**Key Files and Architecture Explanation**

**Core Architecture Pattern: MVC + Component System**

**Models Layer (/src/models/)**

**Pure data and business logic - no UI dependencies**

**Game.js - Central State Manager**

class Game {

constructor() {

this.gold = 0;

this.reputation = 0;

this.currentLocation = null;

this.activePig = null;

this.retiredPigs = [];

this.achievements = [];

}

// Core game logic methods

update(deltaTime) { }

save() { }

load() { }

}

**Pig.js - Pig Entity**

class Pig {

constructor(color) {

this.color = color;

this.sniffingLevel = 1;

this.luckLevel = 1;

this.speedLevel = 1;

this.trufflesFound = 0;

this.legendariesFound = 0;

}

getSuccessRate() { }

canRetire() { }

retire() { }

}

**Controllers Layer (/src/controllers/)**

**Coordinate between models and views**

**GameController.js - Main Game Loop**

class GameController {

constructor(game, gameView) {

this.game = game;

this.gameView = gameView;

this.isRunning = false;

}

start() { }

update(deltaTime) { }

handleClick(x, y) { }

}

**TruffleController.js - Truffle Logic**

class TruffleController {

constructor(game) {

this.game = game;

this.spawnTimer = 0;

this.activeTruffles = [];

}

update(deltaTime) { }

spawnTruffle() { }

collectTruffle(truffle) { }

}

**Views Layer (/src/views/)**

**UI components and rendering**

**GameView.js - Main View Coordinator**

class GameView {

constructor(container) {

this.container = container;

this.pigRenderer = new PigRenderer();

this.forestRenderer = new ForestRenderer();

this.uiManager = new UIManager();

}

render(gameState) { }

update(deltaTime) { }

}

**Component Examples:**

**PigRenderer.js** - Handles pig animations

class PigRenderer {

constructor() {

this.pigElement = null;

this.walkAnimation = null;

this.digAnimation = null;

}

createPig(color) { }

startWalking() { }

startDigging() { }

playFoundAnimation() { }

}

**UIPanel.js** - Base UI Component

class UIPanel {

constructor(selector) {

this.element = document.querySelector(selector);

this.isVisible = false;

}

show(animated = true) { }

hide(animated = true) { }

update(data) { }

}

**CSS Architecture**

**main.css - CSS Variables and Base Styles**

:root {

--primary-color: #8B4513;

--secondary-color: #228B22;

--gold-color: #FFD700;

--font-family: 'Game Font', sans-serif;

--border-radius: 8px;

--transition-speed: 0.3s;

}

\* {

box-sizing: border-box;

}

body {

font-family: var(--font-family);

margin: 0;

padding: 0;

}

**Component CSS Example: pig.css**

.pig {

position: absolute;

width: 64px;

height: 64px;

background-image: url('../assets/images/pigs/pig-idle.png');

transition: transform var(--transition-speed);

}

.pig--walking {

animation: pig-walk 1s infinite;

}

.pig--digging {

animation: pig-dig 0.5s ease-in-out;

}

@keyframes pig-walk {

0%, 100% { transform: translateY(0); }

50% { transform: translateY(-4px); }

}

@keyframes pig-dig {

0% { transform: scale(1); }

50% { transform: scale(1.1) rotate(5deg); }

100% { transform: scale(1); }

}

**Configuration-Driven Design**

**GameConfig.js - Centralized Balance**

export const GAME\_CONFIG = {

TRUFFLE\_VALUES: {

'Button Mushrooms': 3,

'Summer Truffles': 18,

// ... rest of truffle values

},

SPAWN\_RATES: {

'Beginner Forest': {

'Button Mushrooms': 0.40,

'Summer Truffles': 0.35,

// ... rest of spawn rates

}

},

UPGRADE\_COSTS: {

sniffing: [0, 50, 125, 300, /\* ... \*/],

speed: [0, 75, 188, 469, /\* ... \*/]

}

};

**Rapid Development Features**

**Prebuilt Components (/lib/custom-components/)**

**NumberCounter.js - Animated Number Display**

class NumberCounter {

constructor(element, options = {}) {

this.element = element;

this.duration = options.duration || 1000;

this.formatter = options.formatter || (n => n.toLocaleString());

}

animateTo(targetValue) {

gsap.to(this, {

duration: this.duration / 1000,

currentValue: targetValue,

onUpdate: () => {

this.element.textContent = this.formatter(Math.floor(this.currentValue));

}

});

}

}

**ParticleSystem.js - Visual Effects**

class ParticleSystem {

constructor(container) {

this.container = container;

this.particles = [];

}

createTruffleFoundEffect(x, y, truffleType) {

// Create sparkle particles using GSAP

const colors = this.getColorsForTruffle(truffleType);

for (let i = 0; i < 10; i++) {

this.createParticle(x, y, colors[i % colors.length]);

}

}

}

**File Loading Strategy**

**main.js - Application Bootstrap**

import { Game } from './models/Game.js';

import { GameController } from './controllers/GameController.js';

import { GameView } from './views/GameView.js';

import { GAME\_CONFIG } from './config/GameConfig.js';

class TruffleHunterTycoon {

constructor() {

this.game = new Game(GAME\_CONFIG);

this.gameView = new GameView(document.getElementById('game-container'));

this.gameController = new GameController(this.game, this.gameView);

}

async init() {

await this.loadAssets();

this.gameController.start();

}

}

// Initialize game when DOM is ready

document.addEventListener('DOMContentLoaded', () => {

const game = new TruffleHunterTycoon();

game.init();

});

**Benefits of This Structure**

1. **Maintainable**: Each class has a single responsibility
2. **Testable**: Pure functions and clear dependencies
3. **Scalable**: Easy to add new features without refactoring
4. **Rapid Development**: Reusable components and configuration-driven
5. **Performance**: Efficient CSS and JavaScript organization
6. **Team-Friendly**: Clear file organization for multiple developers

This structure follows modern web development best practices while maintaining simplicity for rapid prototyping and iteration.