Das Projekt ist eine objektorientierte Implementierung eines Pac-Man-Spiels in C++ unter Verwendung der Raylib-Bibliothek für die grafische Darstellung. Die Architektur ist modular aufgebaut und folgt dem Prinzip der Trennung von Verantwortlichkeiten. Die wichtigsten Komponenten sind in eigene Klassen und Dateien ausgelagert, was die Wartbarkeit, Erweiterbarkeit und Übersichtlichkeit des Codes fördert.

**Der Einstiegspunkt des Programms befindet sich in der Datei main.cpp.** Diese Datei übernimmt die wichtige Rolle, den Programmstart zu organisieren und grundlegende Initialisierungen vorzunehmen. Insbesondere wird hier der Zufallszahlengenerator initialisiert, was für die zufällige Labyrinth-Generierung sowie für andere Spielereignisse essentiell ist. Danach wird eine Instanz der zentralen Game-Klasse erzeugt. Durch den Aufruf der Methode run() auf dem Game-Objekt wird die Hauptspielschleife gestartet, die den gesamten Spielfluss steuert. Diese Struktur sorgt dafür, dass die komplexe Spiellogik aus der main.cpp ausgelagert wird und die Datei selbst sehr schlank und übersichtlich bleibt. Sie kümmert sich ausschließlich darum, das Spiel zu starten und den Programmfluss einzuleiten sowie das Spiel am Ende sauber zu beenden.

**Im Mittelpunkt des Spiels steht die Game-Klasse, die in Game.h und Game.cpp definiert ist.** Diese Klasse fungiert als zentraler Controller, der alle wesentlichen Spielzustände verwaltet und die Steuerung des Spiels übernimmt. Die Game-Klasse hält den aktuellen Zustand des Spiels in Form eines GameState-Enums vor, welches die verschiedenen Phasen wie Startmenü, Spielen, Pause, Game Over oder Bestenliste repräsentiert. Neben der Verwaltung des Spielzustands hält die Klasse auch Referenzen auf alle wichtigen Objekte, die für das Spiel benötigt werden. Dazu zählen das Spielfeld (repräsentiert durch die GameBoard-Klasse), der Spieler (Player-Klasse), die Geister (Ghost-Klasse) und die Bestenliste (Leaderboard-Klasse).

Beim Programmstart öffnet die Game-Klasse das Spielfenster und lädt sämtliche benötigten Ressourcen wie Grafiken und Sounds. Die Hauptspielschleife läuft in einer Schleife, die so lange aktiv ist, wie das Spielfenster geöffnet bleibt. Je nach aktuellem Zustand ruft die Game-Klasse unterschiedliche Methoden auf, um entsprechende Bildschirme anzuzeigen oder Aktionen durchzuführen. Beispielsweise wird im Startmenü auf Benutzereingaben gewartet, im Spielzustand wird die Spiellogik abgearbeitet und im Pausemodus können Spielpausen verwaltet werden. Während des Spielmodus (STATE\_PLAYING) werden Bewegung der Spielfiguren, Kollisionserkennung, Punktevergabe und das Spawnen von Früchten abgehandelt. Sobald das Spiel beendet ist, wird geprüft, ob ein neuer Highscore erreicht wurde. Dieser wird dann gegebenenfalls in die Bestenliste eingetragen. Am Ende des Spiels werden alle Ressourcen freigegeben, bevor das Spiel komplett beendet wird.

**Die Klasse GameBoard, implementiert in GameBoard.h und GameBoard.cpp, verwaltet das Spielfeld.** Das Spielfeld ist als zweidimensionales Array von Zellen aufgebaut, wobei jede Zelle eine Instanz der Klasse Cell darstellt. Die Hauptaufgabe der GameBoard-Klasse besteht darin, das Labyrinth zu generieren, die Positionen von Münzen, Wänden und Früchten zu bestimmen und deren Zustand während des Spiels zu verwalten. Die Labyrinth-Generierung erfolgt durch die Methode generateRandomMap(), die einen Tiefensuchalgorithmus (Depth-First Search, DFS) verwendet. Dieser Algorithmus durchläuft das Spielfeld rekursiv, um ein zufälliges und zugleich zusammenhängendes Labyrinth zu erstellen. Neben der Generierung des Labyrinths sorgt die Klasse auch dafür, dass Münzen und Früchte an passenden Stellen platziert werden. Die Startpositionen der Geister werden ebenfalls in der GameBoard-Klasse definiert, damit sie bei Spielbeginn korrekt gesetzt sind. Somit ist die Klasse die zentrale Komponente für die gesamte Spielfeldverwaltung und bildet die Grundlage für das Spielgeschehen.

**Während der Labyrinth-Generierung wird zusätzlich die Struktur MazeCell verwendet, die in MazeCell.h definiert ist.** Diese Struktur ist ein Hilfsobjekt, das nicht direkt Teil des finalen Spielfelds ist, sondern temporär verwendet wird. Sie speichert für jede Zelle, ob diese bereits vom Algorithmus besucht wurde und ob sie als Wand gilt. Die MazeCell-Struktur erleichtert die korrekte und effiziente Erzeugung des Labyrinths, indem sie während der Tiefensuche Zustände protokolliert.

