Implementation Guide v2.0 - Snake Evolution

Development Setup & Development Workflow

Redatto da: Senior Software Architect
Data: Novembre 2025
Versione: 2.0 (Adattato da Tic-Tac-Toe)
Status: Production Ready

Executive Summary

Questa guida fornisce **step-by-step instructions** per setup ambiente, configurazione, development workflow, testing, debugging, e deployment di Snake Evolution v2.0. È il documento di riferimento per sviluppatori che iniziano il progetto.

1. Prerequisiti e Setup Iniziale

1.1 Requisiti di Sistema

Requisito	Minimo	Consigliato
OS	Windows 10, macOS 10.15, Ubuntu 20.04	Latest LTS
RAM	4GB	8GB+
Disk	2GB libero	10GB+
Node.js	16.x	18.x LTS
npm	8.x	9.x+

1.2 Verifica Prerequisiti

```
# Verifica Node.js
node --version
# Output: v18.x.x o superiore

# Verifica npm
npm --version
# Output: npm 9.x.x o superiore

# Verifica git
git --version
# Output: git version 2.x.x
```

2. Clonazione Repository & Setup

2.1 Clone Repository

```
# Clone from GitHub
git clone https://github.com/yourusername/snake-evolution.git
cd snake-evolution

# Verifica branch
git branch -a
# Output: * main
#           develop
```

2.2 Setup Node.js Environment

Windows:

```
# Create virtual environment
npm install

# Install all dependencies
npm install

# Verify installation
npm list
```

macOS/Linux:

```
# Install dependencies
npm install

# Make scripts executable
chmod +x scripts/*.sh

# Verify installation
npm ls
```

2.3 Project Structure

```
snake-evolution/
├── src/
│   ├── core/
│   │   ├── GameEngine.js
│   │   ├── GameLoop.js
│   │   └── Constants.js
│   ├── entities/
│   │   ├── Snake.js
│   │   ├── Food.js
│   │   └── Grid.js
│   ├── systems/
│   │   ├── EvolutionSystem.js
│   │   ├── CollisionSystem.js
│   │   ├── InputManager.js
│   │   └── AudioManager.js
│   ├── ui/
│   │   ├── SceneManager.js
│   │   ├── UIRenderer.js
│   │   └── scenes/
│   │       ├── MainMenuScene.js
│   │       ├── GameScene.js
│   │       ├── GameOverScene.js
│   │       └── LeaderboardScene.js
│   ├── storage/
│   │   ├── StorageManager.js
│   │   └── HighScoreRepository.js
│   ├── events/
│   │   └── EventBus.js
│   ├── utils/
│   │   ├── Logger.js
│   │   ├── Validators.js
│   │   └── Profiler.js
│   ├── index.html
│   ├── style.css
│   └── main.js
├── tests/
│   ├── unit/
│   │   ├── collision.test.js
│   │   ├── evolution.test.js
│   │   ├── input.test.js
│   │   └── storage.test.js
│   ├── integration/
│   │   └── game-flow.test.js
│   └── chaos/
│       └── chaos-scenarios.test.js
├── config/
│   ├── development.yaml
│   ├── test.yaml
│   └── production.yaml
├── scripts/
│   ├── build.js
│   ├── profile.js
│   └── deploy.sh
├── package.json
├── package-lock.json
├── webpack.config.js
├── .eslintrc.json
├── .prettierrc.json
├── jest.config.js
├── .github/
│   └── workflows/
│       └── ci-cd.yml
└── README.md
```

3. Development Workflow

3.1 Common Commands

```
# Start development server
npm run dev
# Output: Webpack dev server running at http://localhost:8080

# Run tests
npm test
# Output: PASS  tests/unit/collision.test.js
#         PASS  tests/integration/game-flow.test.js
#         ✓ Test Suites: 5 passed
#         ✓ Coverage: 87%

# Run tests with coverage
npm test -- --coverage

# Watch tests during development
npm test -- --watch

# Lint code
npm run lint

# Fix linting issues
npm run lint:fix

# Format code
npm run format

# Build production bundle
npm run build
# Output: dist/bundle.abc123.js (512 KB)
#         dist/style.def456.css (45 KB)

# Analyze bundle
npm run analyze

# Profile performance
npm run profile

# Deploy to production
npm run deploy
```

3.2 Development Workflow Steps

Step 1: Create feature branch

```
git checkout -b feature/evolution-animation
```

Step 2: Implement feature

```
# Start dev server with hot reload
npm run dev

# Make code changes in src/
# Tests run automatically on save
```

Step 3: Run linting & tests

```
npm run lint:fix
npm test -- --coverage
```

Step 4: Commit changes

```
git add .
git commit -m "feat: add evolution animation with VFX"
# Pre-commit hooks run automatically
```

Step 5: Push & create PR

```
git push origin feature/evolution-animation
# Create Pull Request on GitHub
# CI/CD pipeline runs automatically
```

Step 6: Merge after review

```
# After approval &amp; CI passes
git checkout main
git pull origin main
git merge feature/evolution-animation
git push origin main
# Automatic deployment triggered
```

