Historical Data Package - A War Without Victory

Overview

This package provides comprehensive historical data for the Bosnian War strategic game, including:

- **109 municipality adjacency mappings** (who borders who)
- **24 JNA garrison locations** with detailed unit information
- **8 strategic corridors** with vulnerability assessments
- **Strategic importance ratings** for key locations
- **Ethnic pattern classifications** for stability calculations

This data enables realistic:

- Cascade flip mechanics (adjacent municipalities influence each other)
- Geographic pressure calculations
- JNA garrison impact on control
- Strategic corridor tracking and warnings
- Enhanced control stability calculations

Files in This Package

1. `historical_data.js` (Core Data)

Size: ~850 lines **Purpose**: Raw historical data in JavaScript constants

Contents:

- `MUNICIPALITY_ADJACENCIES` - Complete adjacency graph
- `JNA_GARRISONS` - 24 garrison locations with troop counts
- `STRATEGIC_CORRIDORS` - 8 major corridors with vulnerability ratings
- `STRATEGIC_IMPORTANCE` - Game-mechanical importance ratings
- `ETHNIC_PATTERNS` - Regional ethnic classifications

Usage: Include as first script, provides data foundation

---

2. `historical_data_integration.js` (Integration Functions)

Size: ~650 lines **Purpose**: Functions to merge historical data into game state

Key Functions:

- `integrateHistoricalData()` - Main integration function (call once)
- `getAdjacentMunicipalities()` - Query adjacencies
- `calculateGeographicPressure()` - Pressure from neighbors
- `hasJNAGarrison()` / `getJNAGarrisonStrength()` - Garrison queries
- `checkCorridorIntegrity()` - Monitor corridor status
- `calculateEnhancedControlStability()` - Improved stability with geography
- `generateMunicipalitySitRep()` - Comprehensive status reports

Usage: Include as second script, provides game integration

---

3. `INTEGRATION_GUIDE.md` (How-To Documentation)

Size: ~450 lines **Purpose**: Step-by-step guide to integrate data into your game

Sections:

1. Files overview
2. Integration steps (detailed, with code snippets)
3. Testing and verification procedures
4. Common issues and solutions
5. Performance notes
6. Next steps for development

Usage: Read first before integrating, reference during integration

---

4. `QUICK_REFERENCE.md` (Developer Cheat Sheet)

Size: ~750 lines **Purpose**: Quick lookup for functions and municipality data

Sections:

- Key municipality IDs and importance ratings
- JNA garrison quick reference table
- Complete function reference with examples
- Usage examples for common tasks

- Data constants reference
- Performance tips
- Debugging helpers

Usage: Keep open while coding, quick function lookup

---

5. `GEOGRAPHIC_REFERENCE.md` (Strategic Overview)

Size: ~650 lines **Purpose**: Visual and strategic context

Sections:

- ASCII map of Bosnia showing regions
- Detailed regional breakdowns (9 major regions)
- Strategic chokepoints
- Distance reference between key cities
- JNA deployment summary
- Corridor vulnerability assessment
- Phase-based situation analysis
- Game-mechanical implications

Usage: Strategic planning, understanding context, testing scenarios

---

Quick Start Guide

Step 1: Read the Integration Guide

```bash
# Start here
open INTEGRATION_GUIDE.md
```

Step 2: Add Scripts to Your HTML

```html
<script src="historical_data.js"></script>
<script src="historical_data_integration.js"></script>
```

Step 3: Call Integration After Municipality Creation

```javascript
// In initializeGameState(), after creating all municipalities:
integrateHistoricalData(gameState);

// Recalculate stabilities with geographic data:
Object.keys(gameState.municipalities).forEach(munId => {
    gameState.municipalities[munId].stability =
        calculateEnhancedControlStability(gameState, munId);
});
```

Step 4: Test Integration

```javascript
// In browser console:
console.log(getAdjacentMunicipalities(10129)); // Sarajevo adjacents
console.log(hasJNAGarrison(10812)); // Banja Luka - should be true
console.log(generateMunicipalitySitRep(gameState, 10812));
```

