Subway Surfer Game: CS 1430 Final Project Report

Team Members: Tito Leony-bujanda, Christopher Duong, Samuel Koehnlein, Joshua Mukisa, Jungdo Kim

1. Game Mechanics

Our Subway Surfer-style endless runner implements a vertical-scrolling system where players navigate through three lanes, avoiding obstacles and collecting items. The movement system allows instantaneous lane switching (positions at 225, 475, 725 pixels) and physics-based jumping with realistic gravity (`GRAVITY = 0.6f`) and initial velocity (`JUMP_VELOCITY = -12.0f`).

Obstacle System: Three obstacle types populate the lanes—red barriers (120×50px), large blue trains (180×80px), and green signs (120×50px). Ten obstacles spawn procedurally and scroll at increasing speeds. Collision detection uses axis-aligned bounding box (AABB) intersection, checking if player bounds overlap with obstacles in the same lane.

Collectibles: Two types enhance gameplay—golden coins (10 points × combo multiplier) and pink hearts (extra lives, capped at 5). Collectibles feature sinusoidal bouncing animation and spawn with 50% probability relative to obstacles.

Scoring & Progression: Players earn 5 points (×combo) per obstacle passed and 10 points (×combo) per coin collected. Consecutive coin collection builds combo multipliers that decay after 3 seconds. Every 100 points advances the level, increasing scroll speed by 0.5 units (`scrollSpeed = BASE_SCROLL_SPEED + level × 0.5f`), creating exponential difficulty progression.

The game implements four states: START (instructions), PLAYING (active gameplay), PAUSED (frozen state), and GAME_OVER (final score), with smooth transitions preserving game state.

2. Story/Narrative

While simplified from commercial runners, our game establishes a clear urban subway environment. The player character, a stylized humanoid with animated features, navigates an endless subway system in a daring jailbreak and escape through underground transit to avoid getting caught.

Visual Storytelling: The narrative emerges through environmental details: dynamic cityscape backgrounds with lit windows, subway lane markings, and scrolling terrain creating rapid

movement illusion. The gradient sky transitions subtly with time-based color variation using sine wave functions.

Character Expression: The player demonstrates personality through animation—blinking eyes (8 Hz sinusoidal movement), smiling expression, animated swinging arms and legs, and a dynamic shadow scaling with jump height. These details ground the character in the environment and create emotional connection.

The implicit narrative goal follows the classic "test of skill" archetype—survive as long as possible while the increasing difficulty and combo system create satisfying progression within each session.

3. Aesthetics

Visual Design: Our aesthetic emphasizes clarity and readability using bold colors and strong contrasts for instant element recognition during high-speed gameplay. The carefully selected palette includes sky gradients (cyan to deep blue with dynamic time-based shifts), bright cyan player body with peachy skin tone, color-coded obstacles (red barriers, blue trains, green signs), and standout collectibles (golden coins, pink hearts). UI elements use dark boxes with cyan/orange text for high contrast readability.

Rendering Techniques: Advanced pixel-level rendering creates visual depth through vertical gradient interpolation simulating lighting, multiple simultaneous animation systems (physics-based particles, sinusoidal bouncing, procedural limb swinging), and parallax scrolling for buildings at 30% ground speed. Every action produces visual feedback: jumps spawn 10 blue particles, collisions create 30 explosive red particles, and collections trigger 15-20 colored sparkles.

Typography: A custom 5×7 bitmap font renders all text with pixel-perfect precision, supporting uppercase letters, numbers, and symbols with scaling (1-4×) and shadow rendering (2-pixel black offset) ensuring readability.

Audio: Background trap-style music complements the urban setting via SDL_mixer at 44.1kHz stereo, with volume balanced at 50% to maintain atmospheric presence without overwhelming gameplay.

4. Technology

Development Stack: C++11 with object-oriented design, SDL2 for graphics, SDL2_mixer for audio, GNU Make build system, configured for macOS/Homebrew with Linux/Windows adaptability.

Architecture: The codebase follows modular architecture across nine files (~1,200 lines) while using an object oriented approach:

- **SDL_Plotter:** Provided wrapper abstracting SDL2 complexity
- **SubwaySurferGame:** Central controller managing game logic, physics, entities, state machine, and progression
- **Renderer:** Pure rendering class with 15+ specialized drawing functions, text rendering, and visual effects
- **GameObjects:** Data structures for Obstacle, Collectible, and Particle entities
- **Constants:** Screen dimensions, physics parameters, lane positions, and font bitmap data

Key Algorithms:

- **Collision Detection:** AABB intersection checking player-obstacle overlap in matching lanes
- **Physics Integration:** Verlet method for velocity/position updates with gravity
- **Particle System:** Position/velocity updates with lifetime-based alpha fade
- **Procedural Spawning:** Probability-based generation (33% obstacles, 50% coins, 6.67% hearts)

Performance: Fixed 16ms timestep (62.5 FPS target), efficient dirty rectangle updates, bounds checking preventing off-screen rendering, and object pooling through respawning. Cross-platform compatibility maintained through SDL2 abstraction.

5. Goals

Player Objectives: The primary goal is surviving as long as possible while maximizing score—an open-ended objective creating unique session stories. Secondary goals include reaching higher levels (every 100 points), building combo multipliers, collecting coin sequences, and maintaining full lives as personal challenge.

