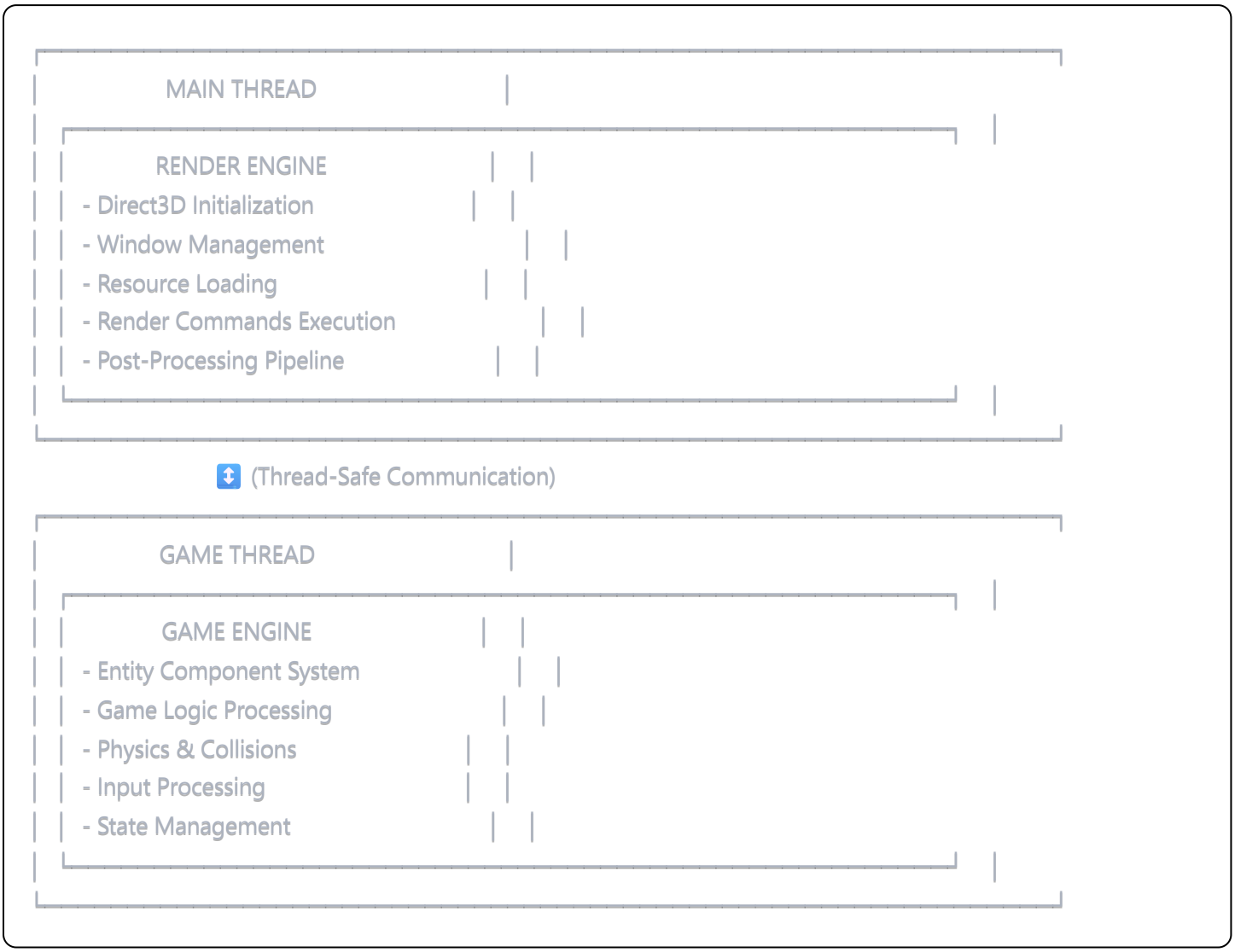
TRON 3D Game Engine Structure

Core Architecture Overview



Directory Structure

```
TRON_Engine/
├── Core/
│   ├── Engine.h/.cpp                 // Main engine class
│   ├── Application.h/.cpp            // Application lifecycle
│   ├── Timer.h/.cpp                  // Time management system
│   └── Threading/
│       ├── ThreadPool.h/.cpp
│       ├── RenderThread.h/.cpp
│       └── GameThread.h/.cpp
│
├── Rendering/                        // MAIN THREAD COMPONENTS
│   ├── RenderEngine.h/.cpp           // Main render coordinator
│   ├── D3D/
│   │   ├── D3DContext.h/.cpp         // Direct3D setup
│   │   ├── D3DRenderer.h/.cpp        // Core D3D rendering
│   │   ├── SwapChain.h/.cpp
│   │   └── CommandQueue.h/.cpp
│   ├── Resources/
│   │   ├── MeshManager.h/.cpp
│   │   ├── TextureManager.h/.cpp
│   │   ├── ShaderManager.h/.cpp
│   │   └── MaterialManager.h/.cpp
│   ├── Shaders/
│   │   ├── BasicShader.hlsl
│   │   ├── PostProcessShader.hlsl
│   │   └── ParticleShader.hlsl
│   ├── Camera/
│   │   ├── Camera.h/.cpp
│   │   └── CameraManager.h/.cpp
│   ├── Culling/
│   │   ├── FrustumCuller.h/.cpp
│   │   └── OcclusionCuller.h/.cpp
│   └── PostProcessing/
│       ├── PostProcessor.h/.cpp
│       └── Effects/
│           ├── SaturationEffect.h/.cpp
│           ├── ContrastEffect.h/.cpp
│           └── BrightnessEffect.h/.cpp
│
├── Game/                             // GAME THREAD COMPONENTS
│   ├── GameEngine.h/.cpp             // Main game coordinator
│   ├── ECS/
│   │   ├── Entity.h/.cpp
│   │   ├── Component.h
│   │   ├── ComponentManager.h/.cpp
│   │   ├── System.h
```

```
│   │   └── SystemManager.h/.cpp
│   ├── Components/
│   │   ├── Transform.h/.cpp          // Position, rotation, scale
│   │   ├── Mesh.h/.cpp               // Geometry data
│   │   ├── Collider.h/.cpp           // Collision shapes
│   │   ├── Rigidbody.h/.cpp          // Physics properties
│   │   ├── Controller.h/.cpp         // Input handling
│   │   ├── UI.h/.cpp                 // UI elements
│   │   └── Gameplay/
│   │       ├── Health.h/.cpp
│   │       ├── Weapon.h/.cpp
│   │       └── Enemy.h/.cpp
│   ├── Systems/
│   │   ├── TransformSystem.h/.cpp    // Handle transforms
│   │   ├── PhysicsSystem.h/.cpp      // Physics updates
│   │   ├── CollisionSystem.h/.cpp    // Collision detection
│   │   ├── InputSystem.h/.cpp        // Input processing
│   │   ├── GameplaySystem.h/.cpp     // Game-specific logic
│   │   └── ParticleSystem.h/.cpp     // Particle updates
│   ├── Physics/
│   │   ├── CollisionDetector.h/.cpp
│   │   ├── SpatialPartitioning.h/.cpp  // Octree/Grid for optimization
│   │   └── CollisionResolver.h/.cpp
│   ├── Input/
│   │   ├── InputManager.h/.cpp
│   │   ├── Keyboard.h/.cpp
│   │   └── Mouse.h/.cpp
│   └── States/
│       ├── StateManager.h/.cpp
│       ├── GameState.h/.cpp
│       ├── MenuState.h/.cpp
│       └── SplashState.h/.cpp
│
├── Communication/                  // THREAD SYNCHRONIZATION
│   ├── RenderQueue.h/.cpp          // Commands from Game to Render