Step 5: Reference Documentation As Needed

- **QUICK_REFERENCE.md** - Function usage while coding
- **GEOGRAPHIC_REFERENCE.md** - Strategic context and testing

---

Integration Checklist

- [ ] Add both .js files to HTML
- [ ] Call `integrateHistoricalData(gameState)` after municipality creation
- [ ] Verify console shows "Historical data integration complete"
- [ ] Test: Check Sarajevo (10129) shows adjacents
- [ ] Test: Check Banja Luka (10812) shows JNA garrison
- [ ] Test: Check Gradiška (10863) shows Posavina Corridor
- [ ] Update municipality detail panel to show new data
- [ ] Add corridor warnings to situation display
- [ ] Update stability calculations to use `calculateEnhancedControlStability()`
- [ ] Test flip mechanics with cascade pressure
- [ ] Optional: Add JNA presence map mode

Data Completeness

✅ Complete (Ready to Use)

- All municipality adjacencies (109 municipalities)
- Major JNA garrisons (24 locations)
- Strategic corridors (8 corridors)
- Strategic importance ratings (key locations)
- Ethnic pattern classifications

⚠ Partial (Can Be Enhanced)

- Minor JNA posts (some small garrisons omitted)
- Detailed TO armory control by municipality
- Police chief loyalties (1991)
- Exact SDS/SDA organization strength ratings

❌ Not Included (Future Additions)

- Individual JNA unit types and equipment
- Detailed facility layouts
- Historical commander names
- Day-by-day garrison changes

Testing Scenarios

Scenario 1: Posavina Corridor Integrity

```javascript
const corridorStatus = checkCorridorIntegrity(gameState, 'posavina');
console.log('Posavina Corridor Status:', corridorStatus);

// Expected: Should show municipalities Gradiška → Derventa → Doboj
// Test: Flip one municipality and check integrity again
```

Scenario 2: Prijedor Flip Risk

```javascript
const munId = 10855; // Prijedor
console.log('Prijedor Analysis:');
console.log('  Has JNA:', hasJNAGarrison(munId));
console.log('  JNA Size:', getJNAGarrisonSize(munId));
console.log('  Adjacents:', getAdjacentMunicipalities(munId));
console.log('  Pressure:', calculateGeographicPressure(gameState, munId));
console.log('  Stability:', gameState.municipalities[munId].stability);

// Expected: JNA garrison 3000 troops, high flip risk despite SDA election win
```

Scenario 3: Drina Valley Cascade

```javascript
// Simulate Zvornik falling to RS
gameState.municipalities[11134].effectiveControl = 'rs';
gameState.municipalities[11134].hasFlipped = true;

// Check cascade pressure on Vlasenica
const cascadePressure = calculateCascadeFlipPressure(gameState, 11029, 'rs');
console.log('Vlasenica cascade pressure:', cascadePressure);

// Expected: High pressure, should trigger flip risk warning
```

Scenario 4: Sarajevo Siege Setup

```javascript
// Check Sarajevo surroundings
const sarajevoId = 10129;
const adjacents = getAdjacentMunicipalities(sarajevoId);

console.log('Sarajevo Adjacents:');
adjacents.forEach(adjId => {
    const mun = gameState.municipalities[adjId];
    console.log(`  ${mun.name}: ${mun.effectiveControl}`);
});

// Expected: Pale, Sokolac (RS), others (RBiH/Contested)
```

Key Municipality IDs (Quick Reference)

Critical Strategic Locations

```javascript
const CRITICAL_LOCATIONS = {
  // Capitals
  sarajevo: 10129,
  banjaLuka: 10812,
  pale: 10161, // RS wartime capital

  // Major cities
  tuzla: 11185,
  mostar: 10413,
  bihac: 10227,

  // Posavina Corridor
  gradiska: 10863,
  derventa: 10820,
  doboj: 11037,
  brcko: 11045, // Odžak area

  // Drina Valley
  zvornik: 11134,
  srebrenica: 10927,

  // Other strategic
  prijedor: 10855,
  hanPijesak: 10919, // VRS Main Staff
};
```

