**Design-und Implementierungsentscheidungen:**

**Entwurfsmuster:**

Als Entwurfsmuster kommt das Singleton zum Einsatz, um eine effiziente und konsistente Verwaltung aller Spielressourcen zu gewährleisten. Die Klasse Assets ist als Singleton implementiert und ermöglicht den globalen Zugriff auf zentrale Ressourcen wie Bilder, Audiodateien und Konfigurationsparameter. Es wird sichergestellt, dass nur eine Instanz existiert, was die Konsistenz und Koordination der Spielressourcen optimiert. Die übersichtliche Organisation der Ressourcen und die Einbindung von Schriftarten tragen zur visuell ansprechenden Gestaltung des Spiels bei. Die Wahl des Singleton-Entwurfsmusters verbessert die Skalierbarkeit, Pflege und Erweiterung der Ressourcen, fördert die Wartbarkeit des Codes und optimiert die Performance des Pygame-Spiels.

**Weitere Entscheidungen:**

Die objektorientierte Design-Struktur ermöglicht eine klare und modulare Organisation des Codes, wobei Klassen und Vererbung eine effiziente Hierarchie schaffen. Die Klasse Entity definiert gemeinsame Eigenschaften, wodurch redundanter Code minimiert wird. Die Verwendung von Enums für Spielzustände verbessert die Lesbarkeit und verhindert potenzielle Fehler. Die Integration einer config.json-Datei ermöglicht eine einfache Anpassung sämtlicher Spielparameter ohne Quellcodeänderungen, was entscheidend für die schnelle Umsetzung von Nutzer-Feedbacks und die Optimierung des Spielgleichgewichts ist. Das SaveLoadSystem ermöglicht eine nahtlose Fortsetzung des Spiels über verschiedene Spielsessions hinweg, indem es wichtige Spielerdaten speichert.