

Lab 11 Report: Analyzing C# Console Games for Bugs

Course: CS202 Software Tools and Techniques for CSE
Author: Pathan Mohammad Rashid (22110187)

1. Introduction, Setup, and Tools

Objective: Learn to use the Visual Studio debugger to understand control flow and locate/fix runtime bugs in C# console games.

Environment:

- **OS:** Windows 11
 - **IDE:** Visual Studio 2022 Community Edition with .NET 6.0 SDK
 - **Repository:** `dotnet-console-games` (cloned from GitHub) [GitHub](#).
 - **Language:** C# with top-level statements (no explicit `Main()` needed)
-

2. Methodology and Execution

2.1 Cloning and Opening the Solution

```
git clone https://github.com/dotnet/dotnet-console-games.git
cd dotnet-console-games
```

- Open `dotnet-console-games.sln` in Visual Studio and allow NuGet package restore.

2.2 Selecting the “Gravity” Game Project

- In **Solution Explorer**, expand **Projects** → **Gravity**.
- Entry point is in `Program.cs` via **top-level statements** (compiler generates `Main()` automatically).

2.3 Debugger Basics

1. **Set a breakpoint** at the first game-loop statement in `Program.cs` by clicking the left margin.
2. **Start Debugging** (F5) to pause at your breakpoint.

3. Use **Step Into (F11)** to enter methods, **Step Over (F10)** to execute without entering, and **Step Out (Shift+F11)** to finish the current method and return.
4. Observe variable values by hovering over them or inspecting in the **Autos/ Locals** windows.

Note: All the Screenshots are attached to the end of the report in order.

2.4 Mutation Testing: Injecting Five Bugs

Above the main loop, we applied five separate mutations, each illustrating a distinct bug. For each, record:

- **What?** Behavior observed
- **When?** Execution point
- **Why?** Root cause
- **Where?** File and line number

Bug 1: Incorrect Starting Position

```
415 -     return (j + 2, i + 1);  
415 +     return (j + 0, i + 0);
```

- **What?** Player spawns at the wrong coordinates.
- **When?** Game start, before rendering level.
- **Why?** Offset parameters changed from `(j+2, i+1)` to `(j, i)`.
- **Where?** `GetStartingPlayerPositionFromLevel()`, `Program.cs`, line 415.

Bug 2: Treating Spikes as Victory

```
638 - case 'X': gameState |= GameState.Died; break;  
638 + case 'X': gameState |= GameState.Won; break;
```

- **What?** Colliding with spikes instantly wins the game.
- **When?** Collision detection loop.
- **Why?** `GameState.Won` used instead of `GameState.Died`.
- **Where?** Collision check in `Program.cs`, line 638.

Bug 3: Level Skipping on Win

```
669 -     level++;  
669 +     level = level + 2;
```

- **What?** Pressing Enter after victory jumps two levels ahead.
- **When?** After `GameState.Won` branch, on level completion.
- **Why?** `level` increment mutated to add 2 instead of 1.
- **Where?** Win-handling block in `Program.cs`, line 669.

Bug 4: Rendering Out-of-Bounds Index

```

if (c is not 'X' and not '●' &&
-     PlayerPosition.X - 2 <= j &&
-     PlayerPosition.X + 2 >= j &&
-     PlayerPosition.Y - 1 <= i &&
-     PlayerPosition.Y + 1 >= i)

if (c is not 'X' and not '●' &&
+     PlayerPosition.X - 3 <= j &&
+     PlayerPosition.X + 3 >= j &&
+     PlayerPosition.Y - 2 <= i &&
+     PlayerPosition.Y + 2 >= i)

• What? IndexOutOfRangeException during render.
• When? In Render() when building render string.
• Why? Logical mutation removed Y-bounds check, causing invalid array access.
• Where? Rendering loop in Program.cs, line 717-720.

```

Bug 5: Overlapping Bit-flags for GameState

```

- Died      = 1 << 0,
851 + Died      = 1 << 1
- Won       = 1 << 1
851 + Won       = 1 << 1

```

-
- **What?** Died and Won share the same flag bit, making state ambiguous.
 - **When?** Anywhere `gameState.HasFlag(..)` is tested.
 - **Why?** Both enum values shifted by one bit rather than defining unique bits.
 - **Where?** `enum GameState` in `Program.cs`, line 851.

2.5 Fixing and Verifying Bugs

For each bug:

1. **Revert** the diff to original correct code.
 2. **Rebuild** (Ctrl+Shift+B) and **Run** (F5).
 3. **Confirm** expected behavior: correct spawn, spikes kill, single-level increment, no exception, distinct win/die states.
-

3. Results and Analysis

- **Bug 1 Fix:** Player now starts at intended offset (•) [.](#)
- **Bug 2 Fix:** Spikes correctly kill player, not win.
- **Bug 3 Fix:** Each win advances exactly one level.
- **Bug 4 Fix:** Render no longer throws `IndexOutOfRangeException`.
- **Bug 5 Fix:** `GameState.Died` and `GameState.Won` use distinct bits per [enum flags guidelines].

Key insight: systematic mutation testing quickly uncovers both logic and configuration errors [.](#)

4. Discussion and Conclusion

Challenges: Precisely locating the rendering-bounds bug required careful stepping and call-stack inspection. **Lessons Learned:**

- Visual Studio's step commands (F10/F11/Shift+F11) are essential for understanding control flow [Microsoft Learn](#) .
- Mutation testing (even manual) is a powerful technique to stress-test code behavior [Stryker Mutator](#).
- Proper use of bit-flags enums prevents unintended state overlap [Aaron Bos' Blog](#).

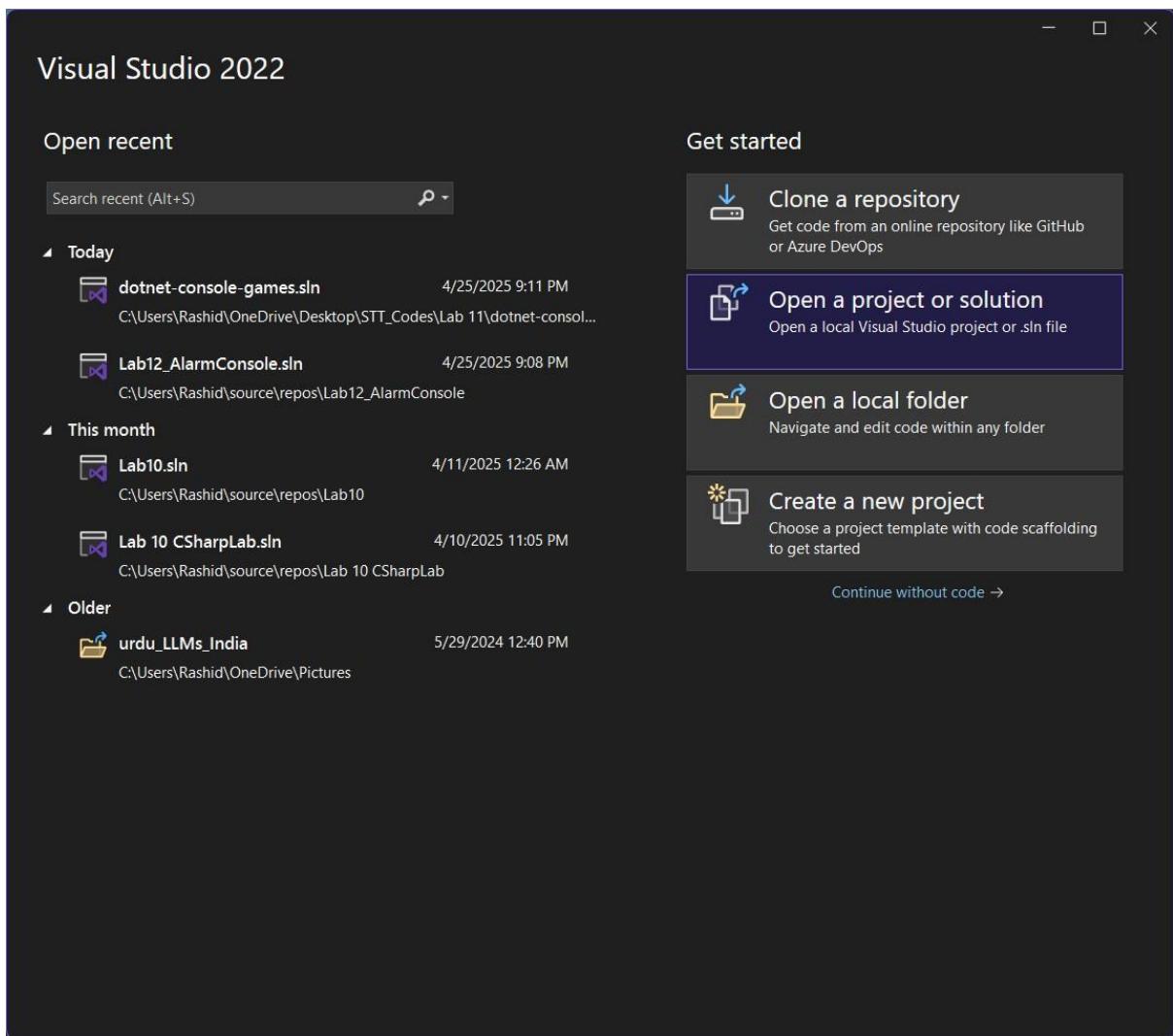
5. GitHub Repository

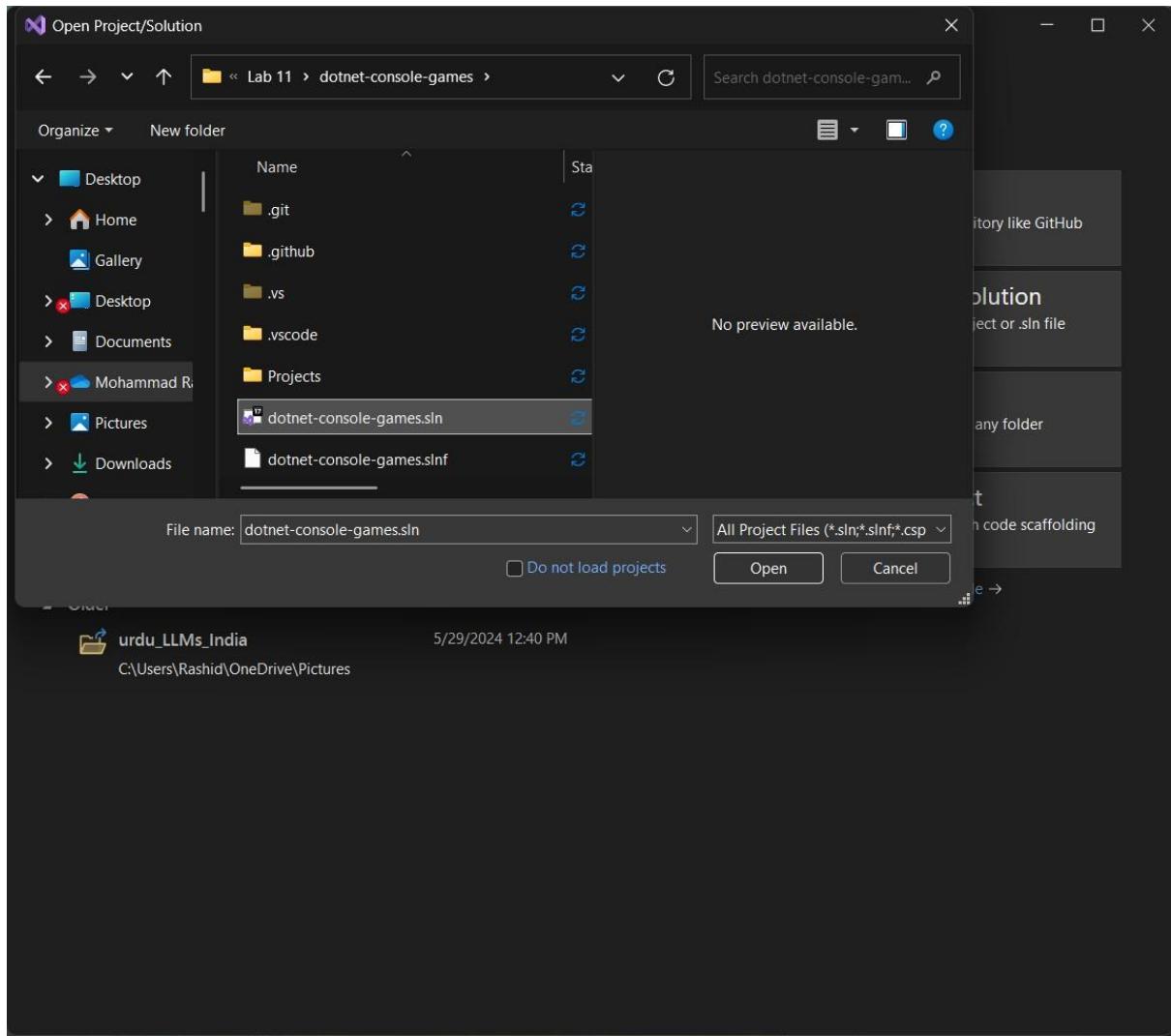
- All code files, the detailed report, and additional documentation can be found in my GitHub repository:
https://github.com/Pathan-Mohammad-Rashid/STT_Labs.git
- Onedrive Link: https://iitgnacin-my.sharepoint.com/:f/q/personal/22110187_iitgn_ac_in/EnECtzFbUNZKkBWY6aa8QJYBp6liTXfpuyRtJNMd1TwR6A?e=nsND4y

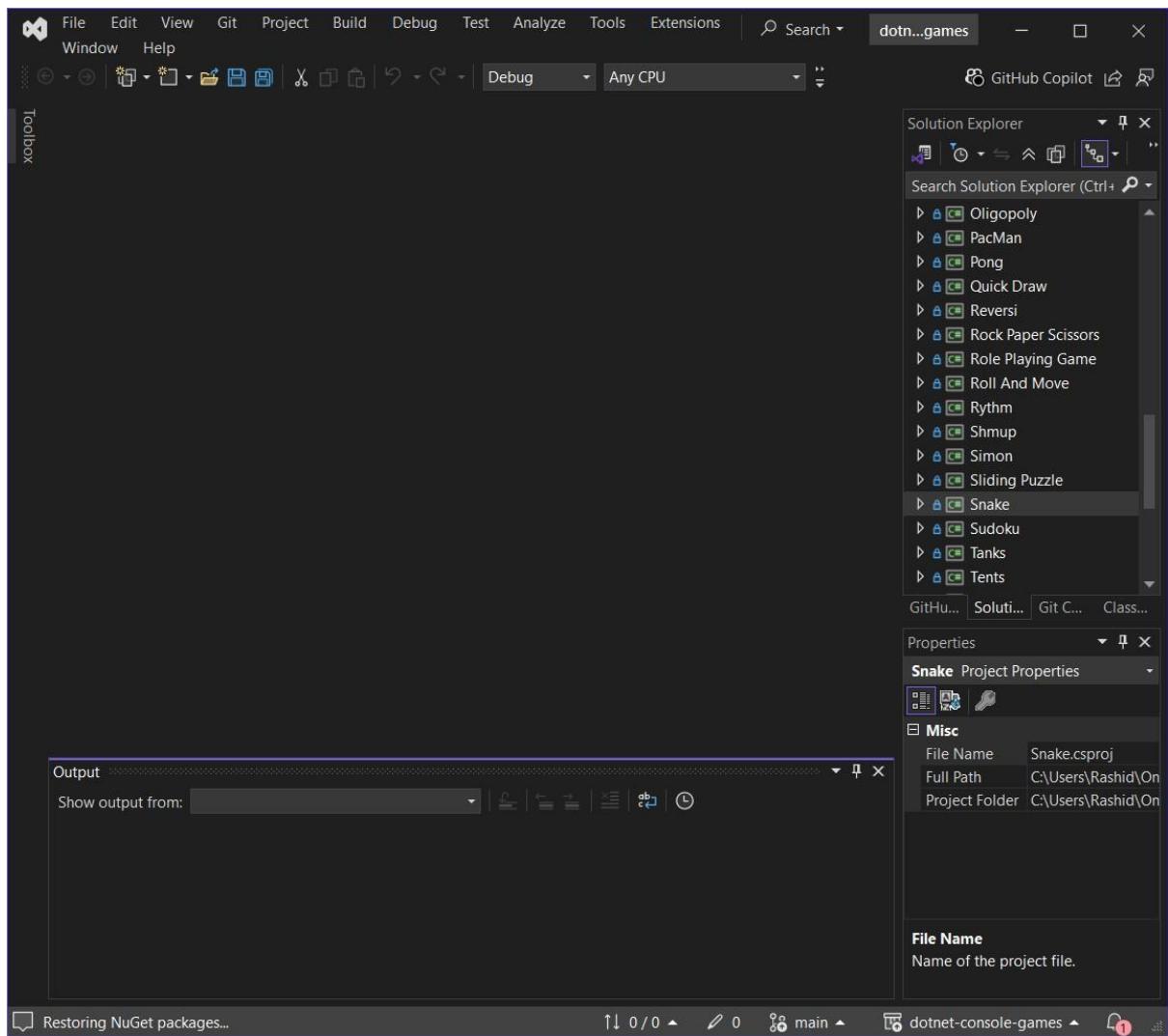
1. Lab 12: Environment Setup and Initialization

This section illustrates the initial setup process for Lab 12, including configuring the development environment and preparing the necessary files.

```
PS C:\Users\Rashid\OneDrive\Desktop\STT_Codes\Lab 11\lab11> git clone https://github.com/dotnet/dotnet-console-games.git
Cloning into 'dotnet-console-games'...
remote: Enumerating objects: 9991, done.
remote: Counting objects: 100% (99/99), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 9991 (delta 87), reused 75 (delta 75), pack-reused 9892 (from 2)
Receiving objects: 100% (9991/9991), 141.47 MiB | 7.77 MiB/s, done.
Resolving deltas: 65% (4384/6744)
Resolving deltas: 100% (6744/6744), done.
Updating files: 100% (661/661), done.
PS C:\Users\Rashid\OneDrive\Desktop\STT_Codes\Lab 11\lab11>
```







2. Game 1: Guess a Number

- Standard Execution Demonstrates the normal flow of the game without debugging intervention.
- Debugging with Breakpoints and Step Controls Showcases the use of breakpoints and debugging tools such as step into, step over, and step out for detailed code execution analysis.