Performance Characteristics

Fast Operations (O(1) - constant time)

- `getAdjacentMunicipalities()` - Direct lookup
- `hasJNAGarrison()` - Direct lookup
- `getStrategicImportance()` - Direct lookup
- `isOnStrategicCorridor()` - Simple check

Medium Operations (O(n) - linear)

- `calculateGeographicPressure()` - Loops through adjacents (typically 3-8)
- `getAdjacentMunicipalitiesByFaction()` - Filters adjacents
- `checkCorridorIntegrity()` - Loops through corridor municipalities (5-10)
- `calculateEnhancedControlStability()` - Multiple calculations

Slower Operations (O(n*m) - nested)

- `getCorridorWarnings()` - Checks all corridors
- `identifyStrategicChokepoints()` - Analyzes all corridors
- `generateMunicipalitySitRep()` - Comprehensive analysis

Recommendation: Cache results when calling multiple times per frame. Recalculate only when municipality control changes.

Common Use Cases

Use Case 1: Municipality Detail Panel

```javascript
```

```javascript
function renderMunicipalityDetail(munId) {
    const mun = gameState.municipalities[munId];

    // Basic info
    let html = `<h2>${mun.name}</h2>`;
    html += `<p>Control: ${mun.effectiveControl}</p>`;
    html += `<p>Stability: ${mun.stability}/100</p>`;

    // Strategic importance
    const importance = getStrategicImportance(munId);
    const icon = getStrategicImportanceIcon(munId);
    html += `<p>Importance: ${icon} ${importance.toUpperCase()}</p>`;

    // JNA garrison
    if (hasJNAGarrison(munId)) {
        const size = getJNAGarrisonSize(munId);
        const strength = getJNAGarrisonStrength(munId);
        html += `<p class="warning">⚠ JNA GARRISON: ${strength} (${size} troops)</p>`;
    }

    // Corridors
    const corridors = getCorridorsForMunicipality(munId);
    if (corridors.length > 0) {
        html += `<div class="corridors">`;
        corridors.forEach(c => {
            html += `<p>📍 ${c.name}</p>`;
        });
        html += `</div>`;
    }

    // Adjacents
    const adjacents = getAdjacentMunicipalities(munId);
    const friendly = getAdjacentMunicipalitiesByFaction(gameState, munId, mun.effectiveControl);
    html += `<p>Adjacent: ${adjacents.length} (${friendly.length} friendly)</p>`;

    return html;
}
```

Use Case 2: Flip Risk Warning System

javascript

```
function checkFlipRisks() {
    const warnings = [];

    Object.entries(gameState.municipalities).forEach(([id, mun]) => {
        // Skip if already belongs to faction
        if (mun.effectiveControl === gameState.currentEntity) return;

        const stability = mun.stability;
        const geoPressure = calculateGeographicPressure(gameState, id);
        const cascadePressure = calculateCascadeFlipPressure(gameState, id, gameState.currentEntity);

        // High risk conditions
        if (stability < 40 && geoPressure > 60) {
            warnings.push({
                municipality: mun.name,
                id: id,
                risk: 'HIGH',
                stability: stability,
                pressure: geoPressure
            });
        } else if (cascadePressure > 80) {
            warnings.push({
                municipality: mun.name,
                id: id,
                risk: 'CASCADE',
                stability: stability,
                cascade: cascadePressure
            });
        }
    });

    return warnings;
}
```

