



CONSTRUCTION COMPILER LAB TERMINAL (Q5)

SUBMITTED BY : FAJAR AAMIR SHEIKH

REGISTRATION NO: SP22-BCS-031

SUBMITTED TO : SIR BILAL BUKHARI

SUBMISSION DATE: 18THJUNE2025

QUESTION NO 5:

Design a Domain-Specific Language (DSL) in C# to define and generate gameplay elements like police units, criminal waves, backup support, and city levels for a dynamic police shooter game.

DSL Specification

Purpose

This Domain-Specific Language (DSL) defines the gameplay elements of a police shooter game: police units, criminal waves, backup support, zones, and objectives.

DSL Keywords & Syntax

Keyword	Description	Syntax Example
level	Level title	level Downtown Standoff
zone	Game map location (Downtown, Suburbs, Industrial)	zone Downtown
difficulty	Level difficulty (Easy, Medium, Hard)	difficulty Medium
time	Time limit in minutes	time limit 15
unit	Police unit definition	unit SWAT 2 150 80 Rifle
backup	Backup unit configuration	backup Helicopter 60 OnDemand
objective	Game or wave objective	objective Defeat all criminals
wave	Start of a new criminal wave	wave 1
criminals	Criminal group in a wave	criminals Gangster 4 80 50 Shotgun

Keyword	Description	Syntax Example
trigger	Trigger condition to spawn wave	trigger Immediate

Example DSL Script

```

level Downtown Standoff
zone Downtown
difficulty Medium
time limit 15

unit Patrol 4 100 60 Pistol
unit SWAT 2 150 80 Rifle

backup SWAT 30 AutoWhenLow
backup Helicopter 60 OnDemand

objective Defeat all criminals
objective Protect civilians

```

```

wave 1
criminals Thief 8 50 30 Pistol
criminals Gangster 4 80 50 Shotgun
trigger Immediate
objective Stop initial assault

wave 2
criminals Gangster 6 100 60 Rifle
trigger PreviousDefeated
objective Secure area

```

Parser Implementation

- Function: Parse(string script)

- Responsibility: Parses a DSL script and builds an in-memory Level object.

Key Elements Parsed:

- Level → Name, Zone, Difficulty, Time
- Units → PoliceUnit objects with stats
- Backups → Backup records with arrival delay and condition
- Objectives → Global objectives
- Waves → Multiple criminal groups, each with a trigger and objective

Example Object Created:

```
new PoliceUnit(UnitType.SWAT, 2, 150, 80, Weapon.Rifle)
```

Interpreter

The interpreter executes the game logic based on the parsed Level object:

Key Functions:

- Load() – Initializes level data
- Run() – Runs the game loop
- SpawnWave() – Displays a wave when triggered
- Attack() – Simulates attacking criminal waves
- CallBackup() – Adds backup units to player's team
- Update() – Reduces health of both sides each round

Input Commands:

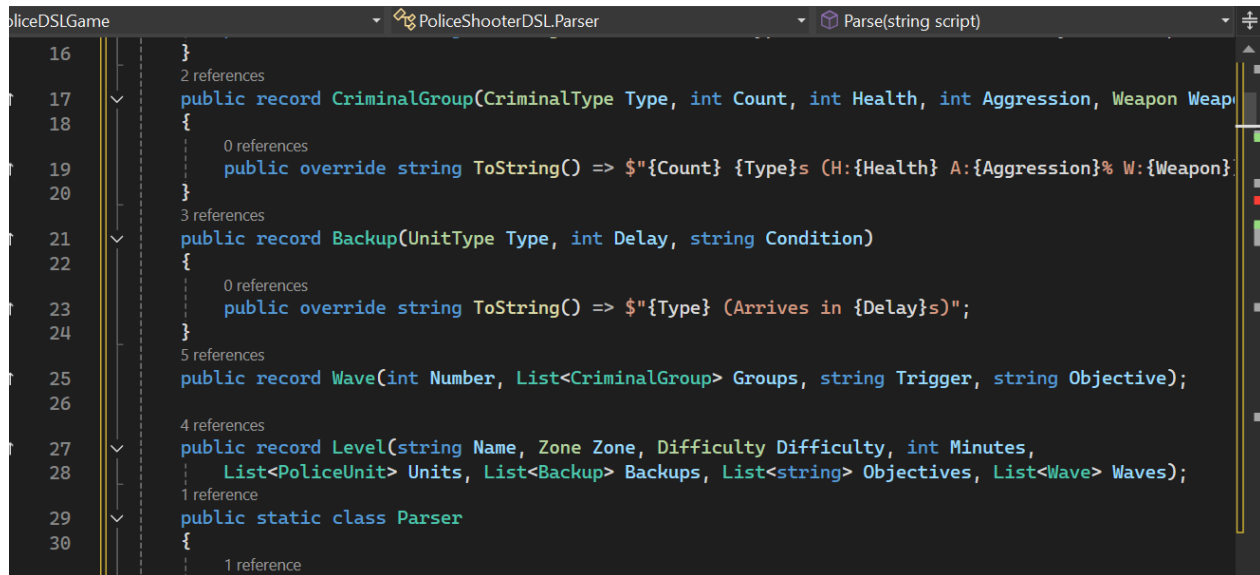
Key Action

- A Attack
- B Call Backup
- H Show Help
- X Surrender

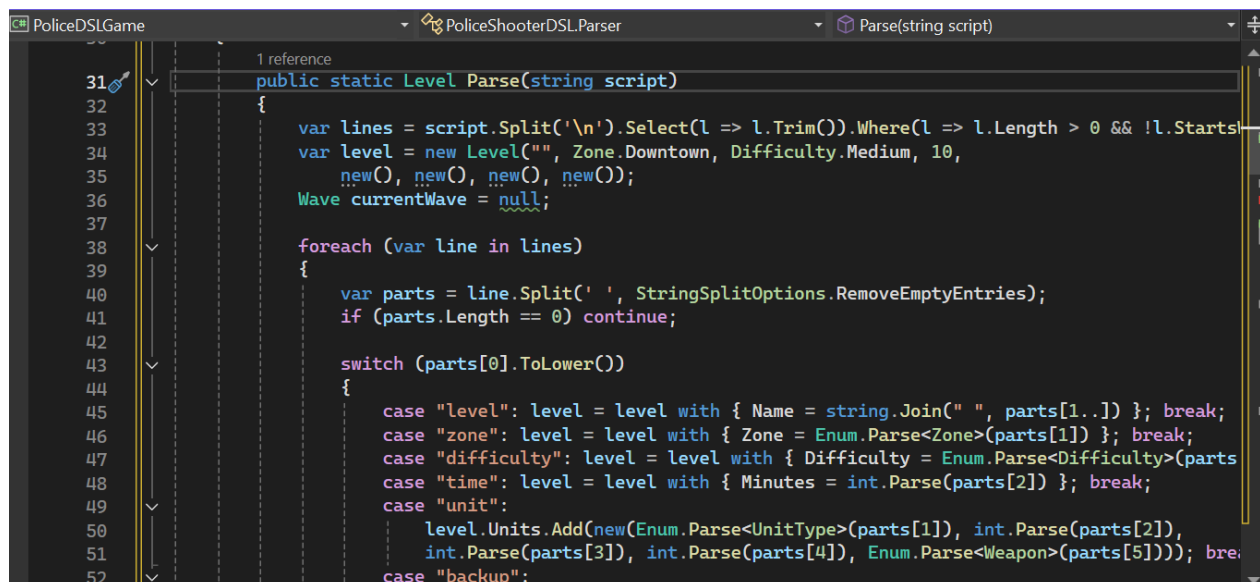
Game Prototype

While the current prototype is console-based, the structure allows easy expansion to a 2D or 3D game engine like Unity or Godot:

- Units and Waves are data-driven.
- Level design is fully dynamic via DSL.
- Could be plugged into a Unity prefab spawner.



```
16 }
17 2 references
18 public record CriminalGroup(CriminalType Type, int Count, int Health, int Aggression, Weapon Weapon)
19 {
20     0 references
21     public override string ToString() => $"{Count} {Type}s (H:{Health} A:{Aggression}% W:{Weapon})";
22 }
23 3 references
24 public record Backup(UnitType Type, int Delay, string Condition)
25 {
26     0 references
27     public override string ToString() => $"{Type} (Arrives in {Delay}s)";
28 }
29 5 references
30 public record Wave(int Number, List<CriminalGroup> Groups, string Trigger, string Objective);
31
32 4 references
33 public record Level(string Name, Zone Zone, Difficulty Difficulty, int Minutes,
34     List<PoliceUnit> Units, List<Backup> Backups, List<string> Objectives, List<Wave> Waves);
35 1 reference
36 public static class Parser
37 {
38     1 reference
```



```
31 1 reference
32 public static Level Parse(string script)
33 {
34     var lines = script.Split('\n').Select(l => l.Trim()).Where(l => l.Length > 0 && !l.StartsWith('#'));
35     var level = new Level("", Zone.Downtown, Difficulty.Medium, 10,
36         new(), new(), new(), new());
37     Wave currentWave = null;
38
39     foreach (var line in lines)
40     {
41         var parts = line.Split(' ', StringSplitOptions.RemoveEmptyEntries);
42         if (parts.Length == 0) continue;
43
44         switch (parts[0].ToLower())
45         {
46             case "level": level = level with { Name = string.Join(" ", parts[1..]) }; break;
47             case "zone": level = level with { Zone = Enum.Parse<Zone>(parts[1]) }; break;
48             case "difficulty": level = level with { Difficulty = Enum.Parse<Difficulty>(parts[1]) }; break;
49             case "time": level = level with { Minutes = int.Parse(parts[1]) }; break;
50             case "unit":
51                 level.Units.Add(new(Enum.Parse<UnitType>(parts[1]), int.Parse(parts[2]),
52                     int.Parse(parts[3]), int.Parse(parts[4]), Enum.Parse<Weapon>(parts[5]))); break;
53             case "backup":
```

```
PoliceDSLGame PoliceShooterDSL.Parser Parse(string script)
42
43 switch (parts[0].ToLower())
44 {
45     case "level": level = level with { Name = string.Join(" ", parts[1..]) }; break;
46     case "zone": level = level with { Zone = Enum.Parse<Zone>(parts[1]) }; break;
47     case "difficulty": level = level with { Difficulty = Enum.Parse<Difficulty>(parts[1]) }; break;
48     case "time": level = level with { Minutes = int.Parse(parts[2]) }; break;
49     case "unit":
50         level.Units.Add(new(Enum.Parse<UnitType>(parts[1]), int.Parse(parts[2]),
51             int.Parse(parts[3]), int.Parse(parts[4]), Enum.Parse<Weapon>(parts[5]))); break;
52     case "backup":
53         level.Backups.Add(new(Enum.Parse<UnitType>(parts[1]),
54             int.Parse(parts[2]), parts.Length > 3 ? parts[3] : "")); break;
55     case "objective": level.Objectives.Add(string.Join(" ", parts[1..])); break;
56     case "wave":
57         currentWave = new(int.Parse(parts[1]), new(), "", "");
58         level.Waves.Add(currentWave); break;
59     case "criminals" when currentWave != null:
60         currentWave.Groups.Add(new(Enum.Parse<CriminalType>(parts[1]),
61             int.Parse(parts[2]), int.Parse(parts[3]), int.Parse(parts[4]),
62             Enum.Parse<Weapon>(parts[5]))); break;
63     case "trigger" when currentWave != null:
64         currentWave = currentWave with { Trigger = string.Join(" ", parts[1..]) }; break;
```

```
PoliceDSLGame PoliceShooterDSL.Parser Parse(string script)
71 }
72 2 references
73 public class Game
74 {
75     private readonly Random rnd = new();
76     private Level level;
77     private List<PoliceUnit> units = new();
78     private int waveIndex = 0;
79     private readonly List<Backup> calledBackup = new();
80
81     1 reference
82     public void Load(Level level)
83     {
84         this.level = level;
85         units = new(level.Units);
86         waveIndex = 0;
87         calledBackup.Clear();
88
89         Console.WriteLine($"=== {level.Name} ===\nZone: {level.Zone}\nDifficulty: {level.Difficulty}\nUnits: " + string.Join(", ", units));
90         Console.WriteLine("\nObjectives:\n- " + string.Join("\n- ", level.Objectives));
91         Console.WriteLine("\nPress any key to start...");