The screenshot shows the Visual Studio IDE interface with the following details:

- MenuBar:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions.
- Search Bar:** Search dotn...games
- Toolbox:** Guess A Number
- Code Editor:** Program.cs (C# file) containing the following code:

```
1  using System;
2
3  int value = Random.Shared.Next(1, 101);
4  while (true)
5  {
6      Console.Write("Guess a number (1-100): ");
7      bool valid = int.TryParse(Console.ReadLine() ?? "", out int input);
8      if (!valid) Console.WriteLine("Invalid.");
9      else if (input == value) break;
10     else Console.WriteLine($"Incorrect. Too {input < value ? "Low" : "High"}.");
11 }
12 Console.WriteLine("You guessed it!");
13 Console.Write("Press any key to exit...");
14 Console.ReadKey(true);
```
- Solution Explorer:** Shows a solution named "dotnet-console-games" with the following projects:
 - Duck Hunt
 - Fighter
 - First Person Shooter
 - Flappy Bird
 - Flash Cards
 - Gravity
 - Guess A Number (selected)
 - Dependencies
 - Program.cs
 - README.md
 - Hangman
 - Helicopter
 - Hurdles
 - Lights Out
 - Mancala
 - Maze
- Properties Window:** Program.cs File Properties (Advanced tab selected)
 - Build Action: C# compiler
 - Copy to Output: Do not copy
 - Custom Tool
 - Custom Tool N

Misc

 - File Name: Program.cs
 - Full Path: C:\Users\Rashid\OneDrive\Desktop\dotnet-console-games\Program.cs

File Name

Name of the file or folder.
- Output Window:** Shows the console output of the application.

```
Guess a number (1-100): 50
Incorrect. Too Low.
Guess a number (1-100): 80
Incorrect. Too High.
Guess a number (1-100): 65
Incorrect. Too High.
Guess a number (1-100): 55
Incorrect. Too Low.
Guess a number (1-100): 60
Incorrect. Too Low.
Guess a number (1-100): 62
Incorrect. Too Low.
Guess a number (1-100): 63
You guessed it!
Press any key to exit...
```

The screenshot shows the Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Search, and Help. The toolbar below has icons for Undo, Redo, Cut, Copy, Paste, Find, Replace, and others. The main code editor window displays a C# program named 'Program.cs' with the following code:

```
using System;
int value = Random.Shared.Next(1, 101);
while (true)
{
    Console.WriteLine("Guess a number (1-100)");
    bool valid = int.TryParse(Console.ReadLine(), out value);
    if (!valid) Console.WriteLine("Invalid input");
    else if (input == value) break;
    else Console.WriteLine($"Incorrect. {value}");
}
Console.WriteLine("You guessed it!");
Console.WriteLine("Press any key to exit...");
Console.ReadKey(true);
```

The Diagnostic Tools window is open, showing a 'Diagnostics session: 1:02 minutes (1:02 min selected)' timeline. It includes sections for Events, Process Memory (MB), and CPU (% of all processors). The Solution Explorer window on the right lists various projects and files, including 'dotn...games', 'Duck Hunt', 'Fighter', 'First Person Shooter', 'Flappy Bird', 'Flash Cards', 'Gravity', 'Guess A Number', 'Dependencies', 'Program.cs', 'README.md', 'Hangman', 'Helicopter', 'Hurdles', 'Lights Out', 'Mancala', and 'Maze'. The Properties window is also visible.

This screenshot shows the Visual Studio interface after running the 'Program.cs' file. The Output window displays the console logs from the application's execution:

```
'Guess A Number.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\2.1.8\mscorlib.dll'.
'Guess A Number.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\2.1.8\System.Private.CoreLib.dll'.
'Guess A Number.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\2.1.8\System.Runtime.dll'.
The thread '.NET TP Worker' (20812) has exited with code 0 (0x0).
The thread '.NET TP Worker' (5984) has exited with code 0 (0x0).
The thread '.NET TP Worker' (5540) has exited with code 0 (0x0).
The program '[11048] Guess A Number.exe' has exited with code 0 (0x0).
```

The Diagnostic Tools window shows the same session details and memory usage. The Solution Explorer window lists the same projects and files as the previous screenshot. The Properties window is also present.

The screenshot displays two separate sessions of Microsoft Visual Studio running on a Windows operating system. Both sessions are focused on a project named "dotnet-console-games" containing a file "Program.cs".

Top Session (Diagnostic Session: 11 seconds):

- Code:**

```
1  using System;
2
3  int value = Random.Shared.Next(1, 101);
4
5  while (true)
6  {
7      Console.WriteLine("Guess a number (1-100): ");
8      bool valid = int.TryParse(Console.ReadLine());
9      if (valid) Console.WriteLine("Invalid.");
10     else if (input == value) break;
11     else Console.WriteLine("Incorrect. Too {{input}}");
12
13     Console.WriteLine("You guessed it!");
14     Console.ReadKey(true);
15 }
```
- Diagnostic Tools:** Shows a summary of performance metrics over 11 seconds. Key values include:

 - Process Memory: 11 MB
 - CPU (% of all processors): 100%

Bottom Session (Diagnostic Session: 1 second):

- Code:** The same "Program.cs" code as the top session.
- Diagnostic Tools:** Shows a summary of performance metrics over 1 second. Key values include:

 - Process Memory: 5 MB
 - CPU (% of all processors): 100%

User Interface and Environment:

- Solution Explorer:** Shows the project structure and files.
- Taskbar:** Displays the Windows Start button, taskbar icons for various applications like File Explorer, Edge, and File History, and the system tray showing the date and time (4/25/2025, 9:28 PM).
- System Tray:** Shows the weather (33°C), battery status, and other system icons.

The screenshot displays two windows of a Microsoft Windows desktop environment. Both windows are titled "dotnet-games" and show a console application named "Guess A Number".

Top Window (Debugger View):

- Code:** Shows the C# source code for the "Program.cs" file. The code generates a random number between 1 and 100, prompts the user for an input, and checks if it's correct or too low.
- Variables:** Shows the current values of variables: `input` (10), `valid` (true), and `value` (46).
- Call Stack:** Shows the current call stack.
- Diagnostic Tools:** Shows performance monitoring data for the process, including CPU usage, memory usage, and event logs.
- Solution Explorer:** Shows the project structure with files like README.md, Hangman, Helicopter, Hurdles, Lights Out, Mancala, Maze, Memory, Minesweeper, Oligopoly, PacMan, Pong, Quick Draw, Reversi, Rock Paper Scissors, Role Playing Game, Roll And Move, Rythm, Shmup, and Simon.

Bottom Window (Application Window):

- Output:** Displays the application's interaction with the user:

```
Guess a number (1-100): 10
Incorrect. Too Low.
Guess a number (1-100): |
```

The taskbar at the bottom of the screen shows other open applications, including a browser and various system icons.

File Edit View Git Project Build Debug Test Analyze Tools Extensions | Search | dotnet_games | Process: [2428] Guess A Number.exe | Lifecycle Events | Thread: [13504] Main Thread

Program.cs

```
1 using System;
2 
3 value = Random.Shared.Next(1, 101);
4 if (true)
5 {
6     Console.WriteLine("Guess a number (1-100): ");
7     bool valid = int.TryParse(Console.ReadLine() ?? "", out int valid);
8     if (!valid) Console.WriteLine("Invalid.");
9     else if (Input == value) break;
10    else Console.WriteLine($"Incorrect. Too {((Input < value) ? "Low" : "High")}. Try again!");
11    else WriteLine("You guessed it!");
12    else Write("Press any key to exit...");
```

Diagnostic Tools

Diagnostics session: 43 seconds (1 ms selected)

Add counter graphs by checking counters from counter options

Events

Process Memory (MB)

CPU % of all processors

Summary Events Counters Memory Usage

Events

.NET Counters

Memory Usage

Autos

Name	Value	Type
input	46	int
valid	true	bool
value	46	int

Call Stack

Search (Ctrl+E)

Solution Explorer

Search Solution Explorer

Guess A Number

Dependencies

Program.cs

README.md

Hangman

Helicopter

Hurdles

Lights Out

Mancala

Maze

Memory

Minesweeper

Oligopoly

PacMan

Pong

Quick Draw

Reversi

Rock Paper Scissors

Role Playing Game

Roll And More

Rhythm

Shmup

Simon

Summary Events Counters Memory Usage

Events

.NET Counters

Memory Usage

Autos Locals Watch 1

Call Stack Breakpoints Exceptions Commas Immediate Output Solution Explorer Git Changes

Ready

Sports headline New Zealand Cr...

File Edit View Git Project Build Debug Test Analyze Tools Extensions | Search | dotnet_games | Process: [2428] Guess A Number.exe | Lifecycle Events | Thread: [13504] Main Thread

Program.cs

```
1 using System;
2 
3 value = Random.Shared.Next(1, 101);
4 if (true)
5 {
6     Console.WriteLine("Guess a number (1-100): ");
7     bool valid = int.TryParse(Console.ReadLine() ?? "", out int valid);
8     if (!valid) Console.WriteLine("Invalid.");
9     else If (Input == value) break;
10    else Console.WriteLine($"Incorrect. Too {((Input < value) ? "Low" : "High")}. Try again!");
11    else WriteLine("You guessed it!");
12    else Write("Press any key to exit...");
```

Diagnostic Tools

Diagnostics session: 43 seconds (1 ms selected)

Add counter graphs by checking counters from counter options

Events

Process Memory (MB)

CPU % of all processors

Summary Events Counters Memory Usage

Events

.NET Counters

Memory Usage

Autos

Name	Value	Type
valid	true	bool

Call Stack

Search (Ctrl+E)

Solution Explorer

Search Solution Explorer

Guess A Number

Dependencies

Program.cs

README.md

Hangman

Helicopter

Hurdles

Lights Out

Mancala

Maze

Memory

Minesweeper

Oligopoly

PacMan

Pong

Quick Draw

Reversi

Rock Paper Scissors

Role Playing Game

Roll And More

Rhythm

Shmup

Simon

Summary Events Counters Memory Usage

Events

.NET Counters

Memory Usage

Autos Locals Watch 1

Call Stack Breakpoints Exceptions Commas Immediate Output Solution Explorer Git Changes

Ready

Sports headline New Zealand Cr...

File Edit View Git Project Build Debug Test Analyze Tools Extensions | Search | dotnet_games | C:\Users\Rashid\OneDrive\De | Process: [2428] Guess A Number.exe | Lifecycle Events | Thread: [13504] Main Thread

Guess a number (1-100): 10

Incorrect. Too Low.

Guess a number (1-100): 46

9:29 PM 4/25/2025 ENG IN

The screenshot shows two instances of Microsoft Visual Studio running side-by-side. Both instances are debugging a .NET Core console application named 'Guess A Number'. The top instance shows the application's output window displaying a user interaction where the user has guessed the number 46, which is correct. The bottom instance shows the application's output window displaying a user interaction where the user has guessed the number 10, which is too low. Both instances have the Diagnostic Tools window open, showing memory usage and event logs. The system tray at the bottom indicates the date and time as 4/25/2025.

```

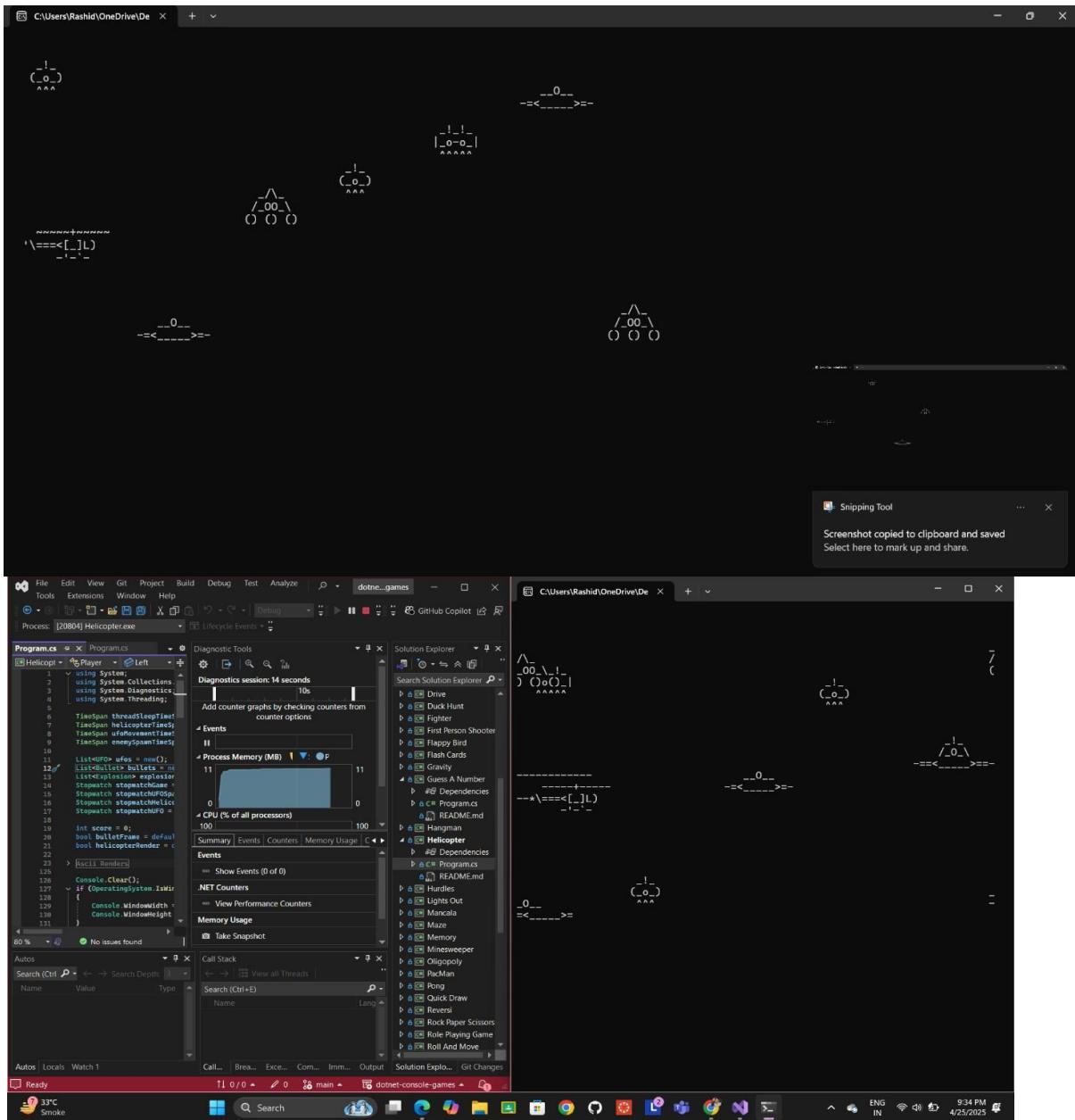
using System;
int value = Random.Shared.Next(1, 101);
while (true)
{
    Console.WriteLine("Guess a number (1-100):");
    bool valid = int.TryParse(Console.ReadLine(), out int input);
    if (!valid) Console.WriteLine("Invalid input");
    else if (input == value) break;
    else Console.WriteLine($"Incorrect.");
}
Console.WriteLine("You guessed it!");
Console.WriteLine("Press any key to exit...");
Console.ReadKey(true);

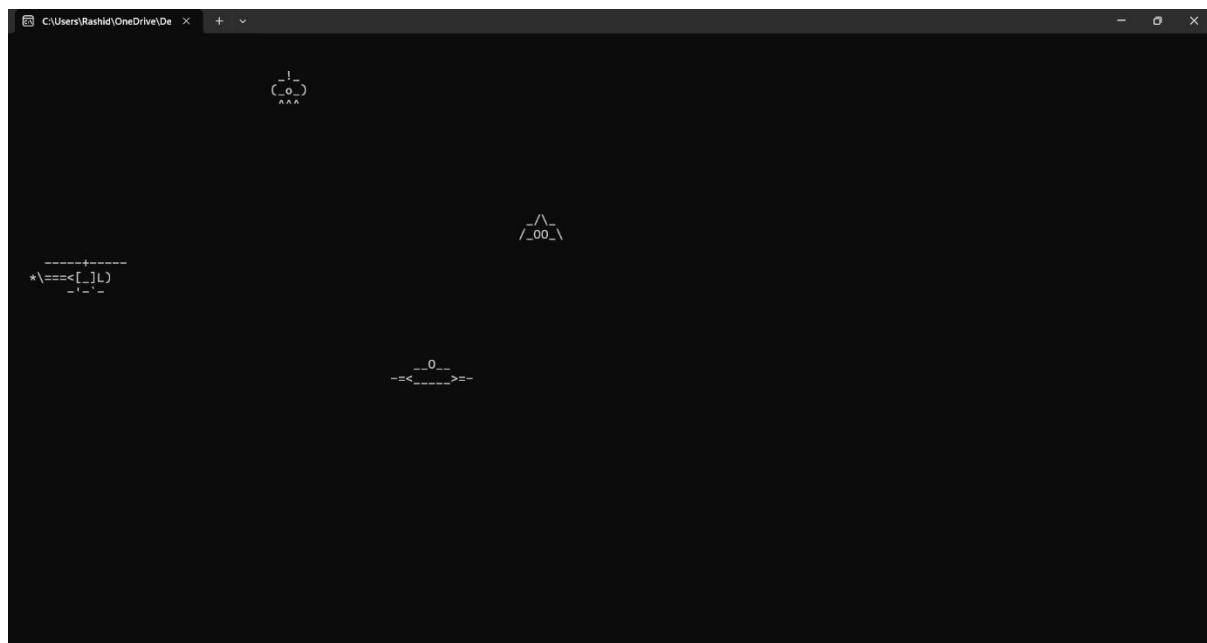
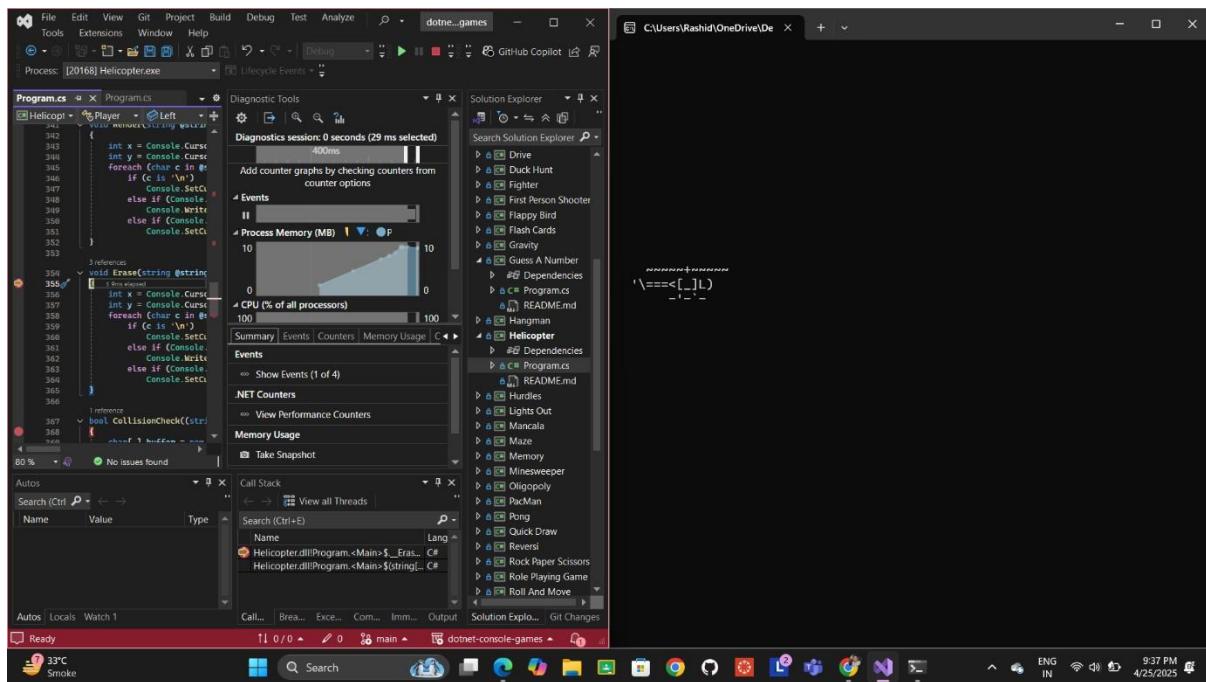
```