Use Case 3: Situation Report

javascript

```
function generateFactionsRep() {
    console.log("===== STRATEGIC SITUATION =====\n");

    // Check critical corridors
    const posavina = checkCorridorIntegrity(gameState, 'posavina');
    console.log("Posavina Corridor:", posavina.intact ? "INTACT" : "⚠ DISRUPTED");
    console.log(`  ${posavina.secureCount}/${posavina.municipalityCount} secure\n`);

    // Check JNA garrisons
    const rsJNA = [10812, 10863, 10820, 11037, 10855].filter(id =>
        gameState.municipalities[id].effectiveControl === 'rs'
    );
    console.log(`RS controls ${rsJNA.length}/5 major JNA garrisons\n`);

    // Check flip risks
    const flipRisks = checkFlipRisks();
    console.log(`Flip risk warnings: ${flipRisks.length}`);
    flipRisks.forEach(w => {
        console.log(`  ⚠ ${w.municipality}: ${w.risk}`);
    });
}
```

Future Enhancements

Priority 1 (High Impact)

- [ ] TO armory control by municipality (affects initial unit strength)
- [ ] Police chief loyalties (affects control stability)
- [ ] Road quality ratings (affects movement speed)
- [ ] Railway network data (strategic movement)

Priority 2 (Medium Impact)

- [ ] Detailed SDS/SDA organization ratings per municipality
- [ ] Historical commander assignments
- [ ] Economic production values
- [ ] Urban vs. rural classification

Priority 3 (Nice to Have)

- [ ] Individual JNA unit types (infantry, armor, artillery)

- [ ] Facility details (barracks, depots, command centers)
- [ ] Historical timeline events by location
- [ ] Refugee flow patterns

Contribution Guidelines

If adding or modifying historical data:

1. **Source Your Data**: Include references to historical sources
2. **Follow Format**: Match existing data structure exactly
3. **Test Changes**: Verify no breaking changes to existing queries
4. **Document**: Update QUICK_REFERENCE.md if adding new functions
5. **Accuracy**: Prioritize historical accuracy over game balance

---

License & Attribution

This historical data is compiled from various public historical sources about the Bosnian War (1992-1995). It is provided for educational and game simulation purposes.

Sources consulted:

- CIA Maps of Bosnia and Herzegovina (1992-1995)
- ICTY trial documents and evidence
- Academic research on the Bosnian War
- Contemporary news reports
- Official JNA documents (public domain)

Note: Some data points (exact troop counts, minor garrisons) are estimates based on available evidence. The game prioritizes strategic accuracy over exact numerical precision.

Support & Questions

For questions about this historical data package:

1. Check **INTEGRATION_GUIDE.md** for implementation questions
2. Check **QUICK_REFERENCE.md** for function usage
3. Check **GEOGRAPHIC_REFERENCE.md** for strategic context

4. Review source code comments for detailed explanations

Version History

v1.0 (Current)

- Initial release
- 109 municipality adjacencies
- 24 JNA garrison locations
- 8 strategic corridors
- Complete function library
- Comprehensive documentation

Planned Updates:

- v1.1: TO armory control data
- v1.2: Police chief loyalties
- v1.3: Enhanced organization ratings
- v2.0: Dynamic historical events system

File Size Reference

```
historical_data.js              ~40 KB
historical_data_integration.js  ~30 KB
INTEGRATION_GUIDE.md            ~25 KB
QUICK_REFERENCE.md              ~35 KB
GEOGRAPHIC_REFERENCE.md         ~30 KB
README.md                       ~15 KB
----------------------------------------
TOTAL PACKAGE SIZE:             ~175 KB
```

Note: All files are heavily commented for learning and maintainability. Minified versions could be ~50% smaller if needed.

Getting Help

Can't find adjacency data? → Check MUNICIPALITY_ADJACENCIES in historical_data.js

Function not working? → See QUICK_REFERENCE.md for correct usage and examples

Don't understand strategic situation? → Read GEOGRAPHIC_REFERENCE.md for context

Integration issues? → Follow step-by-step INTEGRATION_GUIDE.md

Need to debug? → Use debugging helpers in QUICK_REFERENCE.md

Generated for A War Without Victory - Bosnia 1990-1995 Strategic Wargame Historical Data Package
Version 1.0 - January 2026