assessment.
- Read the provided requirements.
- Brief report for testing with brief explanation and above coversheet.
- You must use the provided startup project.
- You are allowed to use Generative AI tools (e.g., Copilot, ChatGPT) for assistance, but you must clearly document their use.
- After your reference list, include:
    - A list of GenAI tools used.
    - A brief explanation of how each tool contributed to your work (e.g., code suggestions, debugging help, UI design inspiration).
    - Ensure that all sources, including AI-generated content, are properly acknowledged and cited where applicable.

Assessment submission instructions:

You must upload following files via the link provided on the Moodle course page:

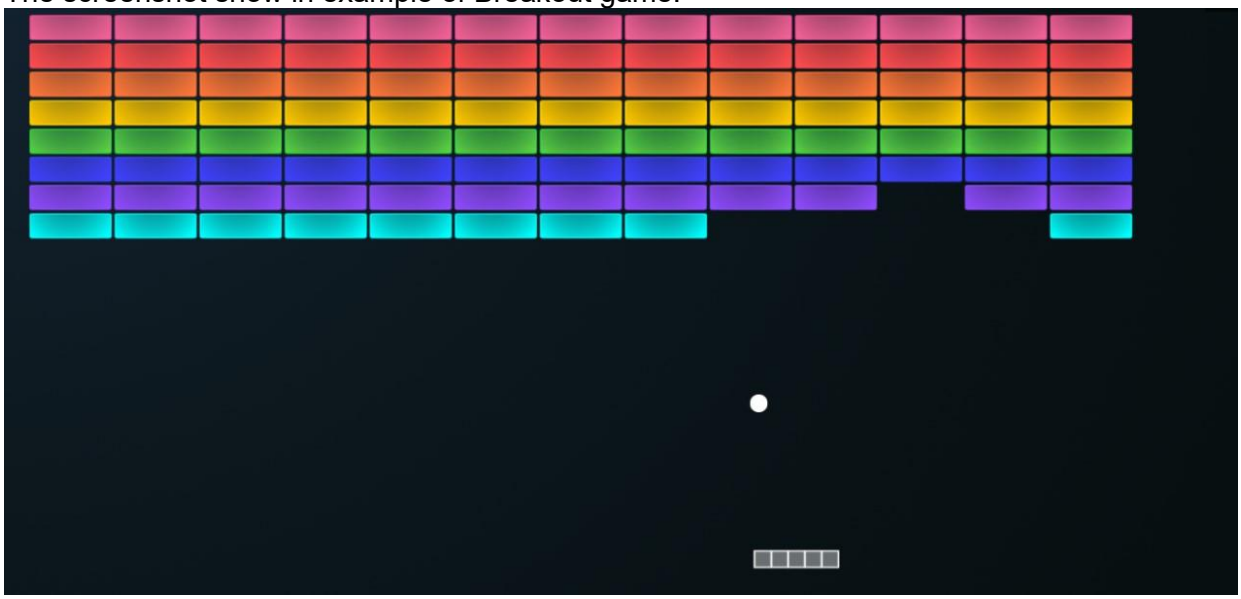
- Document with Test cases either in pdf or Microsoft word file format.
- Your Visual Studio project with all necessary files (zip it).
- Add the above coversheet to your report and upload it on Moodle.

Problem statement

Breakout is and arcade game developed in 1970. In Breakout a layer of bricks lines are at the top of the form and the goal is to destroy them by bouncing ball off a paddle into the bricks. The user can move the paddle back and forth at the bottom of the screen with arrow keys or mouse. The ball bounce off the paddle and off the sides of the form. if the ball miss the paddle at the bottom the game ends.

Your Task is to create Breakout game using C# and .NET Framework Windows Forms.

The screenshot show in example of Breakout game.



https://en.wikipedia.org/wiki/Breakout_(video_game)

1. Requirements

You are provided with a C# starter project that you must use for this assignment. The starter project contains all necessary classes and file for you to complete the assignment. However, you can add more Forms or classes.

a) **Object-Oriented Design**: The game should be designed and implemented using object-oriented programming principles.
b) **Initial Screen:** Implement an introductory splash screen that allows the player to start the game.
c) **Game Interface:** The interface should display a paddle, a ball, and a layer of bricks at the top of the screen. Bricks must respond to ball collisions by either weakening (decreasing in durability) or disappearing when their durability reaches zero.
d) **Ball Dynamics:** The ball should rebound off the game window's edges. It must also bounce off the paddle. If the ball misses the paddle and hits the bottom edge, the game ends.
e) **Paddle Control:** The player should control the paddle using either the mouse or arrow keys.
f) **Animation and Timing:** Use a single timer to manage game animations and movements.
g) **Pause Feature:** Include an option for players to pause and resume the game.