3. Game 2: Helicopter

a. Standard Execution

Depicts the unaltered gameplay and normal execution of the *Helicopter* game.





4. Game 3: Gravity

a. Standard Execution

Illustrates the default execution of the game without modifications.

b. Debugging with Breakpoints and Step Controls

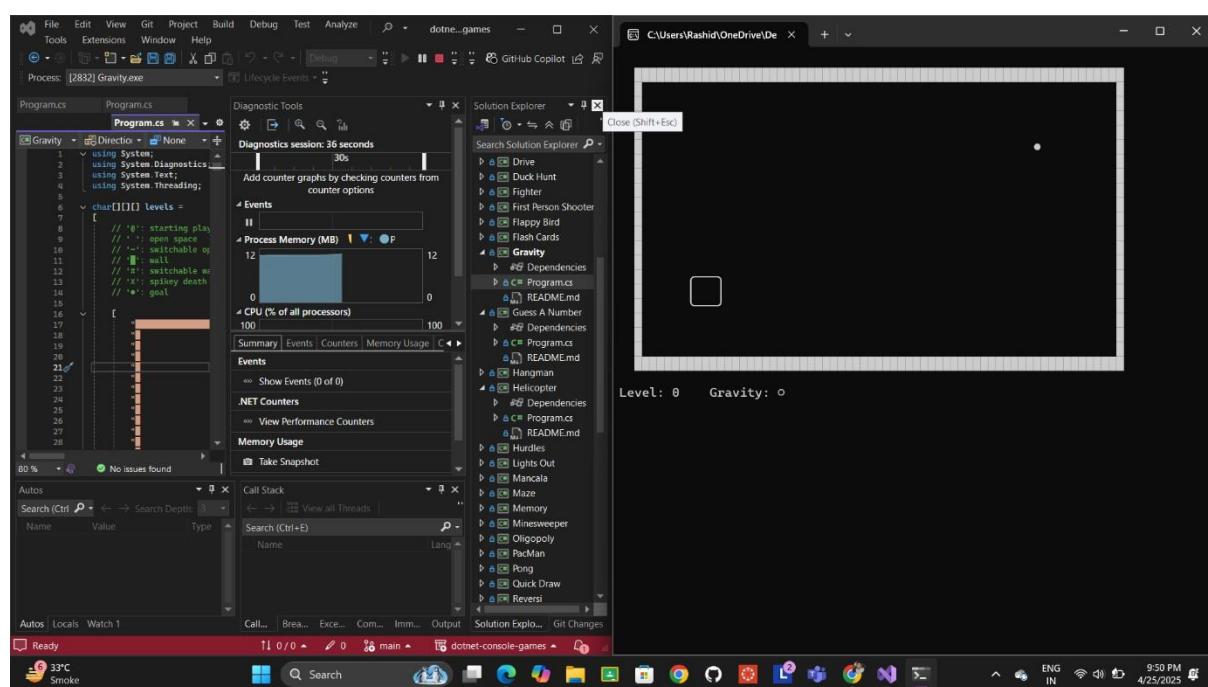
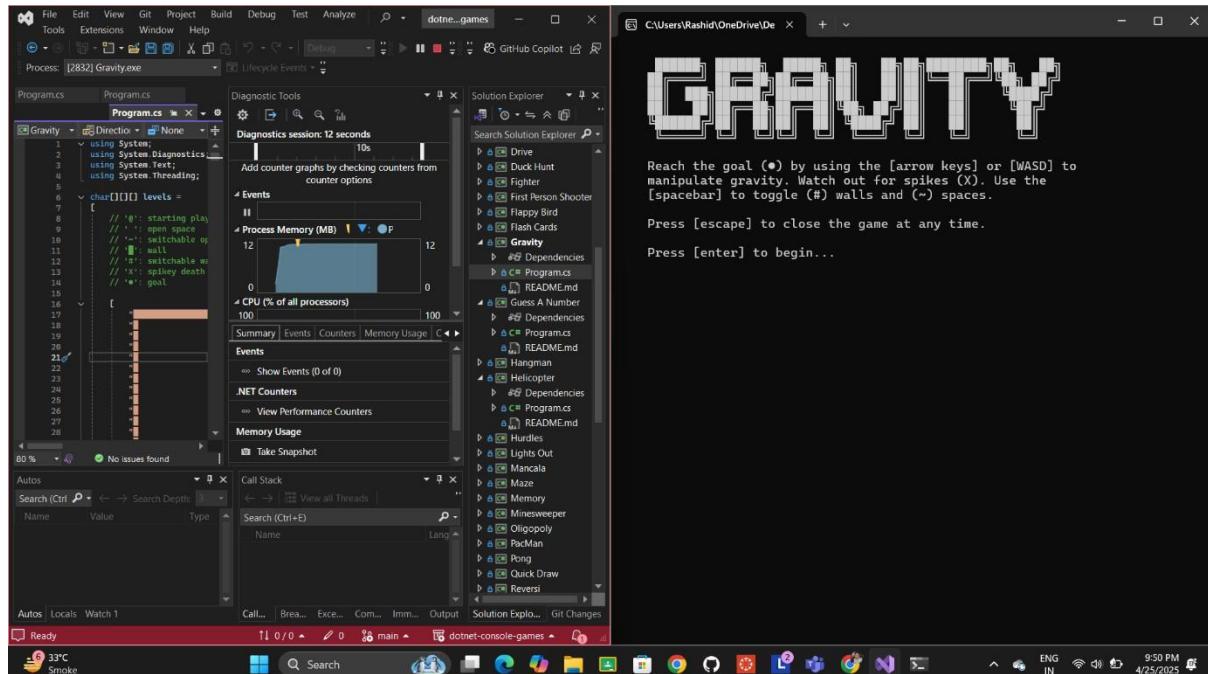
Displays debugging techniques applied to the game's logic using breakpoints and step functions.

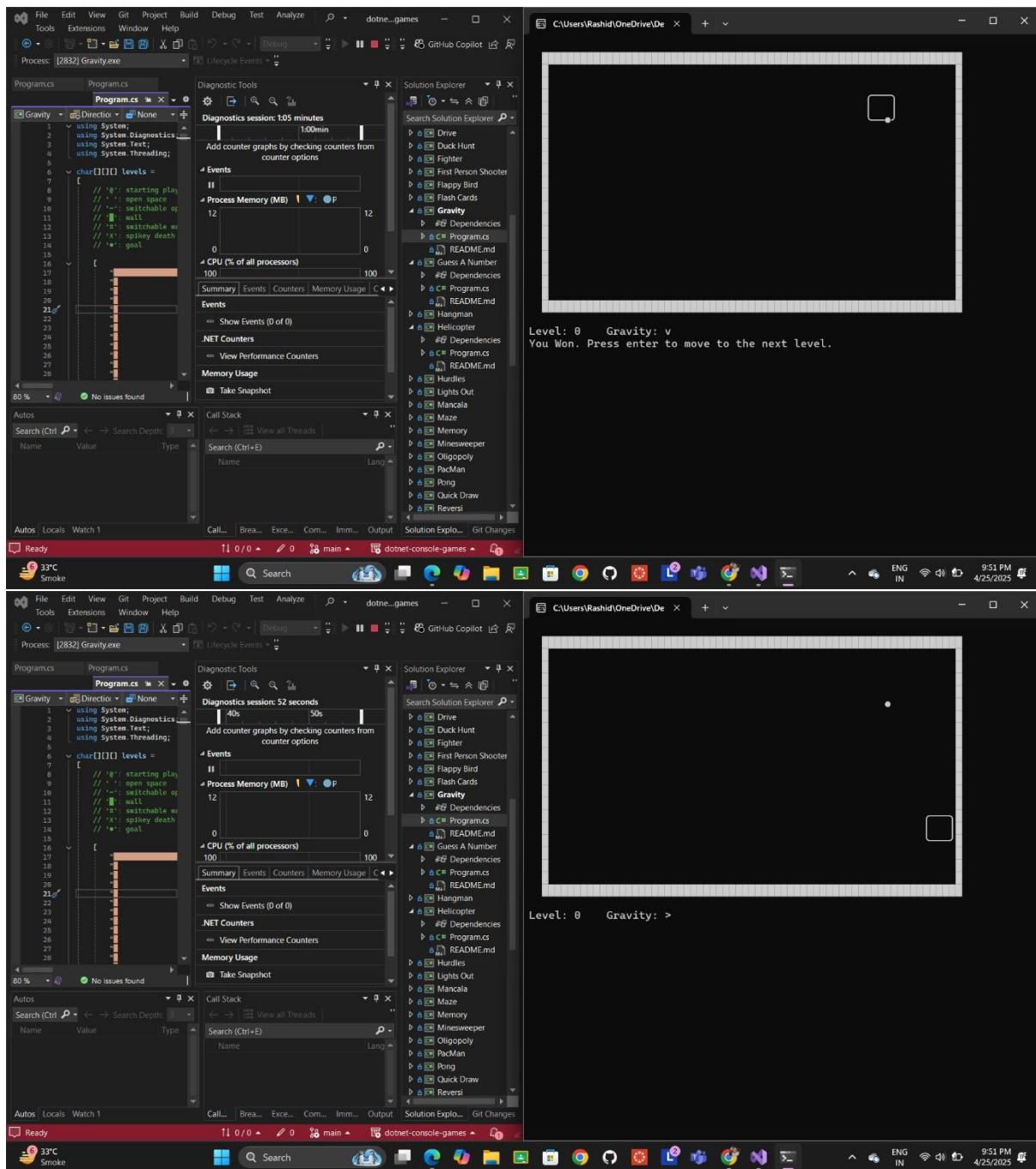
c. Bug Introduction and Execution

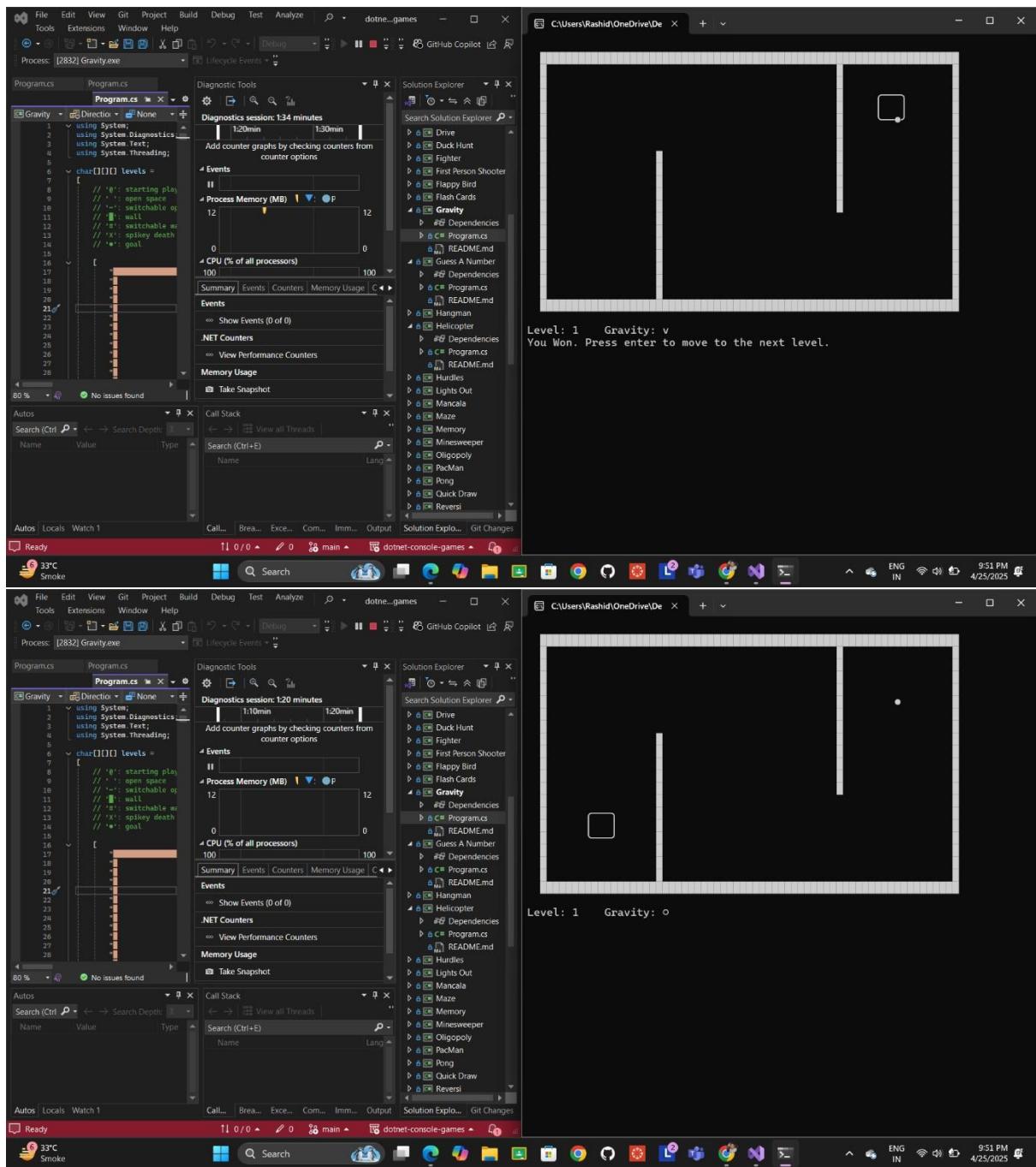
Highlights how intentionally added bugs affect the program's behavior during execution.