│   ├── EventSystem.h/.cpp          // Event communication
│   ├── SharedData.h/.cpp           // Thread-safe shared data
│   └── Synchronization/
│       ├── Mutex.h/.cpp
│       ├── ConditionVariable.h/.cpp
│       └── AtomicOperations.h/.cpp
│
├── Math/
│   ├── Vector3.h/.cpp
│   ├── Matrix4.h/.cpp
│   ├── Quaternion.h/.cpp
│   ├── Transform.h/.cpp
```

```
│   └── MathUtils.h/.cpp
│
├── Utils/
│   ├── Logger.h/.cpp
│   ├── FileSystem.h/.cpp
│   ├── StringUtils.h/.cpp
│   └── ProceduralGeneration.h/.cpp      // For level generation
│
└── TRON_Game/                      // GAME IMPLEMENTATION
    ├── TronGame.h/.cpp              // Main game class
    ├── Levels/
    │   ├── ProceduralLevel.h/.cpp
    │   └── LevelGenerator.h/.cpp
    ├── Entities/
    │   ├── Player.h/.cpp
    │   ├── Enemy.h/.cpp
    │   ├── Projectile.h/.cpp
    │   └── Obstacle.h/.cpp
    ├── UI/
    │   ├── HUD.h/.cpp
    │   ├── Menu.h/.cpp
    │   └── ScoreDisplay.h/.cpp
    └── Gameplay/
        ├── GameRules.h/.cpp
        ├── ScoreSystem.h/.cpp
        └── SpawnManager.h/.cpp
```

Key Classes and Responsibilities

Main Thread (Rendering)

- **RenderEngine**: Coordinates all rendering operations
- **D3DRenderer**: Direct3D API calls and resource management
- **RenderQueue**: Processes render commands from game thread
- **ResourceManagers**: Load and manage GPU resources
- **PostProcessor**: Handle screen effects (saturation, contrast, brightness)

Game Thread (Logic)

- **GameEngine**: Coordinates all game systems
- **ECS Framework**: Entity-Component-System architecture
- **Systems**: Process component data each frame
- **StateManager**: Handle game states (menu, playing, paused)
- **PhysicsSystem**: Collision detection and response

Communication Layer

- **RenderQueue**: Thread-safe command queue for rendering instructions
- **EventSystem**: Publish/subscribe system for cross-thread communication
- **SharedData**: Thread-safe containers for shared state

Threading Model

Main Thread Flow:

1. Initialize Direct3D and window
2. Start game thread
3. **Render Loop**:
   - Process render commands from queue
   - Execute Direct3D rendering
   - Present frame
   - Handle Windows messages

Game Thread Flow:

1. Initialize game systems
2. **Game Loop**:
   - Process input
   - Update all systems
   - Generate render commands
   - Send commands to render queue
   - Sleep to maintain target framerate

Synchronization Points:

- **Frame Synchronization**: Game waits for render completion before next frame
- **Resource Loading**: Coordinated loading of meshes, textures, shaders
- **State Changes**: Menu transitions, level loading

Dynamic Library Structure

The engine compiles as a DLL with the following exports:

- `CreateEngine()` - Factory function
- `DestroyEngine()` - Cleanup function
- Engine interface for game implementation

This structure ensures clear separation between rendering and game logic while maintaining optimal performance through proper threading and minimal synchronization overhead.