Prison Cell Corridors - Design Guide

🔒 Overview

Prison cell corridors are special structures in the maze featuring **11-cell long corridors** lined with **individual prison cells** on both sides. Prisoners are preferentially spawned in these thematic cells, creating authentic dungeon prison blocks.

📐 Structure Design

Single Prison Cell Corridor

Horizontal Layout (11 cells long):

```
# # # # # # # # # # # # #
# ╔═╗ # ╔═╗ # ╔═╗ # ╔═╗ # ← North cells (5 total)
# ‖ ‖ # ‖ ‖ # ‖ ‖ # ‖ ‖ #
# ╚╝ # ╚╝ # ╚╝ # ╚╝ #
# · · · · · · · · · · · # ← Main corridor (11 cells)
# ╔╗ # ╔╗ # ╔╗ # ╔╗ #
# ‖ ‖ # ‖ ‖ # ‖ ‖ # ‖ ‖ #
# ╚═╝ # ╚═╝ # ╚═╝ # ╚═╝ # ← South cells (5 total)
# # # # # # # # # # # # #
```

Legend:
· = Corridor path (11 cells)
‖ = Cell interior (prisoner location)
╔╗ ╚═╝ = Walls (3 sides per cell)
= Maze walls

Total: 10 cells per corridor (5 each side)

Detailed Cell Structure

Single Prison Cell (Top View):

```
 # # #
 # 🔒 # ← Opening to corridor
 ╔═══╗
 ‖   ‖ ← 1×1 interior
 ‖ 🟠 ‖ ← Prisoner position
 ╚═══╝
```

Components:
- 1 cell opening (faces corridor)

- 3 walls (back + 2 sides)
- 1×1 interior space
- Orange light when occupied

Vertical Prison Corridor

```
# # ╔═╗ · ╔═╗ # #
# # ╢ ╟ · ╢ ╟ # #  ← West cells
# # ╚═╝ · ╚═╝ # #
# # # # · # # # #
# # ╔═╗ · ╔═╗ # #
# # ╢ ╟ · ╢ ╟ # #  ← East cells
# # ╚═╝ · ╚═╝ # #
# # # # · # # # #
# # ╔═╗ · ╔═╗ # #
# # ╢ ╟ · ╢ ╟ # #
# # ╚═╝ · ╚═╝ # #
# # # # · # # # #
    ↑
 Corridor (11 cells vertical)
```

🏗️ Generation Algorithm

Step 1: Find Suitable Locations

1. Scan maze for straight paths
2. Check 11-cell straight availability
3. Verify space for cells on both sides
4. Calculate distribution score
5. Sort by quality (distance-based)

Step 2: Create Corridors

For each selected location:
1. Clear 11-cell corridor path
2. Every 2nd cell (odd indices 1,3,5,7,9):
   → Create cell on left side
   → Create cell on right side
3. Track cell positions for prisoners
4. Maintain wall structure

Step 3: Place Prisoners

Priority System:
1. 70% in prison cells (thematic)

2. 30% in regular maze (variety)
3. Even distribution maintained
4. Distance algorithm still applies

📊 Technical Specifications

Dimensions

yaml

Corridor Length: 11 cells
Cell Spacing: Every 2 cells (odd positions)
Cells Per Corridor: 10 (5 each side)
Cell Size: 1×1 interior
Total Footprint: 13×3 cells (horizontal)
or 3×13 cells (vertical)

Generation Parameters

csharp

Prison Corridor Count: 8 (default, configurable 3-15)
Cell Count: 80 maximum (8 corridors × 10 cells)
Coverage: ~0.8% of 100×100 dungeon
Prisoner Preference: 70% in cells, 30% maze

🎮 Gameplay Impact

Visual Recognition

Player sees:
- Long straight corridor
- Cells on both sides
- Orange lights (prisoners)
- Symmetrical layout

Player thinks:
"This is a prison block!"
"Lots of prisoners here"
"I should explore these"

Exploration Flow

Enter prison corridor
↓
"Wow, prison cells!"