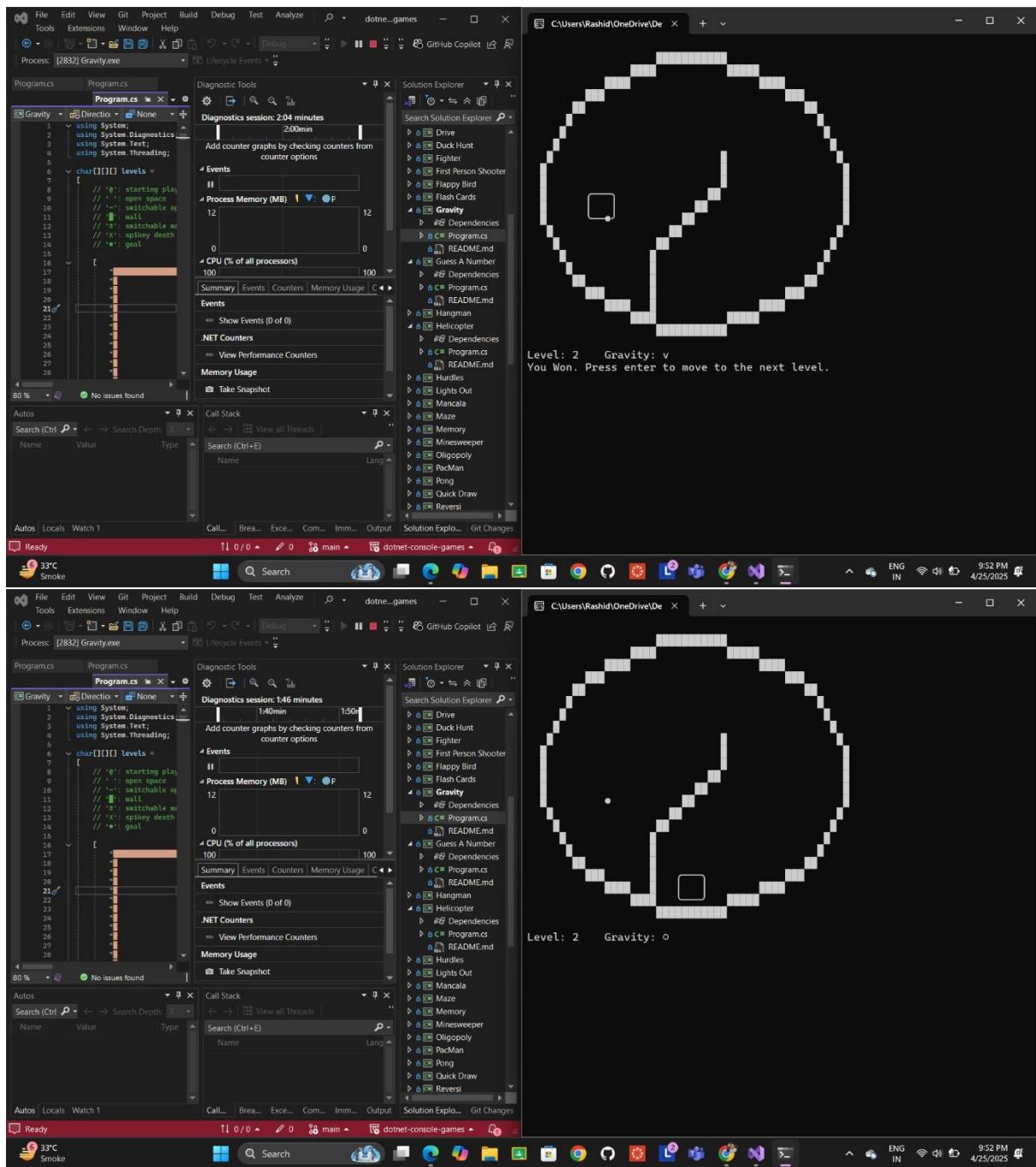
d. Bug Fixing and Re-execution

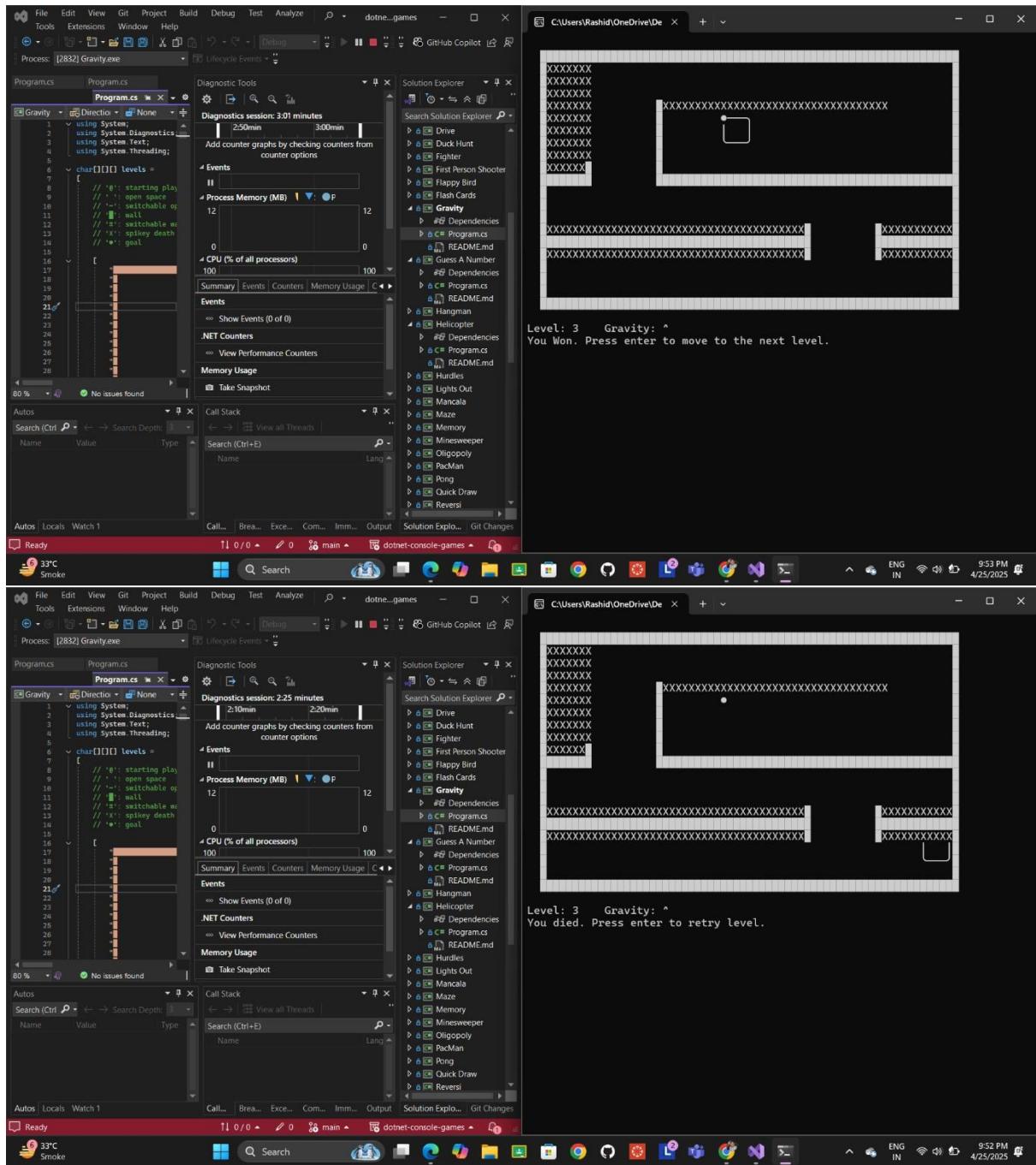
Shows the restoration of the game's correct behavior after resolving the introduced bugs.











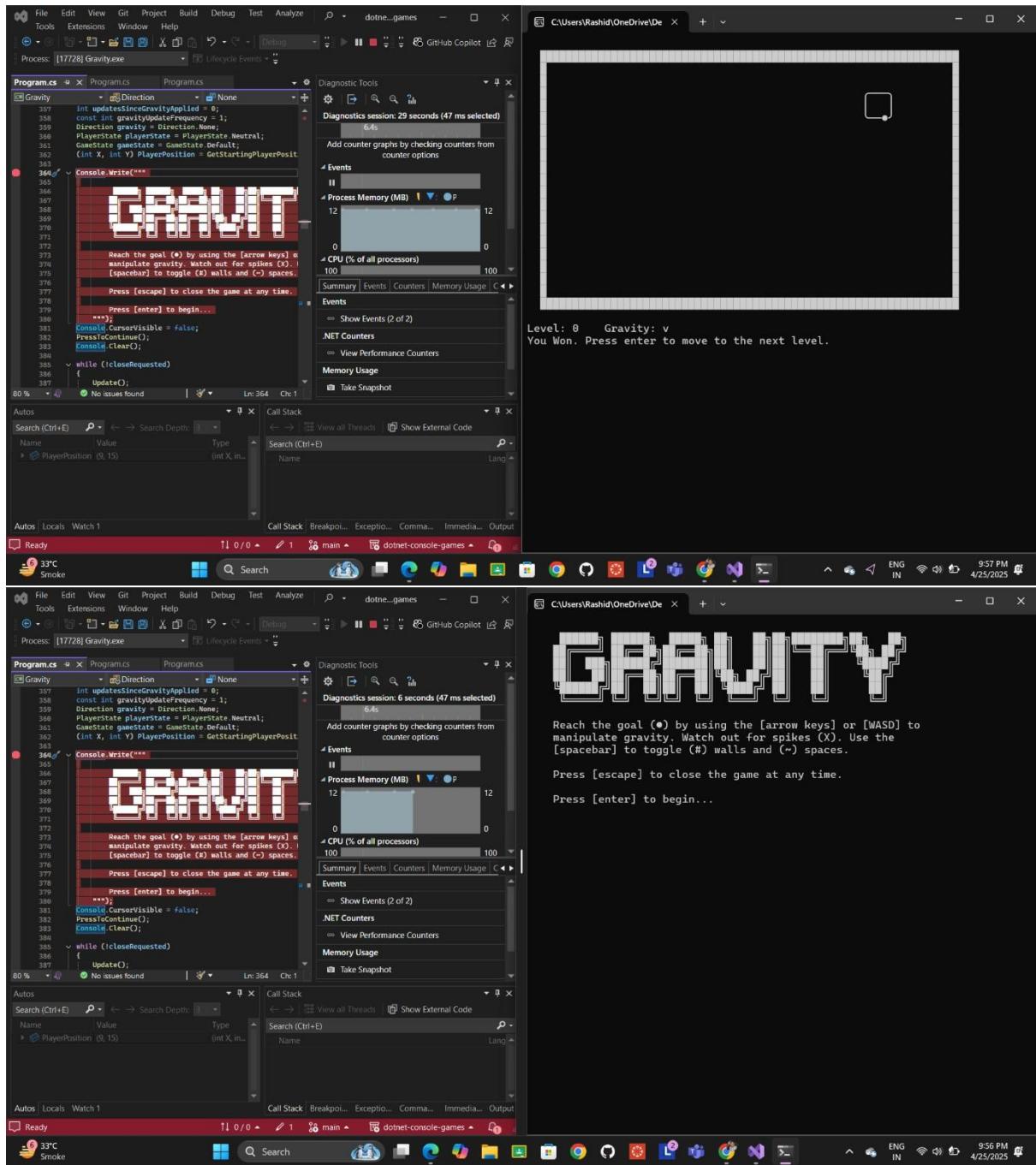
The screenshot displays two side-by-side instances of a game development environment in Visual Studio. Both instances show the same code file, Program.cs, which contains the logic for a game called "GRAVITY".

Top Instance (PlayerPosition at (9, 15)):

- Code:** Shows the main loop where the player's position is updated based on input and gravity.
- Output Window:** Displays the game map titled "GRAVITY" with a player at coordinates (9, 15).
- Diagnostic Tools:** Shows a summary of the diagnostic session, including Process Memory (MB) and CPU (% of all processors) usage over time.
- Watch Window:** Shows the variable "PlayerPosition" with a value of (9, 15).

Bottom Instance (levels is null):

- Code:** Shows the main loop with a check for "levels" being null.
- Output Window:** Displays the game map titled "GRAVITY" with a player at coordinates (9, 15).
- Diagnostic Tools:** Shows a summary of the diagnostic session, including Process Memory (MB) and CPU (% of all processors) usage over time.
- Watch Window:** Shows the variable "levels" with a value of null.



The image displays a dual-monitor setup of a developer's workspace. Both monitors are running instances of Microsoft Visual Studio 2019. On the left monitor, a C# project titled 'Gravity.exe' is open, showing the 'Program.cs' file with code for a gravity-based game. The code includes logic for rendering levels, checking if the player has won, and updating player position. A 'wallUpO' event is triggered when the player reaches the top of a level. On the right monitor, a diagnostic tool is running, specifically the 'Performance' diagnostic session. It shows a graph of CPU usage (% of all processors) over time, with a sharp peak reaching nearly 100% usage. Below the graph, text indicates 'Level: 2 Gravity: v You Won. Press enter to move to the next level.' The taskbar at the bottom of the screen lists several other open applications, including a web browser, file explorers, and system monitoring tools.

The screenshot displays two separate sessions of Microsoft Visual Studio running on a Windows operating system. Both sessions are focused on the same C# project, "dotnet-console-games".

Code Editor:

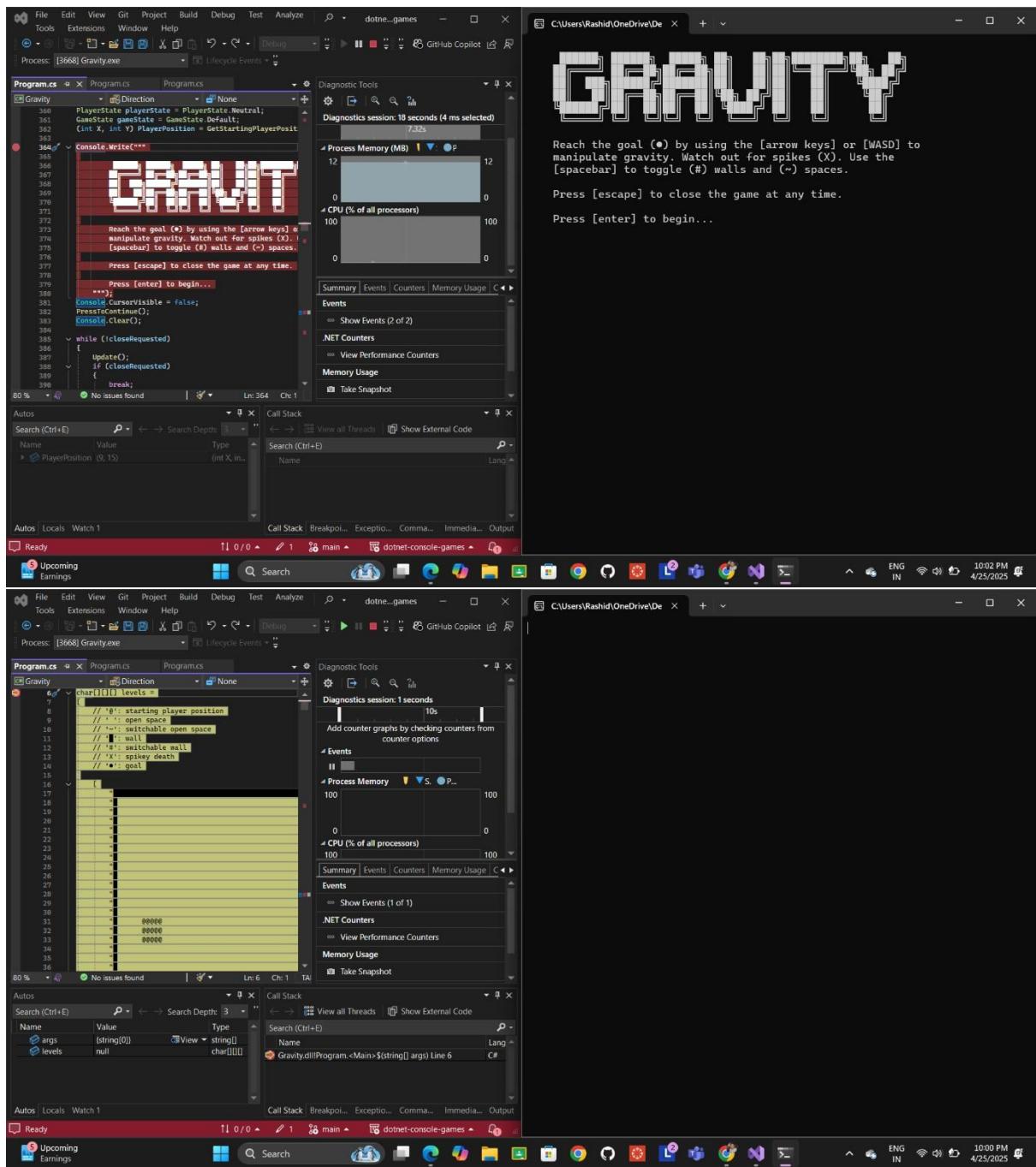
```
    654     else if (gameState.HasFlag(GameState.Won))
    655     {
    656         Render();
    657         if (Level <= levels.Length - 1)
    658         {
    659             Console.WriteLine("You Won. You beat all levels!");
    660             Console.WriteLine("Press enter to exit game");
    661             PressToContinue();
    662             closeRequested = true;
    663             return;
    664         }
    665         Console.WriteLine("You Won. Press enter to move");
    666         PressToContinue();
    667         Console.Clear();
    668         Level++;
    669         if (Level >= levels.Length)
    670         {
    671             PlayerPosition = GetStartingPlayerPositionFromGravity();
    672             gravity = Direction.None;
    673             velocity = (0, 0);
    674             gameState = GameState.Default;
    675         }
    676     }
    677     bool WallUp() =>
    678     levels[Level][PlayerPosition.Y - 2][PlayerPosition.X] == 1
    679     levels[Level][PlayerPosition.Y - 2][PlayerPosition.X] == 2
    680     levels[Level][PlayerPosition.Y - 2][PlayerPosition.X] == 3
    681     levels[Level][PlayerPosition.Y - 2][PlayerPosition.X] == 4
    682     levels[Level][PlayerPosition.Y - 2][PlayerPosition.X] == 5
    683 }
```

Diagnostic Tools:

The Diagnostic Tools windows in both sessions show a "Diagnostics session: 59 seconds (19.013 s selected)" with a timeline from 40s to 49s. It includes sections for Events, Process Memory (MB), CPU (% of all processors), .NET Counters, and Memory Usage.

Taskbar and System Tray:

The taskbar shows several pinned icons, including a browser, file explorer, and development tools. The system tray indicates the date (4/25/2025) and time (9:58 PM). The language setting is ENG IN.



The screenshot displays two separate instances of the Microsoft Visual Studio 2022 IDE running on a Windows operating system. Both instances are focused on the same C# project named 'Gravity'.

Left Instance (Top): This instance shows the 'Program.cs' file containing code related to rendering player states. The code includes logic for appending lines to a render buffer based on player positions and directions. A diagnostic tool window is open, showing a memory usage chart with a single bar at approximately 12 MB and a CPU usage chart with values fluctuating between 0% and 100%.

Right Instance (Bottom): This instance also shows the 'Program.cs' file, but with different code. It contains logic for updating player gravity based on console key inputs. The diagnostic tool window here also shows memory and CPU usage data.

Both instances have their respective 'dotnet-console-games' windows open at the bottom, indicating they are running as console applications. The taskbar at the very bottom of the screen shows various pinned icons for common Windows applications like File Explorer, Task View, and Control Panel, along with the system tray displaying network and battery status.

The image shows a dual-monitor setup with two instances of the Visual Studio IDE running side-by-side. Both instances have the following configuration:

- Code Editor:** Both instances of Visual Studio are displaying the same code file, `Program.cs`, which contains C# code for a game. The code includes methods like `RenderPlayerState()` and `Update()`.
- Diagnostic Tools:** Both instances have the "Diagnostics Tools" window open, showing a timeline from 26.32s to 26.36s. The timeline displays memory usage and CPU usage over time.
- Terminal:** Both instances have a terminal window titled "C:\Users\Rashid\OneDrive\De" open. The terminal displays a black screen with a small white square in the center, likely representing a rendered frame or a placeholder image.