h) **Scoring System:** Display the score, updating it by 10 points each time a brick is hit.
i) **Power-Ups:** Integrate at least two types of power-ups (e.g., paddle size increase, multi-ball feature) that activate when certain bricks are hit.
j) **End-of-Game Feedback:** Provide clear feedback and options to replay the game upon either losing or winning (all bricks destroyed).
k) **Additional Features:** Add at least two unique features to enhance gameplay and engagement. Suggestions include a leveling system, different brick types (e.g., unbreakable, explosive), or special level challenges.
l) Visual Design:
Ensure the game's visual design is visually appealing and coherent, contributing to an enjoyable user experience.

2. Classes

Your project must be object oriented. You should add following classes:

- Ball
- Brick
- Paddle
- Manager

Your class will need fields and methods. For example for the ball, brick and Paddle objects needs to know where it is on the screen, its size and colour. The ball and paddle also need to know how to move. The ball should also know if it is collided with bricks or paddle. The Manger will manage all the objects, manage animation and calculate game.

3. Extra Features Examples

- Add different levels to game
- Add appropriate sounds to game
- Keep track of user score especially record the highest score or top three scores
- Allows the players to start, pause, resume, and restart games.
- Provides a way to Save the current game to a file.
- Provides a way to Load a saved game from a file and continue playing the loaded game.

4. Getting input from the Keyboard

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:

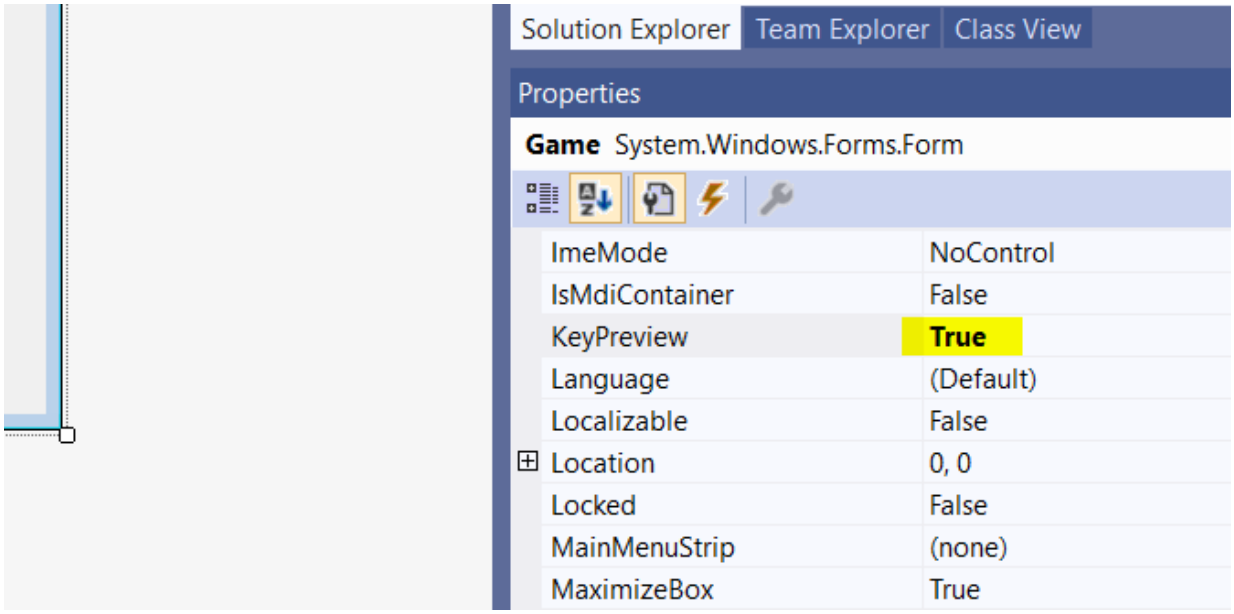
```
private void Game_KeyDown(object sender, KeyEventArgs e)
```