**Globale Konstanten und wichtige Konfigurationswerte sind in der Datei Constants.h zusammengefasst.** Dort werden unter anderem die Spielfeldgröße, die Kachelgröße (Tile-Size) und die Definition der verschiedenen Zelltypen in Form eines Enums (WALL, EMPTY, COIN, FRUIT) hinterlegt. Diese zentralisierte Definition erlaubt eine einfache Anpassung des Spiels, ohne den Quellcode an vielen Stellen anpassen zu müssen. Gleichzeitig wird dadurch die Lesbarkeit erhöht, da feste Werte nicht hartkodiert an unterschiedlichen Stellen auftauchen.

**Die Bewegung von Spielfiguren wird über die abstrakte Basisklasse Entity realisiert, welche in Entity.h und Entity.cpp implementiert ist.** Diese Klasse definiert gemeinsame Eigenschaften aller beweglichen Spielfiguren, etwa die Position auf dem Spielfeld und die Bewegungsrichtung. Die Verwendung einer Basisklasse erlaubt es, sowohl den Spieler (Pac-Man) als auch die Geister einheitlich zu behandeln und erleichtert die spätere Erweiterbarkeit, falls weitere bewegliche Entitäten hinzugefügt werden sollen.

**Die Klasse Player erbt von Entity und repräsentiert Pac-Man, die Spielfigur, die der Spieler steuert.** Player verwaltet neben der Bewegung auch den Punktestand und die Spiellogik für das Einsammeln von Münzen und Früchten. Ebenso kümmert sie sich um die Kollisionsprüfung mit den Geistern, da eine solche Kollision das Spielende auslöst. Damit bündelt die Player-Klasse alle Funktionen, die für das Verhalten des Spielers notwendig sind.

**Die Geister werden durch die Klasse Ghost dargestellt, die ebenfalls von Entity erbt.** Die Ghost-Klasse implementiert die KI-Logik für die Bewegung der Geister, welche sich an Pac-Man orientieren, um ihn zu verfolgen oder einzufangen. Zusätzlich verwaltet die Klasse weitere Eigenschaften wie die Startposition der Geister, deren Farbe zur Unterscheidung und die Geschwindigkeit. Somit ist die Ghost-Klasse für das gegnerische Verhalten zuständig und trägt entscheidend zur Herausforderung des Spiels bei.

**Für die grafische Darstellung ist die Renderer-Klasse verantwortlich, welche in Renderer.h und Renderer.cpp implementiert ist.** Diese Klasse kapselt das Zeichnen aller visuellen Elemente des Spiels. Dazu gehören das Spielfeld, die Spielfiguren, die Menüs, der Game-Over-Bildschirm und die Anzeige der Bestenliste. Der Renderer lädt und verwaltet die benötigten Texturen, die für das Spiel verwendet werden, und sorgt dafür, dass alle Elemente korrekt und zeitgerecht auf dem Bildschirm dargestellt werden. Durch diese Kapselung wird die grafische Ausgabe vom restlichen Spielcode getrennt und die Wartbarkeit verbessert.

**Die Verwaltung der Highscores übernimmt die Klasse Leaderboard, die in Leaderboard.h und Leaderboard.cpp definiert ist.** Diese Klasse liest die Bestenliste aus einer Datei ein, sortiert die Einträge nach Punkten und bietet Methoden, um neue Highscores hinzuzufügen und abzufragen. Die Darstellung der Bestenliste erfolgt durch den Renderer. Die Leaderboard-Klasse sorgt so dafür, dass die Spielergebnisse dauerhaft gespeichert und übersichtlich präsentiert werden.

**Der gesamte Programmablauf gliedert sich in mehrere Phasen.** Nach dem Start in main.cpp und der Initialisierung des Zufallszahlengenerators wird ein Game-Objekt erzeugt, das die Hauptsteuerung übernimmt. Die Game-Klasse erzeugt daraufhin das Spielfeld, setzt die Startpositionen für Pac-Man und die Geister, lädt die Bestenliste aus der Datei und stellt den Spielzustand auf das Startmenü.

Die Hauptspielschleife läuft so lange, wie das Spielfenster geöffnet ist. Dabei wird der aktuelle Zustand überwacht und verschiedene Bildschirme angezeigt sowie Eingaben verarbeitet. Im Startmenü wartet das Programm auf eine Aktion des Spielers, wie das Starten des Spiels, das Anzeigen der Bestenliste oder das Beenden des Programms. In der Namenseingabe kann der Spieler seinen Namen für die Highscoreliste eingeben. Während des Spiels werden die Bewegungen von Pac-Man und den Geistern verarbeitet, Kollisionen erkannt, Münzen und Früchte eingesammelt und der Punktestand aktualisiert. Im Pausenmodus wird das Spiel angehalten, bis der Spieler das Spiel fortsetzt oder zum Hauptmenü zurückkehrt. Die Bestenliste wird im entsprechenden Zustand angezeigt. Nach dem Spielende wird die Punktzahl ausgewertet, ein neuer Highscore bei Bedarf gespeichert und der Game-Over-Bildschirm angezeigt.

Beim Verlassen des Spiels werden alle geladenen Ressourcen freigegeben und das Spielfenster geschlossen, sodass ein sauberer Abschluss gewährleistet ist.