The system tray at the bottom of the screen shows the date and time as 4/25/2025 and 10:02 PM. The taskbar also shows various icons for other applications like File Explorer, Task Manager, and browser tabs.

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help

Program.cs Program.cs Program.cs

Gravity

```
644 if (gameState.HasFlag(GameState.Died))
645 {
646     Render();
647     Console.WriteLine("You died. Press enter to retry level.");
648     PressToContinue();
649     PlayerPosition = GetStartingPlayerPositionFromLevel();
650     gravity = Direction.None;
651     velocity = (0, 0);
652     gameState = GameState.Default;
653 }
654 else if (gameState.HasFlag(GameState.Won))
655 {
656     Render();
657     if (level >= levels.Length - 1)
658     {
659         Console.WriteLine("You Won. You beat all the levels!");
660         Console.WriteLine("Press enter to exit game...");
661         PressToContinue();
662         closeRequested = true;
663         return;
664     }
665     Console.WriteLine("You Won. Press enter to move to the next level.");
666     PressToContinue();
667     Console.Clear();
668     level = level + 2;
669     PlayerPosition = GetStartingPlayerPositionFromLevel();
670     gravity = Direction.None;
671     velocity = (0, 0);
672     gameState = GameState.Default;
673 }
```

80 % No issues found Ln: 673 Ch: 33 Col: 39 TA

Output

```
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Redist\MSVC\14.30.30705\Host\x64\Microsoft.VC143.Debug.dll'
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NET\6.0.1\Microsoft.NETCore.App\6.0.1\mscorlib.dll'
The thread '.NET TP Worker' (596) has exited with code 0 (0x0).
The thread '.NET TP Worker' (13364) has exited with code 0 (0x0).
The thread '.NET TP Worker' (6952) has exited with code 0 (0x0).
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NET\6.0.1\Microsoft.NETCore.App\6.0.1\System.Numerics.Vectors.dll'
The program '[3668] Gravity.exe' has exited with code 4294967295 (0xffffffff).
```

Ready

The screenshot shows a Microsoft Visual Studio interface for a .NET console application named "dotnet-console-games".

Solution Explorer: Lists various projects and files, including GitHub Actions, 2048, Battleship, Beep Pad, Blackjack, Bound, Checkers, Clicker, Connect 4, Console Monsters, Darts, Dice Game, Draw, Drive, and others.

Properties: Shows settings for the selected item.

Output Window: Displays the following log output:

```
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\Microsoft Visual Studio\2022\Community\Preview\MSBuild\Sdk\Microsoft.NET.Sdk.dll'
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NET\6.0.4\Microsoft.NET.dll'
The thread '.NET TP Worker' (596) has exited with code 0 (0x0).
The thread '.NET TP Worker' (13364) has exited with code 0 (0x0).
The thread '.NET TP Worker' (6952) has exited with code 0 (0x0).
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NET\6.0.4\System.Numerics.Vectors.dll'
The program '[3668] Gravity.exe' has exited with code 4294967295 (0xffffffff).
```

Code Editor: The "Program.cs" file is open, showing C# code for a game. A breakpoint is set at line 659. The code handles game state logic, including rendering, player position, gravity, and level progression. It also handles win and death conditions.

The screenshot shows two instances of Visual Studio running side-by-side. The left instance is focused on the 'Program.cs' file, which contains the code for a game titled 'GRAVITY'. The right instance is focused on the 'Diagnostic Tools' window, showing performance metrics over a 2-second session. The Windows taskbar at the bottom is visible, displaying various pinned icons.

dotnet_games

Program.cs

```

    Gravity -> Direction -> None
    630
    631
    632
    633
    634
    635
    636
    637
    638
    639
    640
    641
    642
    643
    644
    645
    646
    647
    648
    649
    650
    651
    652
    653
    654
    655
    656
    657
    658
    659
    660
    661
    662
    663
    664
    665
    666
    667
    668
    669
    670
    671
    672
    673
    674
    675
    676
    677
    678
    679
    680
    681
    682
    683
    684
    685
    686
    687
    688
    689
    690
    691
    692
    693
    694
    695
    696
    697
    698
    699
    700
    701
    702
    703
    704
    705
    706
    707
    708
    709
    710
    711
    712
    713
    714
    715
    716
    717
    718
    719
    720
    721
    722
    723
    724
    725
    726
    727
    728
    729
    730
    731
    732
    733
    734
    735
    736
    737
    738
    739
    740
    741
    742
    743
    744
    745
    746
    747
    748
    749
    750
    751
    752
    753
    754
    755
    756
    757
    758
    759
    760
    761
    762
    763
    764
    765
    766
    767
    768
    769
    770
    771
    772
    773
    774
    775
    776
    777
    778
    779
    780
    781
    782
    783
    784
    785
    786
    787
    788
    789
    790
    791
    792
    793
    794
    795
    796
    797
    798
    799
    800
    801
    802
    803
    804
    805
    806
    807
    808
    809
    810
    811
    812
    813
    814
    815
    816
    817
    818
    819
    820
    821
    822
    823
    824
    825
    826
    827
    828
    829
    830
    831
    832
    833
    834
    835
    836
    837
    838
    839
    840
    841
    842
    843
    844
    845
    846
    847
    848
    849
    850
    851
    852
    853
    854
    855
    856
    857
    858
    859
    860
    861
    862
    863
    864
    865
    866
    867
    868
    869
    870
    871
    872
    873
    874
    875
    876
    877
    878
    879
    880
    881
    882
    883
    884
    885
    886
    887
    888
    889
    890
    891
    892
    893
    894
    895
    896
    897
    898
    899
    900
    901
    902
    903
    904
    905
    906
    907
    908
    909
    910
    911
    912
    913
    914
    915
    916
    917
    918
    919
    920
    921
    922
    923
    924
    925
    926
    927
    928
    929
    930
    931
    932
    933
    934
    935
    936
    937
    938
    939
    940
    941
    942
    943
    944
    945
    946
    947
    948
    949
    950
    951
    952
    953
    954
    955
    956
    957
    958
    959
    960
    961
    962
    963
    964
    965
    966
    967
    968
    969
    970
    971
    972
    973
    974
    975
    976
    977
    978
    979
    980
    981
    982
    983
    984
    985
    986
    987
    988
    989
    990
    991
    992
    993
    994
    995
    996
    997
    998
    999
    1000
    1001
    1002
    1003
    1004
    1005
    1006
    1007
    1008
    1009
    1010
    1011
    1012
    1013
    1014
    1015
    1016
    1017
    1018
    1019
    1020
    1021
    1022
    1023
    1024
    1025
    1026
    1027
    1028
    1029
    1030
    1031
    1032
    1033
    1034
    1035
    1036
    1037
    1038
    1039
    1040
    1041
    1042
    1043
    1044
    1045
    1046
    1047
    1048
    1049
    1050
    1051
    1052
    1053
    1054
    1055
    1056
    1057
    1058
    1059
    1060
    1061
    1062
    1063
    1064
    1065
    1066
    1067
    1068
    1069
    1070
    1071
    1072
    1073
    1074
    1075
    1076
    1077
    1078
    1079
    1080
    1081
    1082
    1083
    1084
    1085
    1086
    1087
    1088
    1089
    1090
    1091
    1092
    1093
    1094
    1095
    1096
    1097
    1098
    1099
    1100
    1101
    1102
    1103
    1104
    1105
    1106
    1107
    1108
    1109
    1110
    1111
    1112
    1113
    1114
    1115
    1116
    1117
    1118
    1119
    1120
    1121
    1122
    1123
    1124
    1125
    1126
    1127
    1128
    1129
    1130
    1131
    1132
    1133
    1134
    1135
    1136
    1137
    1138
    1139
    1140
    1141
    1142
    1143
    1144
    1145
    1146
    1147
    1148
    1149
    1150
    1151
    1152
    1153
    1154
    1155
    1156
    1157
    1158
    1159
    1160
    1161
    1162
    1163
    1164
    1165
    1166
    1167
    1168
    1169
    1170
    1171
    1172
    1173
    1174
    1175
    1176
    1177
    1178
    1179
    1180
    1181
    1182
    1183
    1184
    1185
    1186
    1187
    1188
    1189
    1190
    1191
    1192
    1193
    1194
    1195
    1196
    1197
    1198
    1199
    1200
    1201
    1202
    1203
    1204
    1205
    1206
    1207
    1208
    1209
    1210
    1211
    1212
    1213
    1214
    1215
    1216
    1217
    1218
    1219
    1220
    1221
    1222
    1223
    1224
    1225
    1226
    1227
    1228
    1229
    1230
    1231
    1232
    1233
    1234
    1235
    1236
    1237
    1238
    1239
    1240
    1241
    1242
    1243
    1244
    1245
    1246
    1247
    1248
    1249
    1250
    1251
    1252
    1253
    1254
    1255
    1256
    1257
    1258
    1259
    1260
    1261
    1262
    1263
    1264
    1265
    1266
    1267
    1268
    1269
    1270
    1271
    1272
    1273
    1274
    1275
    1276
    1277
    1278
    1279
    1280
    1281
    1282
    1283
    1284
    1285
    1286
    1287
    1288
    1289
    1290
    1291
    1292
    1293
    1294
    1295
    1296
    1297
    1298
    1299
    1300
    1301
    1302
    1303
    1304
    1305
    1306
    1307
    1308
    1309
    1310
    1311
    1312
    1313
    1314
    1315
    1316
    1317
    1318
    1319
    1320
    1321
    1322
    1323
    1324
    1325
    1326
    1327
    1328
    1329
    1330
    1331
    1332
    1333
    1334
    1335
    1336
    1337
    1338
    1339
    1340
    1341
    1342
    1343
    1344
    1345
    1346
    1347
    1348
    1349
    1350
    1351
    1352
    1353
    1354
    1355
    1356
    1357
    1358
    1359
    1360
    1361
    1362
    1363
    1364
    1365
    1366
    1367
    1368
    1369
    1370
    1371
    1372
    1373
    1374
    1375
    1376
    1377
    1378
    1379
    1380
    1381
    1382
    1383
    1384
    1385
    1386
    1387
    1388
    1389
    1390
    1391
    1392
    1393
    1394
    1395
    1396
    1397
    1398
    1399
    1400
    1401
    1402
    1403
    1404
    1405
    1406
    1407
    1408
    1409
    1410
    1411
    1412
    1413
    1414
    1415
    1416
    1417
    1418
    1419
    1420
    1421
    1422
    1423
    1424
    1425
    1426
    1427
    1428
    1429
    1430
    1431
    1432
    1433
    1434
    1435
    1436
    1437
    1438
    1439
    1440
    1441
    1442
    1443
    1444
    1445
    1446
    1447
    1448
    1449
    1450
    1451
    1452
    1453
    1454
    1455
    1456
    1457
    1458
    1459
    1460
    1461
    1462
    1463
    1464
    1465
    1466
    1467
    1468
    1469
    1470
    1471
    1472
    1473
    1474
    1475
    1476
    1477
    1478
    1479
    1480
    1481
    1482
    1483
    1484
    1485
    1486
    1487
    1488
    1489
    1490
    1491
    1492
    1493
    1494
    1495
    1496
    1497
    1498
    1499
    1500
    1501
    1502
    1503
    1504
    1505
    1506
    1507
    1508
    1509
    1510
    1511
    1512
    1513
    1514
    1515
    1516
    1517
    1518
    1519
    1520
    1521
    1522
    1523
    1524
    1525
    1526
    1527
    1528
    1529
    1530
    1531
    1532
    1533
    1534
    1535
    1536
    1537
    1538
    1539
    1540
    1541
    1542
    1543
    1544
    1545
    1546
    1547
    1548
    1549
    1550
    1551
    1552
    1553
    1554
    1555
    1556
    1557
    1558
    1559
    1560
    1561
    1562
    1563
    1564
    1565
    1566
    1567
    1568
    1569
    1570
    1571
    1572
    1573
    1574
    1575
    1576
    1577
    1578
    1579
    1580
    1581
    1582
    1583
    1584
    1585
    1586
    1587
    1588
    1589
    1590
    1591
    1592
    1593
    1594
    1595
    1596
    1597
    1598
    1599
    1600
    1601
    1602
    1603
    1604
    1605
    1606
    1607
    1608
    1609
    1610
    1611
    1612
    1613
    1614
    1615
    1616
    1617
    1618
    1619
    1620
    1621
    1622
    1623
    1624
    1625
    1626
    1627
    1628
    1629
    1630
    1631
    1632
    1633
    1634
    1635
    1636
    1637
    1638
    1639
    1640
    1641
    1642
    1643
    1644
    1645
    1646
    1647
    1648
    1649
    1650
    1651
    1652
    1653
    1654
    1655
    1656
    1657
    1658
    1659
    1660
    1661
    1662
    1663
    1664
    1665
    1666
    1667
    1668
    1669
    1670
    1671
    1672
    1673
    1674
    1675
    1676
    1677
    1678
    1679
    1680
    1681
    1682
    1683
    1684
    1685
    1686
    1687
    1688
    1689
    1690
    1691
    1692
    1693
    1694
    1695
    1696
    1697
    1698
    1699
    1700
    1701
    1702
    1703
    1704
    1705
    1706
    1707
    1708
    1709
    1710
    1711
    1712
    1713
    1714
    1715
    1716
    1717
    1718
    1719
    1720
    1721
    1722
    1723
    1724
    1725
    1726
    1727
    1728
    1729
    1730
    1731
    1732
    1733
    1734
    1735
    1736
    1737
    1738
    1739
    1740
    1741
    1742
    1743
    1744
    1745
    1746
    1747
    1748
    1749
    1750
    1751
    1752
    1753
    1754
    1755
    1756
    1757
    1758
    1759
    1760
    1761
    1762
    1763
    1764
    1765
    1766
    1767
    1768
    1769
    1770
    1771
    1772
    1773
    1774
    1775
    1776
    1777
    1778
    1779
    1780
    1781
    1782
    1783
    1784
    1785
    1786
    1787
    1788
    1789
    1790
    1791
    1792
    1793
    1794
    1795
    1796
    1797
    1798
    1799
    1800
    1801
    1802
    1803
    1804
    1805
    1806
    1807
    1808
    1809
    1810
    1811
    1812
    1813
    1814
    1815
    1816
    1817
    1818
    1819
    1820
    1821
    1822
    1823
    1824
    1825
    1826
    1827
    1828
    1829
    1830
    1831
    1832
    1833
    1834
    1835
    1836
    1837
    1838
    1839
    1840
    1841
    1842
    1843
    1844
    1845
    1846
    1847
    1848
    1849
    1850
    1851
    1852
    1853
    1854
    1855
    1856
    1857
    1858
    1859
    1860
    1861
    1862
    1863
    1864
    1865
    1866
    1867
    1868
    1869
    1870
    1871
    1872
    1873
    1874
    1875
    1876
    1877
    1878
    1879
    1880
    1881
    1882
    1883
    1884
    1885
    1886
    1887
    1888
    1889
    1890
    1891
    1892
    1893
    1894
    1895
    1896
    1897
    1898
    1899
    1900
    1901
    1902
    1903
    1904
    1905
    1906
    1907
    1908
    1909
    1910
    1911
    1912
    1913
    1914
    1915
    1916
    1917
    1918
    1919
    1920
    1921
    1922
    1923
    1924
    1925
    1926
    1927
    1928
    1929
    1930
    1931
    1932
    1933
    1934
    1935
    1936
    1937
    1938
    1939
    1940
    1941
    1942
    1943
    1944
    1945
    1946
    1947
    1948
    1949
    1950
    1951
    1952
    1953
    1954
    1955
    1956
    1957
    1958
    1959
    1960
    1961
    1962
    1963
    1964
    1965
    1966
    1967
    1968
    1969
    1970
    1971
    1972
    1973
    1974
    1975
    1976
    1977
    1978
    1979
    1980
    1981
    1982
    1983
    1984
    1985
    1986
    1987
    1988
    1989
    1990
    1991
    1992
    1993
    1994
    1995
    1996
    1997
    1998
    1999
    2000
    2001
    2002
    2003
    2004
    2005
    2006
    2007
    2008
    2009
    2010
    2011
    2012
    2013
    2014
    2015
    2016
    2017
    2018
    2019
    2020
    2021
    2022
    2023
    2024
    2025
    2026
    2027
    2028
    2029
    2030
    2031
    2032
    2033
    2034
    2035
    2036
    2037
    2038
    2039
    2040
    2041
    2042
    2043
    2044
    2045
    2046
    2047
    2048
    2049
    2050
    2051
    2052
    2053
    2054
    2055
    2056
    2057
    2058
    2059
    2060
    2061
    2062
    2063
    2064
    2065
    2066
    2067
    2068
    2069
    2070
    2071
    2072
    2073
    2074
    2075
    2076
    2077
    2078
    2079
    2080
    2081
    2082
    2083
    2084
    2085
    2086
    2087
    2088
    2089
    2090
    2091
    2092
    2093
    2094
    2095
    2096
    2097
    2098
    2099
    2100
    2101
    2102
    2103
    2104
    2105
    2106
    2107
    2108
    2109
    2110
    2111
    2112
    2113
    2114
    2115
    2116
    2117
    2118
    2119
    2120
    2121
    2122
    2123
    2124
    2125
    2126
    2127
    2128
    2129
    2130
    2131
    2132
    2133
    2134
    2135
    2136
    2137
    2138
    2139
    2140
    2141
    2142
    2143
    2144
    2145
    2146
    2147
    2148
    2149
    2150
    2151
    2152
    2153
    2154
    2155
    2156
    2157
    2158
    2159
    2160
    2161
    2162
    2163
    2164
    2165
    2166
    2167
    2168
    2169
    2170
    2171
    2172
    2173
    2174
    2175
    2176
    2177
    2178
    2179
    2180
    2181
    2182
    2183
    2184
    2185
    2186
    2187
    2188
    2189
    2190
    2191
    2192
    2193
    2194
    2195
    2196
    2197
    2198
    2199
    2200
    2201
    2202
    2203
    2204
    2205
    2206
    2207
    2208
    2209
    2210
    2211
    2212
    2213
    2214
    2215
    2216
    2217
    2218
    2219
    2220
    2221
    2222
    2223
    2224
    2225
    2226
    2227
    2228
    2229
    2230
    2231
    2232
    2233
    2234
    2235
    2236
    2237
    2238
    2239
    2240
    2241
    2242
    2243
    2244
    2245
    2246
    2247
    2248
    2249
    2250
    2251
    2252
    2253
    2254
    2255
    2256
    2257
    2258
    2259
    2260
    2261
    2262
    2263
    2264
    2265
    2266
    2267
    2268
    2269
    2270
    2271
   
```

