De flesta av våra subsystem är gjorda som singletons och flera är det tänkt att man skall ärva av. Tanken att några av klasserna är abstract är att vi ville tillåta spelutvecklare så mycket frihet som möjligt när de använder vår motor för att bygga spel och också för att det är nästan omöjligt för oss att tänka oss alla de olika situationer som kan uppstå under utveckling. Dessa klasser är då tänkta att ta hand om det viktigaste som får spelet att köra men ändå tillåta den frihet som behövs för att kunna göra så många olika spel som möjligt.

Den centrala delen av spelmotorn är *GameCore* som skapar och initierar alla subsystem. Vi byggde det på detta sätt dels för att det är enkelt och väldigt lätt att få en övergripande bild av hur man bygger grunden för ett spel och dels för att GameCore skall kunna sköta så mycket arbete mellan subsystemen som möjligt. I GameCore får man också skapa de olika GameActions som spelet kommer behöva. Dessa GameActions binds initialt till en tangent som kan ses som default binding i spelet. GameActions skapas i GameCore för att de ska vara globala för alla GameStates som spelet kommer använda men kan också bindas om i dessa GameStates.

Själva spelet kommer byggas mest i form av olika *GameStates* som är en abstract klass tänkt att hantera olika delar av spelet. Dessa delar kan tänkas vara olika banor och/eller menyer. Vi valde att dela upp spelet på detta sett mest eftersom det är väldigt enkelt att få en övergripande bild av spelet men också för att det blir väldigt lätt att buggtesta varje GameState. GameState har i sig flera metoder som är bra att anropa i en klass som ärvt av GameState och kräver då att man anropar super metoden. Allt detta betyder att en spelutvecklare som använder vår motor kommer i huvudsak skriva all sin kod i sina egenskapade klasser som ärver av GameState eller i de klasser som ärver av GameObject.

*GameObject* är en abstract klass som föga förvånande är till för att hantera de olika objekten i spelet. GameObject har i sig en referens till en instans av klassen Body som är till för att representera objektet för fysikmotorn samt en referens till en instans av klassen Sprite som representerar grafiken hos objektet. Objektet kan finnas med null referens till både Body och Sprite. Detta medför att objekt kan existera med grafik utan att skapa kollisioner och vice versa.