↓

Check each cell systematically

↓

Multiple prisoners in one area

↓

Efficient rescue (cluster)

↓

Continue to next section

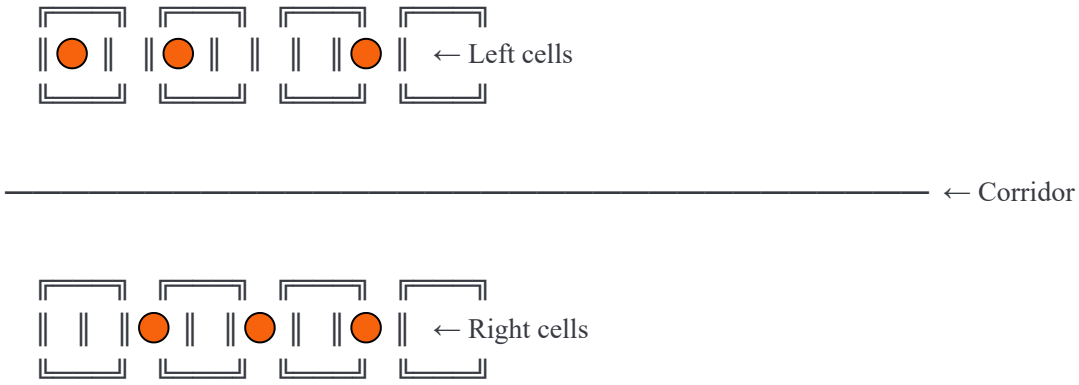
Strategic Benefits

✅ **Efficiency** - Multiple prisoners in one area ✅ **Thematic** - Feels like real dungeon prison ✅ **Visual variety** - Breaks up maze monotony ✅ **Landmark** - Recognizable structures ✅ **Storytelling** - "This was the prison block"

🎨 Visual Examples

3D Perspective (Horizontal)

Player View (Looking Down Corridor):



● = Prisoner with orange light

Empty cells have no prisoner (already rescued or never occupied)

Lighting Pattern

Orange lights create dramatic corridor lighting

Multiple glows visible from distance

Clear visual guide to prison area

🔧 Distribution Algorithm

Cell Prioritization

Total Prisoners: 10
Prison Cells Available: 80 (8 corridors)

Distribution:
- 7 in prison cells (70%)
- 3 in maze (30%)

Cell Selection:
- Spread across different corridors
- Even distribution maintained
- Distance algorithm applied

Example Placement (10 prisoners, 8 corridors)

Corridor 1: 1 prisoner
Corridor 2: 1 prisoner
Corridor 3: 0 prisoners
Corridor 4: 1 prisoner
Corridor 5: 1 prisoner
Corridor 6: 1 prisoner
Corridor 7: 1 prisoner
Corridor 8: 1 prisoner

Maze: 3 prisoners (scattered)

Result: Prisoners across entire dungeon

🎯 Configuration Options

Change Corridor Count

csharp

```
// In MazeGenerator inspector
public int prisonCorridorCount = 8; // ← Change (3-15)

More corridors = More cells = More prisoners in cells
Fewer corridors = More scattered prisoners
```

Change Cell Preference

csharp

```
// In SelectPrisonerPositions()
int cellCount = Mathf.CeilToInt(count * 0.7f); // ← Change 0.7 (70%)

0.9 = 90% in cells (very thematic)
0.5 = 50/50 split (balanced)
0.3 = 30% in cells (prefer maze)
```

Longer Corridors

```
csharp

// In CreateHorizontalPrisonCorridor()
for (int i = 0; i < 15; i++) // ← Change from 11
{
    // More cells per corridor
}
```

Different Cell Spacing

```
csharp

// In CreateHorizontalPrisonCorridor()
if (i % 3 == 1) // ← Change from i % 2 == 1
{
    // Every 3rd cell instead of every 2nd
    // Fewer, more spaced cells
}
```