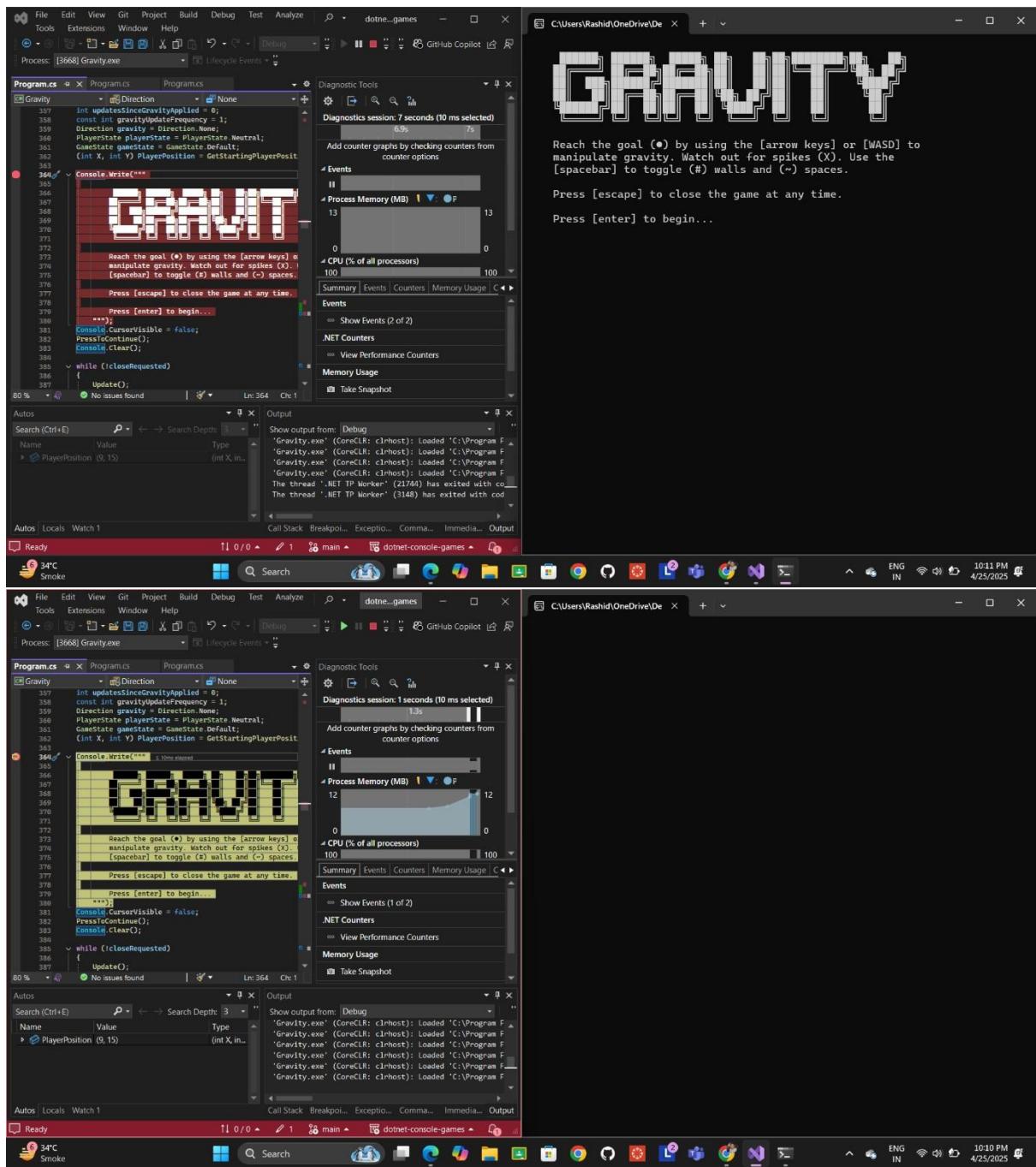
The image shows two side-by-side diagnostic sessions in Visual Studio, each with a code editor, diagnostic tools, and a terminal window.

Top Session (Process ID: 3668 Gravity.exe):

- Code Editor:** Shows `Program.cs` with a character map of a level. Annotations explain symbols: `*` starting player position, `.` open space, `#` switchable open space, `W` switchable wall, `S` spiky death, and `G` goal.
- Diagnostic Tools:** A "Diagnostics session: 1 seconds" window. It shows a timeline from 0 to 10 seconds. Counter graphs for Process Memory (100 MB) and CPU (% of all processors) are displayed. The CPU usage is near zero.
- Output Window:** Shows the command-line output of the application, which loaded several files from the C:\Program Files directory.
- Watch Window:** Shows variables `args` (string[0]) and `levels` (char[10][10]).
- Terminal:** Shows the application's output: "Starting player position", "Switchable open space", "Switchable wall", "Spiky death", and "Goal".

Bottom Session (Process ID: 16072 Gravity.exe):

- Code Editor:** Shows `Program.cs` with a more complex character map and logic for rendering levels and handling game states.
- Diagnostic Tools:** A "Diagnostics session: 14 seconds (18 ms selected)" window. It shows a timeline from 0 to 14 seconds. Counter graphs for Process Memory (13 MB) and CPU (% of all processors) are displayed. The CPU usage is very low.
- Output Window:** Shows the command-line output of the application, which loaded several files from the C:\Program Files directory.
- Watch Window:** Shows variables `GameState.Died`, `GameState.Won`, `d`, `gameState`, `i`, and `f`.
- Terminal:** Shows the application's output: "You died. Press enter to restart.", "Press to continue...", "GetStartingPlayerPosition()", "Gravity = Direction.None", "Velocity = (0, 0)", "Render()", "GameState = GameState.Default", and "Level >= Levels.Length - 1".



Two screenshots of a developer's environment showing the same code editor, diagnostic tools, and terminal window.

Screenshot 1 (Top):

- Code Editor:** Shows `Program.cs` with a break point at line 645. The code handles player movement and rendering. A tooltip for the break point indicates it was triggered by a memory access violation.
- Diagnostic Tools:** A "Diagnostics session: 12 seconds (6 ms selected)" window is open, showing CPU usage (~12%) and Process Memory (~13 MB). It also lists Events (II), .NET Counters (CPU % of all processors ~100%), and Memory Usage (~13 MB).
- Terminal:** Shows the output of the application, which has crashed. It includes error messages about threads exiting with code 0 and the application terminating.

Screenshot 2 (Bottom):

- Code Editor:** Shows the same `Program.cs` code with the break point at line 645.
- Diagnostic Tools:** A "Diagnostics session: 12 seconds (10 ms selected)" window is open, showing CPU usage (~6.9%) and Process Memory (~7 MB). It lists Events (II), .NET Counters (CPU % of all processors ~100%), and Memory Usage (~7 MB).
- Terminal:** Shows the output of the application, which has terminated successfully. It includes messages about threads exiting with code 0 and the application terminating.

The screenshot displays two identical instances of the Visual Studio IDE running on a Windows operating system. Both instances have the following characteristics:

- Code Editor:** Shows the file `Program.cs` with the following code snippet:

```
630     for (int i = -1; i <= 1; i++)
631     {
632         for (int j = -2; j <= 2; j++)
633         {
634             char c = levels[level][i + PlayerPosition];
635             switch (c)
636             {
637                 case 'X': gameState |= GameState.D;
638                 case 'O': gameState |= GameState.W;
639             }
640         }
641     }
642 }
643
644 if (gameState.HasFlag(GameState.Died))
645 {
646     Render();
647     Console.WriteLine("You died. Press enter to retry");
648     PressToContinue();
649     PlayerPosition = GetStartingPlayerPositionFromRow();
650     gravity = Direction.None;
651     velocity = (0, 0);
652     gameState = GameState.Default;
653     //Render();
654     //if (level >= levels.Length - 1)
655     //{
656     //    Console.WriteLine("You Won. You beat all levels!");
657     //    Console.WriteLine("Press enter to exit game");
658     //    PressToContinue();
659     //}
660     closeRequested = true;
661 }
```

- Diagnostics Tools Window:** Shows a timeline from 18.16s to 18.17s with several data series:
 - Events:** Shows two events labeled 'II'.
 - Process Memory (MB):** Shows memory usage fluctuating between 14 and 16 MB.
 - CPU (% of all processors):** Shows CPU usage at 100%.
- Output Window:** Displays the following log output:

```
'Gravity.exe' (CoreCLR: clrhost): loaded 'C:\Program Files\dotnet\shared\Microsoft.NET Core\2.1\ clr.dll'.
'Gravity.exe' (CoreCLR: clrhost): loaded 'C:\Program Files\dotnet\shared\Microsoft.NET Core\2.1\ mscore.dll'.
The thread '.NET TPL Worker' (21744) has exited with code 0.
The thread '.NET TPL Worker' (3148) has exited with code 0.
The thread '.NET TPL Worker' (18860) has exited with code 0.
'Gravity.exe' (CoreCLR: clrhost): loaded 'C:\Program Files\dotnet\shared\Microsoft.NET Core\2.1\ msdia.dll'.
```

- Watch Window:** Shows variable values for `d`, `gameState`, and `r`.
- Taskbar:** Shows pinned icons for File Explorer, Edge, and other development tools, along with the system tray displaying the date (4/25/2025), time (10:12 PM), and battery status.

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help

Program.cs* Program.cs Program.cs

Gravity

```
854     29 references
855     internal enum Direction
856     {
857         None = 0,
858         Up = 1 << 0,
859         Down = 1 << 1,
860         Left = 1 << 2,
861         Right = 1 << 3,
862     }
863
864     9 references
865     internal enum GameState
866     {
867         Default = 0,
868         //Died = 1 << 0,
869         Died = 1 << 1,
870         Won = 1 << 1,
871     }
872     63 references
873     internal enum PlayerState
874     {
875         Neutral = 0,
876         Up = 1 << 0,
877         Down = 1 << 1,
878         Left = 1 << 2,
879         Right = 1 << 3,
880         Sliding = 1 << 4,
881         Squash = 1 << 5,
882     }

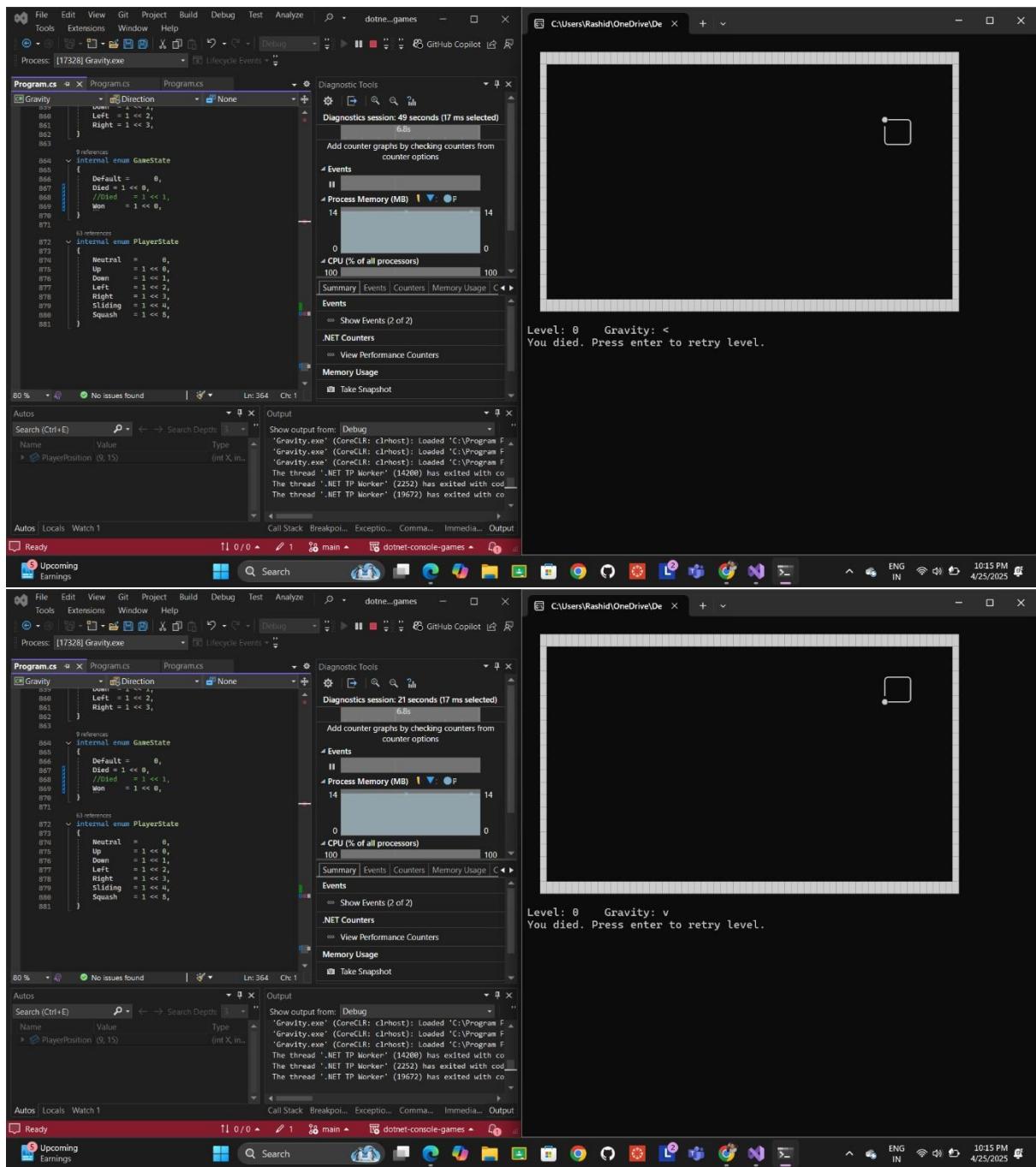
```

80 % No issues found Ln: 867 Ch: 17 Col: 20 TA

Output

```
Show output from: Debug
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\Microsoft Visual
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Mic
The thread '.NET TP Worker' (21744) has exited with code 0 (0x0).
The thread '.NET TP Worker' (3148) has exited with code 0 (0x0).
The thread '.NET TP Worker' (10860) has exited with code 0 (0x0).
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Mic
The program '[3668] Gravity.exe' has exited with code 4294967295 (0xffffffff)
```

A breakpoint... ↑ 0 / 0 ▲ 1 main ▲ dotnet-console-games ▲ 1



Screenshot 1 (Top): Visual Studio IDE showing the first part of the game logic. The code handles player spawning and movement. The diagnostic tools window shows CPU usage and memory usage over a 1-second session.

```

    void SleepAfterRestart()
    {
        TimeSpan sleep = TimeSpan.FromSeconds(1d / 20d);
        if (sleep > TimeSpan.Zero)
        {
            Thread.Sleep(sleep);
        }
        stepmatch.Restart();
    }

    (int X, int Y) GetStartingPlayerPositionFromLevel()
    {
        for (int i = 0; i < levels[level].Length; i++)
        {
            for (int j = 0; j < levels[level][i].Length; j++)
            {
                if (levels[level][i][j] is 'e')
                {
                    return (j + 1, i + 2);
                }
            }
        }
        throw new Exception($"Level {level} has no starting point");
    }

    void Update()
    {
        reference...
    }

```

Screenshot 2 (Bottom): Visual Studio IDE showing the rendering logic. The diagnostic tools window shows CPU usage and memory usage over a 2-second session.

```

    StringBuilder render = new();
    render.AppendLine();
    for (int i = 0; i < levels[level].Length; i++)
    {
        render.Append(' ');
        render.Append(' ');
        for (int j = 0; j < levels[level][i].Length; j++)
        {
            char c = levels[level][i][j];
            if (c is ' ')
            {
                c = ' ';
            }
            else if (c is 'X' and not ' ' * 64)
            {
                PlayerPosition.X = 1 << j * 64;
                PlayerPosition.Y = 1 << i * 64;
                PlayerPosition.Y = 1 >> 1;
            }
            c = playerSprite[i - PlayerPosition.Y];
            render.Append(c);
        }
        render.AppendLine();
    }
    render.AppendLine();
    render.AppendLine($"Level: {level} Gravity: {Gravity}");
    Console.SetCursorPosition(0, 0);
    Console.WriteLine(render);
    Console.CursorVisible = false;
}

```

The screenshot displays two instances of the Visual Studio IDE running on a Windows desktop. Both instances are working on a project named 'Gravity'.

Top Instance:

- Code Editor:** Shows `Program.cs` with logic for handling player movement and gravity manipulation.
- Output Window:** Shows the game's console output, including the title 'GRAVITY' and instructions for using arrow keys and spacebar.
- Diagnostic Tools:** A 'Diagnostics session' is active, showing memory usage (Process Memory MB) and CPU usage (% of all processors).

Bottom Instance:

- Code Editor:** Shows `Program.cs` with code for sleeping after a reference, getting starting player positions, and updating levels.
- Output Window:** Shows the game's console output, including the title 'GRAVITY' and instructions for using arrow keys and spacebar.
- Diagnostic Tools:** A 'Diagnostics session' is active, showing memory usage (Process Memory MB) and CPU usage (% of all processors).

The desktop taskbar at the bottom shows various pinned icons, and the system tray indicates the date as 4/25/2025 and the time as 10:32 PM.