92         Console.ReadKey();
93     }
94 }
```

```
PoliceDSLGame PoliceShooterDSL.Game ShouldSpawn(Wave wave)
94 {
95     while (!IsOver())
96     {
97         Console.Clear();
98         Console.WriteLine($"Wave: {waveIndex}/{level.Waves.Count}\nUnits: {units.Count}\n");
99
100         if (waveIndex < level.Waves.Count && ShouldSpawn(level.Waves[waveIndex]))
101             SpawnWave(level.Waves[waveIndex++]);
102
103         ProcessInput();
104         Update();
105     }
106     Console.WriteLine(units.Count > 0 ? "=== MISSION COMPLETE ===" : "=== MISSION FAILED ===");
107 }
108 1 reference
private bool IsOver() => waveIndex >= level.Waves.Count || units.Count == 0;
109 1 reference
private bool ShouldSpawn(Wave wave) => waveIndex == 0 || wave.Trigger.Contains("Defeated");
110 1 reference
private void SpawnWave(Wave wave)
111 {
112     Console.WriteLine($"\\n=== WAVE {wave.Number} ===\\nObjective: {wave.Objective}");
113     wave.Groups.ForEach(g => Console.WriteLine($"- {g}"));
114     SpawnUnits(wave);
115 }
```

```
PoliceDSLGame PoliceShooterDSL.Game Attack()
115 }
116 1 reference
private void ProcessInput()
117 {
118     Console.Write("\\n(A)ttack (B)ackup (H)elp: ");
119     switch (Console.ReadKey().Key)
120     {
121     case ConsoleKey.A: Attack(); break;
122     case ConsoleKey.B: CallBackup(); break;
123     case ConsoleKey.H: ShowHelp(); break;
124     }
125 }
126 1 reference
private void Attack()
127 {
128     if (waveIndex == 0) return;
129     var wave = level.Waves[waveIndex - 1];
130     var damage = units.Sum(u => u.Accuracy * u.Count / 10);
131     wave.Groups.ForEach(g => g = g with { Health = Math.Max(0, g.Health - damage / wave.Group.Count)});
132     Console.WriteLine($"Dealt {damage} damage!");
133     Console.ReadKey();
134 }
135 1 reference
private void CallBackup()
```

```
PoliceDSLGame PoliceShooterDSL.Game Attack()
134 }
135 1 reference
136 private void CallBackup()
137 {
138     var available = level.Backups.Except(calledBackup).ToList();
139     if (available.Count == 0) return;
140     Console.WriteLine("\nAvailable: " + string.Join(", ", available.Select((b, i) => $"{i + 1} {b.Type}"));
141     if (int.TryParse(Console.ReadKey().KeyChar.ToString(), out int choice) && choice <= available.Count)
142     {
143         var backup = available[choice - 1];
144         calledBackup.Add(backup);
145         units.Add(new(backup.Type, backup.Type == UnitType.Helicopter ? 1 : rnd.Next(2, 5),
146             100, 70 + rnd.Next(20), backup.Type switch
147             {
148                 UnitType.SWAT => Weapon.Rifle,
149                 UnitType.Helicopter => Weapon.MachineGun,
150                 _ => Weapon.Pistol
151             }));
152         Console.WriteLine($"{backup.Type} backup arrived!");
153         Console.ReadKey();
154     }
155 }
156 }
```

```
PoliceDSLGame PoliceShooterDSL.Game ShowHelp()
159 Console.WriteLine("Commands: 'A' Attack and Backup and Help.");
160 Console.ReadKey();
161 }
162 1 reference
163 private void Update()
164 {
165     if (waveIndex == 0) return;
166     var wave = level.Waves[waveIndex - 1];
167     var damage = wave.Groups.Sum(g => g.Aggression * g.Count / 10);
168     units.ForEach(u => u = u with { Health = Math.Max(0, u.Health - damage / units.Count) });
169     units.RemoveAll(u => u.Health <= 0);
170 }
171 0 references
172 class Program
173 {
174     0 references
175     static void Main()
176     {
177         const string script = @"
178             level Downtown Standoff
179             zone Downtown
180             difficulty Medium
181             time limit 15
182
183             unit Patrol 4 100 60 Pistol
184             unit SWAT 2 150 80 Rifle
185         ";
186     }
```



```
PoliceDSLGame
PoliceShooterDSLGame
ShowHelp()

181      unit Patrol 4 100 60 Pistol
182      unit SWAT 2 150 80 Rifle
183
184      backup SWAT 30 AutoWhenLow
185      backup Helicopter 60 OnDemand
186
187      objective Defeat all criminals
188      objective Protect civilians
189
190      wave 1
191      criminals Thief 8 50 30 Pistol
192      criminals Gangster 4 80 50 Shotgun
193      trigger Immediate
194      objective Stop initial assault
195
196      wave 2
197      criminals Gangster 6 100 60 Rifle
198      trigger PreviousDefeated
199      objective Secure area";
200
201      var level = Parser.Parse(script);
202      new Game().Load(level);
203      new Game().Run();
204  }
205  }
206 }
```