4. Testing Workflow

4.1 Running Tests

```
# Run all tests once
npm test

# Run tests in watch mode
npm test -- --watch

# Run specific test file
npm test collision.test.js

# Run tests matching pattern
npm test -- -t "collision"
```

```
# Generate coverage report
npm test -- --coverage

# Generate HTML coverage report
npm test -- --coverage --coverageReporters=html
# Open coverage/index.html in browser
```

4.2 Writing Tests

File: `tests/unit/collision.test.js`

```
describe('CollisionDetector', () =&gt; {
  let detector;

  beforeEach(() =&gt; {
    detector = new CollisionDetector();
  });

  test('should detect wall collision at x=0', () =&gt; {
    const head = { x: -1, y: 10 };
    expect(detector.checkWallCollision(head)).toBe(true);
  });

  test('should detect self-collision with spatial hash', () =&gt; {
    const segments = [
      { x: 10, y: 10 }, // Head
      { x: 9, y: 10 },
      { x: 8, y: 10 },
      { x: 7, y: 10 },
      { x: 6, y: 10 },
      { x: 10, y: 10 } // Body at head position
    ];

    expect(detector.checkSelfCollision(segments)).toBe(true);
  });
});
```

4.3 Property-Based Testing

```
// Use fast-check for property-based tests
import fc from 'fast-check';

test('Evolution stages are deterministic', () =&gt; {
  fc.assert(
    fc.property(fc.integer({ min: 0, max: 200 }), (length) =&gt; {
      const stage1 = evolution.getStageByLength(length);
      const stage2 = evolution.getStageByLength(length);
      return stage1 === stage2;
    })
  );
});
```

5. Debugging

5.1 Browser DevTools

```
// Enable detailed logging
localStorage.setItem('LOG_LEVEL', 'DEBUG');

// Reload to see debug logs
window.location.reload();

// Access game engine in console
window.gameEngine.getGameState()
window.gameEngine.getSnake()
window.gameEngine.getScore()
```

5.2 Debug Breakpoints

```
// In code, set breakpoint
debugger;

// Run with dev server
npm run dev

// Open DevTools (F12), Reload page
// Execution pauses at breakpoint
```

5.3 Performance Profiling

```
// Enable performance profiler
const profiler = new PerformanceProfiler();

// Measure sections
profiler.measureSection('collision-check', () =&gt; {
  // code to measure
});

// Get report
console.log(profiler.getReport());
```

6. Building & Bundling

6.1 Development Build

```
npm run build:dev

# Output:
# Webpack 5.x
# dist/bundle.js (non-minified)
# dist/style.css
# Ready for development
```

6.2 Production Build

```
npm run build

# Output:
# Webpack 5.x (optimized)
# dist/bundle.abc123xyz.js (minified, 512KB)
# dist/style.def456uvw.css (minified, 45KB)
# dist/index.html
# Ready for deployment
```

6.3 Bundle Analysis

```
npm run analyze

# Opens interactive bundle analyzer
# Shows composition of bundle
# Identifies large dependencies
```

7. Configuration Management

7.1 Environment Variables

File: `.env.local` **(for development)**

```
APP_ENV=development
LOG_LEVEL=DEBUG
AUDIO_VOLUME=0.8
CANVAS_WIDTH=500
CANVAS_HEIGHT=500
```

File: `.env.production`

```
APP_ENV=production
LOG_LEVEL=INFO
AUDIO_VOLUME=0.8
```

7.2 Loading Configuration

```
// In src/config/index.js
const config = {
  app: {
    env: process.env.APP_ENV || 'development',
    logLevel: process.env.LOG_LEVEL || 'INFO'
  },
  game: {
    gridSize: 20,
    cellSize: 25,
    targetFPS: 60
  }
};

export default config;
```

8. Deployment

8.1 Deploy to Netlify

```
# Build production bundle
npm run build

# Install Netlify CLI
npm install -g netlify-cli

# Deploy
netlify deploy --prod --dir=dist

# Output:
# ✓ Site deployed to https://snake-evolution.netlify.app
```

8.2 Deploy to GitHub Pages

```
# Configure package.json
"homepage": "https://yourusername.github.io/snake-evolution"

# Build &amp; deploy
npm run build
npm run deploy

# Output:
# ✓ Published to https://yourusername.github.io/snake-evolution
```

9. Troubleshooting

Problem	Cause	Solution
npm install fails	Node version mismatch	Use `nvm use 18`
Tests fail locally	Missing dependencies	Run `npm install` again
Dev server won't start	Port 8080 in use	Change port: `npm run dev -- --port 3000`
Build errors	Webpack config issue	Check `webpack.config.js`
Linting errors	ESLint rules violated	Run `npm run lint:fix`

10. Quick Reference

Common Issues & Solutions

```
# Clear npm cache
npm cache clean --force
npm install

# Force reinstall dependencies
rm -rf node_modules package-lock.json
npm install

# Update all dependencies
npm update

# Check for outdated packages
npm outdated

# Audit for security vulnerabilities
npm audit
npm audit fix
```

Implementation Guide v2.0 - Production Ready
Data: Novembre 2025