The screenshot displays a Windows desktop environment with two instances of Visual Studio running the game 'Gravity.exe'.
Left Instance (Foreground):
- **Code Editor**: Shows the C# source code for the 'Program.cs' file. The code handles rendering levels, player movement, and gravity logic.

```
718     string[] play = died;
719     StringBuilder As = new StringBuilder();
720     render.AppendLine();
721     for (int i = 0; i < levels[level].Length; i++)
722     {
723         render.Append(" ");
724         render.Append(" ");
725         for (int j = 0; j < levels[level][i].Length; j++)
726         {
727             char c = levels[level][i][j];
728             if (c == 'X')
729             {
730                 c = ' ';
731             }
732             if (c != 'X' and not ' ')
733             {
734                 PlayerPosition.X -> j &&
735                 PlayerPosition.X + 2 > j &&
736                 PlayerPosition.Y <= 5 &&
737                 PlayerPosition.Y + 1 >= j
738             }
739             c = playerSprites[i - PlayerPosition.Y];
740             render.Append(c);
741         }
742         render.AppendLine();
743     }
744     render.AppendLine();
745     render.AppendLine($"Level: {level} Gravity: {Gravity:R}");
746     Console.SetCursorPosition(0, 0);
747     Console.WriteLine(render);
748 }
```


- **Diagnostic Tools**: A 'Diagnostics session' is active with 51 seconds selected, showing CPU usage and memory usage graphs.
- **Output Window**: Shows the game's startup logs and thread information.
- **Status Bar**: Shows the current line (Ln 734) and character (Ch 4).
- **Toolbars**: Includes 'Search (Ctrl+E)', 'Call Stack', 'Breakpoints', 'Exceptions', 'Comma...', 'Immediate...', and 'Output'.
- **Autos, Locals, Watch 1**: Standard debugger windows.
- **Taskbar**: Shows the game window titled 'dotnet-console-games' and other system icons.
- **System Tray**: Shows network status, battery level, and date/time (4/25/2025, 10:38 PM).
Right Instance (Background):
- **Game Window**: Displays the word 'GRAVITY' in large, blocky letters made of spikes. Below it, instructions are displayed:

```
Reach the goal (•) by using the [arrow keys] or [WASD] to manipulate gravity. Watch out for spikes (X). Use the [spacebar] to toggle (#) walls and (>) spaces.
```

Press [escape] to close the game at any time.
Press [enter] to begin...

This layout illustrates the development and testing process for a console game like Gravity.

The screenshot displays two separate instances of a game application running in Visual Studio. Both instances show the same code in `Program.cs` and a shared `Diagnostic Tools` window.

Code in Program.cs:

```
    if (Thread.State == ThreadState.Died)
    {
        AsmDisplay("Current document");
        stepwatch.Restart();
    }

    static int X, int Y) GetStartingPlayerPositionFromLevel()
    {
        for (int i = 0; i < levels[level].Length; i++)
        {
            for (int j = 0; j < levels[level][i].Length; j++)
            {
                if ((levels[level][i][j] is 'W'))
                {
                    return (i + 1, i + 2);
                }
            }
        }
        throw new Exception($"Level {level} has no starting position");
    }

    void Update()
    {
        while (Console.KeyAvailable)
        {
            switch (Console.ReadKey(true).Key)
            {
                case ConsoleKey.W or ConsoleKey.UpArrow:
                    if (gravity is not Direction.Up)
                        gravity = Direction.Up;
                    break;
            }
        }
    }

    void Render()
    {
        if (levels.Length - 1)
        {
            Console.WriteLine("You Won. You beat all levels. Press enter to exit game.");
            Console.ReadLine();
            closeRequested = true;
            return;
        }

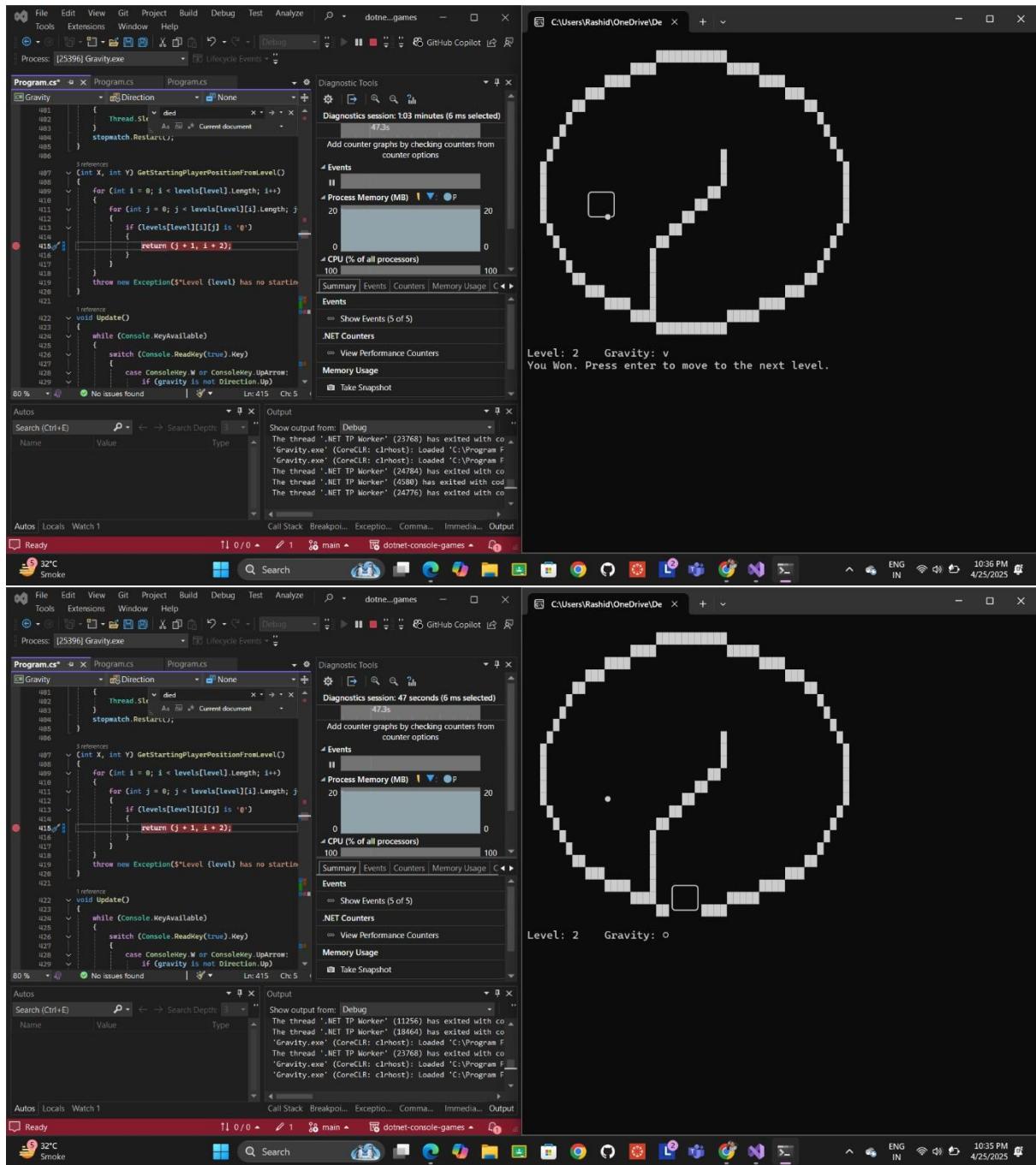
        Console.WriteLine("You Won. Press enter to move on.");
        Console.Clear();
        ConsoleColor previousColor = Console.ForegroundColor;
        PlayerPosition = GetStartingPlayerPositionFromLevel();
        gravity = Direction.None;
        velocity = (0, 0);
        gameState = GameState.Default;
    }

    bool WallingQ ==
    levels[level][PlayerPosition.Y - 2][PlayerPosition.X]
    levels[level][PlayerPosition.Y - 2][PlayerPosition.X + 1]
    levels[level][PlayerPosition.Y - 2][PlayerPosition.X - 1]
    levels[level][PlayerPosition.Y - 2][PlayerPosition.X + 2]
    levels[level][PlayerPosition.Y - 2][PlayerPosition.X - 2]
```

Diagnostic Tools Window:

- Diagnostics session:** 41 seconds (6 ms selected)
- Events:** 11
- Process Memory (MB):** 18
- CPU (% of all processors):** 100

The diagnostic tools also provide options to add counter graphs, view .NET counters, and take a snapshot.



The screenshot displays two separate sessions of the Visual Studio IDE running on a Windows operating system. Both sessions are focused on the same C# project, "dotne_games".

Left Session:

- Code Editor:** Shows the "Program.cs" file with code related to gravity and player movement. A break point is set at line 415.
- Diagnostics Tools:** A "Diagnostics session" has been running for 1:09 minutes. The "Process Memory (MB)" graph shows memory usage fluctuating between 0 and 20 MB. Other tabs in the Diagnostic Tools include "Events", ".NET Counters", and "Memory Usage".
- Output Window:** Displays the command-line output of the application's execution.
- Taskbar:** Shows the Windows Start button, a weather icon (32°C), a "Smoke" icon, and several pinned application icons including File Explorer, Edge, and GitHub Copilot.

Right Session:

- Code Editor:** Shows the same "Program.cs" file with similar code and a break point at line 415.
- Diagnostics Tools:** A "Diagnostics session" has been running for 1:09 minutes. The "Process Memory (MB)" graph shows memory usage fluctuating between 0 and 20 MB. Other tabs in the Diagnostic Tools include "Events", ".NET Counters", and "Memory Usage".
- Output Window:** Displays the command-line output of the application's execution.
- Taskbar:** Shows the Windows Start button, a weather icon (32°C), a "Smoke" icon, and several pinned application icons including File Explorer, Edge, and GitHub Copilot.

The image displays a dual-monitor setup on a Windows operating system. Both monitors show identical configurations of the Visual Studio IDE and its diagnostic tools.

Visual Studio Configuration:

- Code Editor:** Shows `Program.cs` with the following code (lines 401-421):

```
    {
        Thread.Sleep(100);
        stopwatch.Restart();
    }

    static (int X, int Y) GetStartingPlayerPositionFromLevel()
    {
        for (int i = 0; i < levels[level].Length; i++)
        {
            for (int j = 0; j < levels[level][i].Length; j++)
            {
                if ((levels[level][i][j]) is 'P')
                    return (j + 1, i + 2);
            }
        }
        throw new Exception($"Level {level} has no starting position");
    }

    void Update()
    {
        while (Console.KeyAvailable)
        {
            switch (Console.ReadKey(true).Key)
            {
                case ConsoleKey.W or ConsoleKey.UpArrow:
                    if (gravity is not Direction.Up)
```

- Output Window:** Displays the command-line output of the application, showing the loading of the program and the exit of several .NET threads.
- Diagnostic Tools:** Shows performance metrics for a diagnostic session. The session duration is 19 seconds (1 ms selected). The CPU usage is at 100% for all processors, and the process memory usage is at 14 MB.

System Status:

- Bottom Taskbar: Shows the date (4/25/2025), time (10:38 PM), battery level (ENG IN), and other system icons.

The screenshot displays two identical instances of the Visual Studio IDE running on a Windows operating system. Both instances have the project 'dotnet-console-games' open, specifically the file 'Program.cs'. The code in 'Program.cs' is a C# program for a game titled 'Gravity'. It includes logic for rendering a 3x3 grid of characters ('X' and 'O'), handling player movement (direction), and checking for game states like 'Won' or 'Died'. The diagnostic tools windows are open in both instances, showing memory usage and CPU performance over time. The taskbar at the bottom shows various pinned icons and the system clock.

The screenshot displays two separate sessions of a development environment, likely Visual Studio, running on a Windows operating system. Both sessions are focused on a project named 'dotnet_games'.

Top Session (Level 6):

- Code Editor:** Shows `Program.cs` with logic for a gravity simulation game. It includes a nested loop for rows and columns, character placement, and state switching based on character ('X') or dot ('.') detection.
- Diagnostic Tools:** A 'Diagnostics' session is active for 2:50 minutes. The 'Process Memory (MB)' graph shows usage fluctuating between 17 and 18 MB. The 'CPU (%) of all processors' graph shows usage around 100%.
- Output Window:** Displays standard .NET runtime logs and a message from the game: "Gravity.exe" has exited with code 0.
- Taskbar:** Shows the date and time as 4/25/2025 10:42 PM.

Bottom Session (Level 4):

- Code Editor:** Shows the same `Program.cs` file with identical logic.
- Diagnostic Tools:** A 'Diagnostics' session is active for 2:38 minutes. The 'Process Memory (MB)' graph shows usage fluctuating between 16 and 17 MB. The 'CPU (%) of all processors' graph shows usage around 100%.
- Output Window:** Displays game messages: "You Won. Press enter to move to the next level." and "Gravity.exe" has exited with code 0.
- Taskbar:** Shows the date and time as 4/25/2025 10:40 PM.

The screenshot displays two instances of Visual Studio 2022 running on a Windows 10 desktop. Both instances are working on the same project, "Gravity", which contains a single file, Program.cs.

Left Instance (Foreground):

- Code Editor:** Shows the implementation of a gravity game. It includes logic for rendering levels, handling player movement (using a gravity variable), and checking for level completion. A specific section of code handles wall collision detection by comparing player position against level boundaries.
- Diagnostic Tools:** A "Diagnostics session" is active, showing a timeline from 3:05 minutes to 3:06 minutes. The "Process Memory (MB)" graph shows memory usage fluctuating between 17 and 20 MB. The "CPU (% of all processors)" graph shows CPU usage at 0%.
- Output Window:** Displays debug output, including thread exits and file loads.
- Taskbar:** Shows the desktop environment with icons for File Explorer, Task View, and other standard Windows apps.

Right Instance (Background):

- Code Editor:** Shows the same code as the left instance.
- Diagnostic Tools:** A "Diagnostics session" is active, showing a timeline from 2:56:57:33 min to 2:56:57:34 min. The "Process Memory (MB)" graph shows memory usage fluctuating between 17 and 20 MB. The "CPU (% of all processors)" graph shows CPU usage at 0%.
- Output Window:** Displays debug output, including thread exits and file loads.
- Taskbar:** Shows the desktop environment with icons for File Explorer, Task View, and other standard Windows apps.

The screenshot displays two instances of a game titled "Gravity" running in Visual Studio. Both instances show a terminal window with the output of the game's logic.

Top Session (Win State):

```
Level: 8 Gravity: >
You Won. Press enter to move to the next level.
```

Bottom Session (Crash State):

```
Level: 8 Gravity: o
Exception thrown at 419: System.Exception: Level {level} has no starting point
```

The code in both sessions is identical, showing the logic for determining the starting position of a player based on the level and gravity direction.

```
if (gravity == Direction.Up)
{
    if (levels[level][0] != 'o')
        return (j + 0, i + 0);
}
else
{
    if (levels[level][0] != 'x')
        return (j + 0, i + 0);
}
```

Screenshot of Visual Studio showing the development environment for a game project named "dotnet_games".

Top Window (Process ID 24884):

- Code Editor:** Shows `Program.cs` with code related to a game level. It includes a character map and a levels array.
- Output Window:** Shows the command-line output of the application, indicating it has loaded several times.
- Diagnostic Tools:** A performance monitor showing CPU usage (~10%) and memory usage (~17 MB) over a 4-second session.

Bottom Window (Process ID 16364):

- Code Editor:** Shows the same `Program.cs` file with more code, including logic for getting starting player positions and handling console input.
- Output Window:** Shows the application's exit message, indicating it has exited with code 0.
- Diagnostic Tools:** A performance monitor showing CPU usage (~10%) and memory usage (~17 MB) over a 4:17-minute session.

The taskbar at the bottom shows various open applications, including a weather widget (32°C), a system status bar, and the system tray.