Learning Curve: Progressive skill development occurs naturally—minutes 1-2 teach basic controls, minutes 3-5 develop jump/switch timing mastery, and minutes 5+ enable combo optimization and spawn prediction.

Feedback Systems: Goals are reinforced through real-time HUD (score, lives, level), visual combo multiplier with countdown timer, particle effects celebrating achievements, and progressive difficulty providing constant challenge. While endless runners lack traditional "winning," players set personal benchmarks—first 100-point milestone, breaking high scores, achieving high multipliers, or reaching intense Level 5+.

6. Team Work

Tito - Programming Quality Assurance: Conducted systematic bug testing across scenarios including edge cases, code review for logical errors and standards adherence, integration testing ensuring smooth feature compatibility, performance profiling during intensive scenes, and documentation verification. Prevented critical issues including collision detection false positives and particle system memory leaks.

Christopher Duong - Project & File Management: Established Git repository with branching strategy, designed and maintained cross-platform Makefile, architected modular file structure separating Game/Renderer/GameObjects/Constants, managed assets pipeline and dependency coordination, and created comprehensive README with build instructions. His organizational excellence prevented merge conflicts and build issues.

Samuel - Audio Engineer: Researched and selected appropriate background music matching the urban aesthetic, implemented SDL_mixer initialization and music loading, verified audio across platforms, calibrated volume balancing (50%), ensured lag-free audio through proper threading, and documented expansion points for future sound effects.

Joshua Mukisa - Graphics & User Interface Designer: Selected harmonious color schemes ensuring aesthetic appeal and functional clarity, designed player character appearance and animations (limb swing frequency, eye blinking), designed HUD/screen layouts with optimal information hierarchy, conceptualized particle behaviors for events, developed gradient rendering aesthetic, and collaborated on bitmap font design ensuring readability.

JD - Secondary Programmer: Developed physics implementation with realistic acceleration/velocity, implemented AABB collision detection algorithm, programmed combo multiplier system with timer-based decay, created particle spawning and simulation logic, implemented game state machine and transitions, designed object pooling for efficient memory, coded sinusoidal animations, and added comprehensive code documentation.

Collaborative Workflow: Weekly meetings maintained alignment, pair programming sessions combined design with implementation expertise, code review ensured quality standards, and iterative development enabled rapid feedback incorporation. The team overcame platform compatibility challenges, performance optimization needs, visual cohesion requirements, and audio sync issues through coordinated effort.

7. Reflection

Development Journey: Our six-week project progressed through planning/architecture (Week 1), core systems implementation (Weeks 2-3), feature expansion (Weeks 4-5), and polish/testing (Week 6). Each phase presented unique challenges—from establishing solid modular structure to achieving pixel-perfect collision accuracy through three major revisions, to balancing particle visual impact with performance.

Technical Insights: Working directly with SDL2 pixel buffers taught low-level graphics concepts. Object-oriented design demonstrated separation benefits—rendering changes required no game logic modifications. Implementing physics from first principles reinforced calculus concepts, while tuning values required iterative experimentation for satisfying "game feel." The state machine taught finite state concepts applicable beyond games.

Challenges: Cross-platform SDL2 differences created unexpected audio path and compilation hurdles. Early collision code produced false positives requiring mathematical rigor. Particle systems initially tanked frame rates until implementing lifetime culling and reduced spawn rates

(100→10-30 particles). With five people modifying interconnected code, clear file ownership and communication protocols minimized merge conflicts. Recognizing timeline constraints, we focused on polishing core mechanics rather than half-implementing ambitious features like power-ups.

Successes: Our modular architecture proved exceptionally maintainable—adding obstacle types required modifying only two files. The custom font, gradients, and particles created distinctive aesthetics exceeding expectations. Achieving 60+ FPS validated optimization efforts. We delivered every planned core feature with complete UI and comprehensive documentation.

Personal Growth: Each member expanded skills significantly—Tito developed systematic testing methodologies, Christopher mastered build systems and version control, Samuel gained audio programming experience, Joshua refined digital design skills, and JD deepened C++ proficiency with game development patterns.

Key Lessons: Architectural planning time pays exponential dividends. Continuous testing prevents compounding bugs. Proactive communication prevents duplicated effort. First implementations are rarely optimal—embrace iteration. Document as you go rather than retrospectively.

Future Enhancements: Given more time, we envision sound effects for actions, temporary power-ups (invincibility, coin magnets), visual variety (multiple backgrounds, weather effects), algorithm-driven procedural obstacle patterns, local leaderboard system, and mobile touch control adaptation.

Conclusion: This project successfully demonstrates practical CS concepts—data structures, algorithms, OOP, graphics programming, and collaborative development. Beyond technical achievement, we gained invaluable team coordination and project management experience. Most importantly, we created something genuinely enjoyable—the ultimate game development validation. From Christopher's file organization to Tito's testing, Samuel's audio to Joshua's visuals, and JD's architecture—every element contributed to a polished product exceeding the sum of its parts. We emerged with deepened skills, confidence for ambitious challenges, and a complete game representing our growth as computer scientists.