📐 Placement Scoring

Distribution Score Calculation

```
csharp

score = distanceFromEntrance + minDistanceFromExisting

Higher score = Better placement

Prioritizes:
1. Far from entrance (deep in dungeon)
2. Far from other corridors (spread out)
```

Example Scores

```
Corridor A: (90, 20) → Distance 85 + Existing 45 = 130 ✓ Best
Corridor B: (50, 50) → Distance 45 + Existing 20 = 65
```

Corridor C: (20, 80) → Distance 75 + Existing 15 = 90
Corridor D: (60, 10) → Distance 55 + Existing 35 = 90

Selected: A, C, D, B (in order of score)

🐛 Debugging

Visualize Prison Corridors

csharp

```
void OnDrawGizmos()
{
    foreach (var cellPos in prisonCellPositions)
    {
        Gizmos.color = Color.yellow;
        Gizmos.DrawCube(
            new Vector3(cellPos.x * 2, 1, cellPos.y * 2),
            Vector3.one * 0.8f
        );
    }
}
```

Check Generation

csharp

```
Debug.Log($"Prison corridors: {prisonCorridorCount}");
Debug.Log($"Total cells: {prisonCellPositions.Count}");
Debug.Log($"Expected: {prisonCorridorCount * 10}");
```

Test Cell Placement

csharp

```
[ContextMenu("Show Prison Stats")]
void ShowPrisonStats()
{
    int inCells = prisoners.Count(p => prisonCellPositions.Contains(p.gridPosition));
    int inMaze = prisoners.Count - inCells;

    Debug.Log($"Prisoners in cells: {inCells}/{prisoners.Count}");
    Debug.Log($"Prisoners in maze: {inMaze}/{prisoners.Count}");
}
```

🏆 Benefits

Thematic Authenticity

✓ Feels like actual dungeon prison
✓ "Jailer Guild" theme reinforced
✓ Clear purpose to structures
✓ Immersive worldbuilding

Gameplay Advantages

✓ Clustered rescues (efficient)
✓ Visual landmarks (navigation)
✓ Strategic planning (hit prison first?)
✓ Variety in maze layout

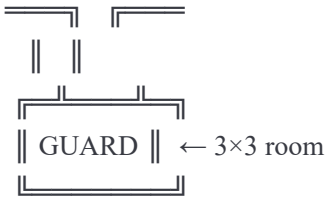
Technical Benefits

✓ Organized structure (not random)
✓ Predictable generation
✓ Easy to modify/extend
✓ Good performance (simple shapes)

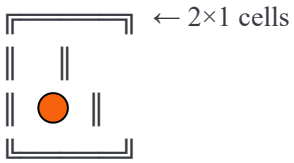
🎨 Alternative Designs

Guard Towers (Central Rooms)

Every 3rd cell position:



Double-Wide Cells



L-Shaped Cells

✅ Testing Checklist

- [ ] Prison corridors generate (8 default)
- [ ] Each corridor is 11 cells long
- [ ] Cells appear every 2 cells
- [ ] 10 cells per corridor (5 each side)
- [ ] Corridors spread across dungeon
- [ ] ~70% prisoners in cells
- [ ] ~30% prisoners in maze
- [ ] Even distribution maintained
- [ ] Orange lights in cells
- [ ] Can walk into cells
- [ ] Cells have 3 walls
- [ ] Works with 100×100 dungeon
- [ ] No generation errors
- [ ] Looks visually distinct

🎯 Recommended Settings

Small Dungeon (15×15)

```
Prison Corridors: 2-3
Prisoner Count: 5
Cell Preference: 60%
```

Medium Dungeon (50×50)

```
Prison Corridors: 5-6
Prisoner Count: 10
Cell Preference: 70%
```

Large Dungeon (100×100)

```
Prison Corridors: 8-10
Prisoner Count: 15-20
Cell Preference: 70-80%
```

Prison Block 🔒 - Authentic dungeon prison corridors with individual cells!