OUTPUT:

```
C:\Users\HP\source\repos\Co  X + v

=== Downtown Standoff ===
Time elapsed: 0.0/15 minutes
Current Wave: None/3

YOUR UNITS:
- 4 PatrolOfficer(s) - Health: 100, Accuracy: 60%, Weapon: Pistol
- 2 SWAT(s) - Health: 150, Accuracy: 80%, Weapon: AssaultRifle

AVAILABLE COMMANDS:
A - Attack enemies
B - Request backup
H - Help
X - Surrender

=== INCOMING WAVE 1 ===
OBJECTIVE: Stop the initial assault
- 8 Thief(s) appeared!
  Armed with: Pistol
  Threat level: 30/100
- 4 Gangster(s) appeared!
  Armed with: Shotgun
  Threat level: 50/100

Press any key to continue...

Enter your command: A

Your units attack for 40 total damage!
8 Thiefs took 20 damage. Remaining health: 30
4 Gangsters took 20 damage. Remaining health: 60

Press any key to continue...
```

```
=== Downtown Standoff ===  
Time elapsed: 0.2/15 minutes  
Current Wave: 1/3
```

YOUR UNITS:

- 4 PatrolOfficer(s) - Health: 78, Accuracy: 60%, Weapon: Pistol
- 2 SWAT(s) - Health: 128, Accuracy: 80%, Weapon: AssaultRifle

ENEMY FORCES:

- 8 Thief(s) - Health: 30
- 4 Gangster(s) - Health: 60

AVAILABLE COMMANDS:

- A - Attack enemies
- B - Request backup
- H - Help
- X - Surrender

Enter your command:

PRESS H-HELP

```
=== Downtown Standoff ===  
Time elapsed: 0.0/15 minutes  
Current Wave: None/3
```

YOUR UNITS:

- 4 PatrolOfficer(s) - Health: 100, Accuracy: 60%, Weapon: Pistol
- === POLICE SHOOTER - HELP ===

COMMANDS:

- A - Attack: All your units will attack the current wave of enemies
- B - Backup: Call for reinforcements (if available)
- H - Help: Show this help screen
- X - Surrender: Give up the mission

GAME MECHANICS:

- Each attack does damage based on your units' accuracy and count
- Enemies will counterattack after your turn
- Some backup arrives automatically when your health is low
- Complete all waves within the time limit to win

Press any key to return to game...

PRESS X-SURRENDER

ENEMY FORCES:

- 8 Thief(s) - Health: 30
- 4 Gangster(s) - Health: 60

AVAILABLE COMMANDS:

- A - Attack enemies
- B - Request backup
- H - Help
- X - Surrender

Enter your command: x

You have surrendered!

=== ENEMY COUNTERATTACK ===

8 Thiefs attack for 24 total damage!

4 Gangsters attack for 20 total damage!