The screenshot displays two instances of the Visual Studio IDE running on a Windows 10 desktop. Both instances are working on a project named 'dotnet_games'. The top instance shows a diagnostic session with a 'Process Memory (MB)' graph and a 'CPU (% of all processors)' graph. The bottom instance also shows a diagnostic session with similar monitoring tools. Both instances have the 'Program.cs' file open, which contains C# code for a game. The code includes logic for player movement, gravity, and level loading. The Windows taskbar at the bottom shows various pinned icons and the system tray indicating the date and time.

```
360     PlayerState playerstate = PlayerState.Neutral;
361     GameState gameState = GameState.BeforeStart;
362     int X, int Y PlayerPosition = GetStartingPlayerPosition();
363
364     Console.WriteLine("GRAVITY");
365
366     Readme the goal (X) by entering the arrow keys to
367     manipulate gravity. Watch out for spikes (Y)
368     (spacebar) to toggle (#) walls and (-) spaces.
369
370     Press [escape] to close the game at any time.
371
372     Press [enter] to begin...
373
374     ***;
375     Console.CursorVisible = false;
376     pressToContinue();
377     Console.Clear();
378
379     while (!closeRequested)
380     {
381         Update();
382         if (closeRequested)
383         {
384             break;
385         }
386     }
387
388     Console.WriteLine("PlayerPosition ({0}, {1})",
389     PlayerPosition.X, PlayerPosition.Y);
390
391     Console.ReadLine();
392
393     void Update()
394     {
395         Thread.Sleep(10);
396         stopwatch.Restart();
397
398         Int X, Int Y GetStartingPlayerPositionFromLevel();
399
400         for (Int i = 0; i < levels[level].Length; i++)
401         {
402             for (Int j = 0; j < levels[level][i].Length; j++)
403             {
404                 If (levels[level][i][j] is 'g')
405                 {
406                     return (j + 0, i + 0);
407                 }
408             }
409         }
410         throw new Exception("Level {0} has no starting point");
411
412         Int update()
413         {
414             While (Console.KeyAvailable)
415             {
416                 switch (Console.ReadKey(true).Key)
417                 {
418                     Case ConsoleKey.W or ConsoleKey.UpArrow:
419                         If (gravity is not Direction.Up)
420                             gravity = Direction.Up;
421
422                     Case ConsoleKey.S or ConsoleKey.DownArrow:
423                         If (gravity is not Direction.Down)
424                             gravity = Direction.Down;
425
426                     Case ConsoleKey.A or ConsoleKey.LeftArrow:
427                         If (gravity is not Direction.Right)
428                             gravity = Direction.Right;
429
430                     Case ConsoleKey.D or ConsoleKey.RightArrow:
431                         If (gravity is not Direction.Left)
432                             gravity = Direction.Left;
433
434                     Case ConsoleKey.Spacebar:
435                         If (walls is true)
436                             walls = false;
437                         Else
438                             walls = true;
439
440                     Case ConsoleKey.Escape:
441                         Close();
442
443                     Default:
444                         Break();
445
446                 }
447             }
448
449             If (walls is true)
450                 walls = false;
451             Else
452                 walls = true;
453
454             If (spikes is true)
455                 spikes = false;
456             Else
457                 spikes = true;
458
459             If (spaces is true)
460                 spaces = false;
461             Else
462                 spaces = true;
463
464             If (walls is true)
465                 walls = false;
466             Else
467                 walls = true;
468
469             If (spikes is true)
470                 spikes = false;
471             Else
472                 spikes = true;
473
474             If (spaces is true)
475                 spaces = false;
476             Else
477                 spaces = true;
478
479             If (walls is true)
480                 walls = false;
481             Else
482                 walls = true;
483
484             If (spikes is true)
485                 spikes = false;
486             Else
487                 spikes = true;
488
489             If (spaces is true)
490                 spaces = false;
491             Else
492                 spaces = true;
493
494             If (walls is true)
495                 walls = false;
496             Else
497                 walls = true;
498
499             If (spikes is true)
500                 spikes = false;
501             Else
502                 spikes = true;
503
504             If (spaces is true)
505                 spaces = false;
506             Else
507                 spaces = true;
508
509             If (walls is true)
510                 walls = false;
511             Else
512                 walls = true;
513
514             If (spikes is true)
515                 spikes = false;
516             Else
517                 spikes = true;
518
519             If (spaces is true)
520                 spaces = false;
521             Else
522                 spaces = true;
523
524             If (walls is true)
525                 walls = false;
526             Else
527                 walls = true;
528
529             If (spikes is true)
530                 spikes = false;
531             Else
532                 spikes = true;
533
534             If (spaces is true)
535                 spaces = false;
536             Else
537                 spaces = true;
538
539             If (walls is true)
540                 walls = false;
541             Else
542                 walls = true;
543
544             If (spikes is true)
545                 spikes = false;
546             Else
547                 spikes = true;
548
549             If (spaces is true)
550                 spaces = false;
551             Else
552                 spaces = true;
553
554             If (walls is true)
555                 walls = false;
556             Else
557                 walls = true;
558
559             If (spikes is true)
560                 spikes = false;
561             Else
562                 spikes = true;
563
564             If (spaces is true)
565                 spaces = false;
566             Else
567                 spaces = true;
568
569             If (walls is true)
570                 walls = false;
571             Else
572                 walls = true;
573
574             If (spikes is true)
575                 spikes = false;
576             Else
577                 spikes = true;
578
579             If (spaces is true)
580                 spaces = false;
581             Else
582                 spaces = true;
583
584             If (walls is true)
585                 walls = false;
586             Else
587                 walls = true;
588
589             If (spikes is true)
590                 spikes = false;
591             Else
592                 spikes = true;
593
594             If (spaces is true)
595                 spaces = false;
596             Else
597                 spaces = true;
598
599             If (walls is true)
600                 walls = false;
601             Else
602                 walls = true;
603
604             If (spikes is true)
605                 spikes = false;
606             Else
607                 spikes = true;
608
609             If (spaces is true)
610                 spaces = false;
611             Else
612                 spaces = true;
613
614             If (walls is true)
615                 walls = false;
616             Else
617                 walls = true;
618
619             If (spikes is true)
620                 spikes = false;
621             Else
622                 spikes = true;
623
624             If (spaces is true)
625                 spaces = false;
626             Else
627                 spaces = true;
628
629             If (walls is true)
630                 walls = false;
631             Else
632                 walls = true;
633
634             If (spikes is true)
635                 spikes = false;
636             Else
637                 spikes = true;
638
639             If (spaces is true)
640                 spaces = false;
641             Else
642                 spaces = true;
643
644             If (walls is true)
645                 walls = false;
646             Else
647                 walls = true;
648
649             If (spikes is true)
650                 spikes = false;
651             Else
652                 spikes = true;
653
654             If (spaces is true)
655                 spaces = false;
656             Else
657                 spaces = true;
658
659             If (walls is true)
660                 walls = false;
661             Else
662                 walls = true;
663
664             If (spikes is true)
665                 spikes = false;
666             Else
667                 spikes = true;
668
669             If (spaces is true)
670                 spaces = false;
671             Else
672                 spaces = true;
673
674             If (walls is true)
675                 walls = false;
676             Else
677                 walls = true;
678
679             If (spikes is true)
680                 spikes = false;
681             Else
682                 spikes = true;
683
684             If (spaces is true)
685                 spaces = false;
686             Else
687                 spaces = true;
688
689             If (walls is true)
690                 walls = false;
691             Else
692                 walls = true;
693
694             If (spikes is true)
695                 spikes = false;
696             Else
697                 spikes = true;
698
699             If (spaces is true)
700                 spaces = false;
701             Else
702                 spaces = true;
703
704             If (walls is true)
705                 walls = false;
706             Else
707                 walls = true;
708
709             If (spikes is true)
710                 spikes = false;
711             Else
712                 spikes = true;
713
714             If (spaces is true)
715                 spaces = false;
716             Else
717                 spaces = true;
718
719             If (walls is true)
720                 walls = false;
721             Else
722                 walls = true;
723
724             If (spikes is true)
725                 spikes = false;
726             Else
727                 spikes = true;
728
729             If (spaces is true)
730                 spaces = false;
731             Else
732                 spaces = true;
733
734             If (walls is true)
735                 walls = false;
736             Else
737                 walls = true;
738
739             If (spikes is true)
740                 spikes = false;
741             Else
742                 spikes = true;
743
744             If (spaces is true)
745                 spaces = false;
746             Else
747                 spaces = true;
748
749             If (walls is true)
750                 walls = false;
751             Else
752                 walls = true;
753
754             If (spikes is true)
755                 spikes = false;
756             Else
757                 spikes = true;
758
759             If (spaces is true)
760                 spaces = false;
761             Else
762                 spaces = true;
763
764             If (walls is true)
765                 walls = false;
766             Else
767                 walls = true;
768
769             If (spikes is true)
770                 spikes = false;
771             Else
772                 spikes = true;
773
774             If (spaces is true)
775                 spaces = false;
776             Else
777                 spaces = true;
778
779             If (walls is true)
780                 walls = false;
781             Else
782                 walls = true;
783
784             If (spikes is true)
785                 spikes = false;
786             Else
787                 spikes = true;
788
789             If (spaces is true)
790                 spaces = false;
791             Else
792                 spaces = true;
793
794             If (walls is true)
795                 walls = false;
796             Else
797                 walls = true;
798
799             If (spikes is true)
800                 spikes = false;
801             Else
802                 spikes = true;
803
804             If (spaces is true)
805                 spaces = false;
806             Else
807                 spaces = true;
808
809             If (walls is true)
810                 walls = false;
811             Else
812                 walls = true;
813
814             If (spikes is true)
815                 spikes = false;
816             Else
817                 spikes = true;
818
819             If (spaces is true)
820                 spaces = false;
821             Else
822                 spaces = true;
823
824             If (walls is true)
825                 walls = false;
826             Else
827                 walls = true;
828
829             If (spikes is true)
830                 spikes = false;
831             Else
832                 spikes = true;
833
834             If (spaces is true)
835                 spaces = false;
836             Else
837                 spaces = true;
838
839             If (walls is true)
840                 walls = false;
841             Else
842                 walls = true;
843
844             If (spikes is true)
845                 spikes = false;
846             Else
847                 spikes = true;
848
849             If (spaces is true)
850                 spaces = false;
851             Else
852                 spaces = true;
853
854             If (walls is true)
855                 walls = false;
856             Else
857                 walls = true;
858
859             If (spikes is true)
860                 spikes = false;
861             Else
862                 spikes = true;
863
864             If (spaces is true)
865                 spaces = false;
866             Else
867                 spaces = true;
868
869             If (walls is true)
870                 walls = false;
871             Else
872                 walls = true;
873
874             If (spikes is true)
875                 spikes = false;
876             Else
877                 spikes = true;
878
879             If (spaces is true)
880                 spaces = false;
881             Else
882                 spaces = true;
883
884             If (walls is true)
885                 walls = false;
886             Else
887                 walls = true;
888
889             If (spikes is true)
890                 spikes = false;
891             Else
892                 spikes = true;
893
894             If (spaces is true)
895                 spaces = false;
896             Else
897                 spaces = true;
898
899             If (walls is true)
900                 walls = false;
901             Else
902                 walls = true;
903
904             If (spikes is true)
905                 spikes = false;
906             Else
907                 spikes = true;
908
909             If (spaces is true)
910                 spaces = false;
911             Else
912                 spaces = true;
913
914             If (walls is true)
915                 walls = false;
916             Else
917                 walls = true;
918
919             If (spikes is true)
920                 spikes = false;
921             Else
922                 spikes = true;
923
924             If (spaces is true)
925                 spaces = false;
926             Else
927                 spaces = true;
928
929             If (walls is true)
930                 walls = false;
931             Else
932                 walls = true;
933
934             If (spikes is true)
935                 spikes = false;
936             Else
937                 spikes = true;
938
939             If (spaces is true)
940                 spaces = false;
941             Else
942                 spaces = true;
943
944             If (walls is true)
945                 walls = false;
946             Else
947                 walls = true;
948
949             If (spikes is true)
950                 spikes = false;
951             Else
952                 spikes = true;
953
954             If (spaces is true)
955                 spaces = false;
956             Else
957                 spaces = true;
958
959             If (walls is true)
960                 walls = false;
961             Else
962                 walls = true;
963
964             If (spikes is true)
965                 spikes = false;
966             Else
967                 spikes = true;
968
969             If (spaces is true)
970                 spaces = false;
971             Else
972                 spaces = true;
973
974             If (walls is true)
975                 walls = false;
976             Else
977                 walls = true;
978
979             If (spikes is true)
980                 spikes = false;
981             Else
982                 spikes = true;
983
984             If (spaces is true)
985                 spaces = false;
986             Else
987                 spaces = true;
988
989             If (walls is true)
990                 walls = false;
991             Else
992                 walls = true;
993
994             If (spikes is true)
995                 spikes = false;
996             Else
997                 spikes = true;
998
999             If (spaces is true)
1000                 spaces = false;
1001             Else
1002                 spaces = true;
```

Two screenshots of a Windows desktop showing a debugger session in Visual Studio 2022.

Screenshot 1 (Top):

- Code View:** Shows the `Program.cs` file with code related to player movement and gravity. A break point is set at line 555.
- Watch View:** Shows variables in the current scope, including `PlayerStateS.Squash`, `PlayerStateS.Up`, `playerState`, `u`, `velocity`, and `velocityY`.
- Output View:** Displays diagnostic output from the application, including thread exits and loaded assemblies.
- Diagnostic Tools View:** Shows a timeline of events over 33 seconds, with a focus on Process Memory (MB) usage peaking at 15 MB and CPU usage at 100%.
- Terminal View:** Shows the command `dotnet-console-games` running in the terminal window.

Screenshot 2 (Bottom):

- Code View:** Same as Screenshot 1, showing the same code and break point at line 555.
- Watch View:** Same as Screenshot 1, showing the same variables.
- Output View:** Displays diagnostic output from the application, including thread exits and loaded assemblies.
- Diagnostic Tools View:** Shows a timeline of events over 10 seconds, with a focus on Process Memory (MB) usage peaking at 12 MB and CPU usage at 100%.
- Terminal View:** Shows the command `dotnet-console-games` running in the terminal window.

Two screenshots of a Windows desktop showing the development environment for a console game.

Screenshot 1 (Top):

- Code Editor:** Shows `Program.cs` with code related to game logic, specifically handling level completion and gravity settings.
- Output Window:** Displays the game's startup logs and exit messages for .NET threads.
- Diagnostic Tools:** A performance monitor showing CPU usage (~18%) and memory usage (~12 MB) over 27 seconds.
- Watch Window:** Shows the variable `level` is 0.
- System Tray:** Shows system status including battery level, signal strength, and system time (10:51 PM, 4/25/2025).

Screenshot 2 (Bottom):

- Code Editor:** Shows the same `Program.cs` file with a break point at line 555, which contains `velocity.Y = 0;`.
- Output Window:** Displays the game's startup logs and exit messages for .NET threads.
- Diagnostic Tools:** A performance monitor showing CPU usage (~100%) and memory usage (~12 MB) over 17 seconds.
- Watch Window:** Shows variables including `PlayerState.Squash`, `PlayerState.Up`, `PlayerState.Neutral`, `u` (value 1), `velocity` (X: 0, Y: -1), and `velocity.Y` (-1).
- System Tray:** Shows system status including battery level, signal strength, and system time (10:51 PM, 4/25/2025).

The screenshot shows two instances of the Microsoft Visual Studio IDE running side-by-side on a Windows desktop. Both instances are focused on a project named 'dotnet_console_games' with a file named 'Program.cs' open. The code in 'Program.cs' contains logic for a game level, including a 'GetStartingPlayerPositionFromLevel' method and a 'void Update()' loop. A break point is set at line 415. The 'Diagnostic Tools' window is visible in both panes, showing a 'Diagnostics session: 35 seconds (1 ms selected)' for the top instance and a 'Diagnostics session: 27 seconds (1 ms selected)' for the bottom instance. The 'Events' tab in the Diagnostic Tools shows several events, notably 'Process Memory (MB)' which fluctuates between 0 and 12. The 'CPU (%) of all processors' chart shows usage around 100%. The bottom instance also displays a terminal window showing the game's output, including logs about loaded assemblies and exiting threads.

Microsoft Visual Studio Debug

```

Unhandled exception: System.IndexOutOfRangeException: Index was outside the
bounds of the array.
   at Program.<>Main>$>g__Render|0_7(<>c__DisplayClass0_0&)
   at Program.<>Main>$>g__Render|0_7()
   at Program.<Main>$>Main()
C:\Users\Rashid\OneDrive\Desktop\STT_Codes\Lab 11\dotnet-console-games\Projects\Gravity\Program.cs:line 723
   at Program.<Main>$>Main()
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```

Program.cs

```

706     append(' ');
707     append(' ');
708     for (j = 0; j < levels[level][i].Length; j++)
709     {
710         c = levels[level][i][j];
711         if (c == 'g')
712             c = ' ';
713         if (c != 'X' and not 'e' &&
714             PlayerPosition.X - 3 <= j &&
715             PlayerPosition.X + 3 >= j &&
716             PlayerPosition.Y - 2 <= i &&
717             PlayerPosition.Y + 2 >= i)
718             c = playersprite[i - PlayerPosition.Y + 1][j] - PlayerPosition.X + 1;
719         else
720             c = ' ';
721         if (c != ' ')
722             layerState();
723         else
724             c = playersprite[i - PlayerPosition.Y + 1][j] - PlayerPosition.X + 2;
725         layer.Append(c);
726         layer.AppendLine();
727     }
728     WriteLine();
729     WriteLine($"Level: {level} Gravity: {RenderGravityIdentifier()}");
730     CursorPosition(0, 0);
731     if (render)
732         isVisible = false;
733     else
734         isVisible = true;
735     layerState();
736     if (isVisible)
737         verState();
738     else
739         verState();
740     switch
    
```

Output

```

The thread 'NET.TP.Worker' (16152) has exited with code 0 (0x0).
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NET\6.0.3\clrhost\Gravity.exe'
The program '[26384] Gravity.exe' has exited with code 0 (0x0).

```

Ready

32°C Smoke

File Edit View Git Project Build Test Analyze GitHub Copilot

Microsoft Visual Studio

File Edit View Git Project Build Test Analyze GitHub Copilot

Program.cs

```

706     append(' ');
707     append(' ');
708     for (j = 0; j < levels[level][i].Length; j++)
709     {
710         c = levels[level][i][j];
711         if (c == 'g')
712             c = ' ';
713         if (c != 'X' and not 'e' &&
714             PlayerPosition.X - 3 <= j &&
715             PlayerPosition.X + 3 >= j &&
716             PlayerPosition.Y - 2 <= i &&
717             PlayerPosition.Y + 2 >= i)
718             c = playersprite[i - PlayerPosition.Y + 1][j] - PlayerPosition.X + 1;
719         else
720             c = ' ';
721         if (c != ' ')
722             layerState();
723         else
724             c = playersprite[i - PlayerPosition.Y + 1][j] - PlayerPosition.X + 2;
725         layer.Append(c);
726         layer.AppendLine();
727     }
728     WriteLine();
729     WriteLine($"Level: {level} Gravity: {RenderGravityIdentifier()}");
730     CursorPosition(0, 0);
731     if (render)
732         isVisible = false;
733     else
734         isVisible = true;
735     layerState();
736     if (isVisible)
737         verState();
738     else
739         verState();
740     switch
    
```

Output

```

Show output from: Debug
The thread 'NET.TP.Worker' (16152) has exited with code 0 (0x0).
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NET\6.0.3\clrhost\Gravity.exe'
The program '[26384] Gravity.exe' has exited with code 0 (0x0).

```

Diagnostic Tools

Diagnostics session: 5 seconds (2.635 s selected)

Events

Process Memory (MB)

CPU (% of all processors)

Summary Events Counters Memory Usage

Events

.NET Counters

View Performance Counters

Memory Usage

Take Snapshot

Autos

Name	Value	Type
PlayerPosition	(9, 15)	(int X, int Y)
PlayerPositionX	9	int
PlayerPositionY	15	int
c	' '	char
i	13	int
j	6	int

Output

```

Show output from: Debug
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\Microsoft Visual Studio\2022\Community\IDE\Prv\Gravity.dll'
'Gravity.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NET\6.0.3\netstandard.dll'
Exception thrown: 'System.IndexOutOfRangeException' in Gravity.dll
An unhandled exception of type 'System.IndexOutOfRangeException' occurred in Gravity.dll
Index was outside the bounds of the array.

```