Game is the name of the form. Following is the sample code

```
private void Game_KeyDown(object sender, KeyEventArgs e)
  {
      switch (e.KeyCode)
      {
          case Keys.Left:
              //write code in response to the left arrow key
              break;
          case Keys.Right:
              //write code in response to the right 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



5. Required Tests

This is the minimum number of tests you need in your testing documentation:

1. Interface is displayed correctly when the program runs
2. Bricks are displayed correctly
3. Paddle is move with key Press or mouse move
4. The ball is moving
5. The ball is bouncing off the paddle
6. The ball is making the bricks disappear when hit by the ball.
7. Score is calculated correctly
8. The won message displayed when the user won
9. The lost message displayed when the user lost
10. User is getting an option to replay the game when the game is won or lost

Please note that your tests can involve more than one step and **must** be reproducible (i.e. explicit test data and user actions) and independent of each other (i.e. please do not use the output of one test as the input to another test.)

Test Cases

Please using the following format for your test cases

Requirement to test	Test Data Input	Expected Outcomes	Actual Outcomes

6. Required Unit Tests

You must write at least 7 Unit tests to test methods for your choice. You can select methods from any class, consider following classes.

- Ball
- Paddle
- Brick

Delivery

A soft copy must be uploaded onto **Moodle** as a single **.zip** file prior to the deadline and it must comprise:

- The testing documentation.
- ALL files needed to compile and run your application from the **Visual Studio Community 2022 or 2019**. 20-30 marks will be deducted if this is not done.

Please note that it is important to upload the correct version of your assignment onto Moodle. If you submit the wrong version onto Moodle, please notify me by email before the deadline date or late penalties may be incurred.

Checkpoints

There is an expectation that you will have completed certain aspects of your assignment at each checkpoint submission. If your submission does not meet the expectations of the checkpoint, your overall mark for the final submission may be penalised.

Expectations

Checkpoint :

- User interfaces are designed, and the ball is coded for the game
- Classes are added for Ball, Paddle and Bricks.

Demo

You must give demo of your assignment and answer all questions about your code. Without demo and Q&A your assignment will not be marked.

Marking

Criteria	Marks	Actual
• Game Interface set up correctly (similarly to the one shown on page 11).	8	
• Main functionalities are added to the game form. 1. The ball must bounce from the edges of the Form. 2. Firmness is added to bricks (1 to 4) 3. Power-ups added 4. The ball must also bounce off the bricks at the top of the screen and causing the brick to explode (disappear) if the firmness is 1. 5. The ball must also bounce off the paddle at the bottom of the Form and if the ball misses the paddle the game ends. 6. The user must be able to control the paddle using mouse or the arrows keys. 7. The game must use single Timer to control animation. (3 Marks for each)	24	
• Score is calculated correctly	5	
• Appropriate scoring calculated for power-ups	4	
• Appropriate messages displayed when the game is lost or won	5	
• An option is added to replay the game	3	
• Appropriate two extra features are added.	6	
• Main or welcome form is added which allows the user to begin a game.	5	
• Application is object oriented (6 marks), Modular (4 marks) and Algorithmic elegance (4 Marks)	14	
• Game is aesthetically pleasing	5	
Testing documentation with appropriate tests	8	
7 Unit tests for appropriate methods	7	
Naming conventions and correct internal documentation	6	
You must give demo of your assignment and answer all questions about your code. 50 marks will be deducted if this is not done.		
Total:	**100**	

- Marks will be deducted for any requirement (or for multi-part requirements, or each part of a requirement) that is not fully implemented or has no or insufficient testing.
- In the **interface behaviour and data processing** section, marks will be deducted for any requirement (or for multi-part requirements, or each part of a requirement) that is not well covered in the test plan or is reported as working in the test plan but does not

work when the assignment is marked. You are also expected to handle exceptions, using message boxes and "**try and catch**" blocks.

- Check "Programming Standards for C Sharp Courses" You can also find some at the end of this document. The standards for C# programming in this document MUST be followed. In particular, this includes putting meaningful comments at the beginning of **each and every** method in the standard format as given in the document.

Assignment Delivery

Electronic submission of all necessary files is required for ALL assignments and must be submitted prior to the due date and time. Assignments submitted after the due date and time without having received an extension through Extensions and Assessment Concessions (AC) will be penalised according to the following:

- 10% of marks deducted if submitted within 24hrs of the deadline
- 20% of marks deducted if submitted after 24hrs and up to 48hrs of the deadline
- 30% of marks deducted if submitted after 48hrs and up to 72hrs of the deadline
- No marks will be awarded for an assignment that is submitted later than 72hrs after the deadline.

For the purposes of academic integrity, students who haven't demonstrated progress work in the class time can be asked to demo/test their working code and explain logic to the lecturer individually after assignment submission.

Extensions and Assessment Concessions (AC)

If the unexpected happens or things don't go to plan, you may be able to apply for an extension or an Assessment Concession(AC) (formerly Affected Performance Consideration (APC)). use the following link for more information and how to apply.

https://www.unitec.ac.nz/extensions