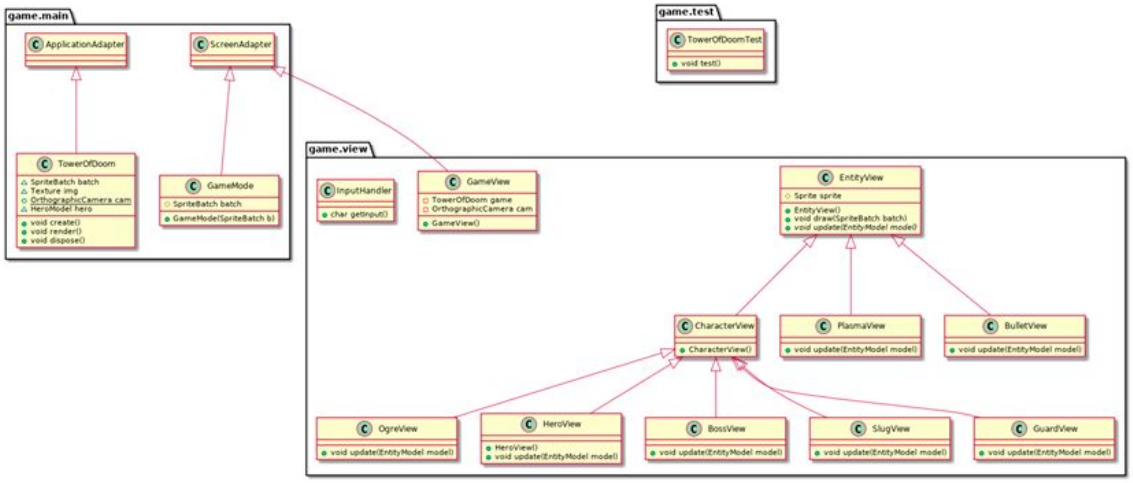
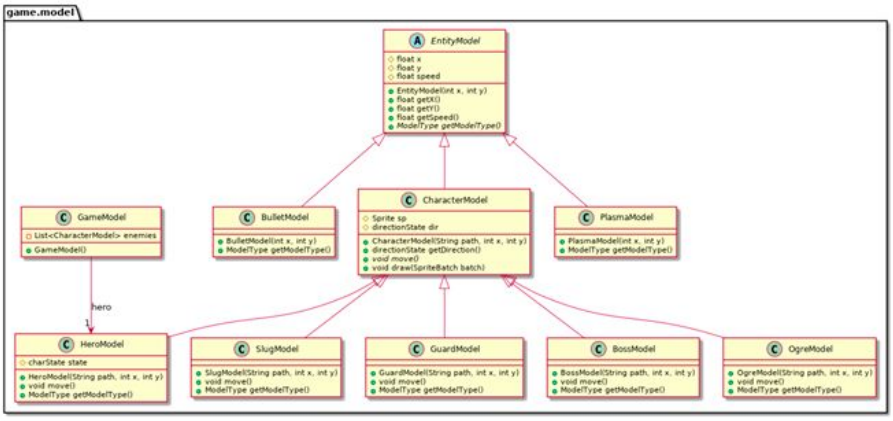
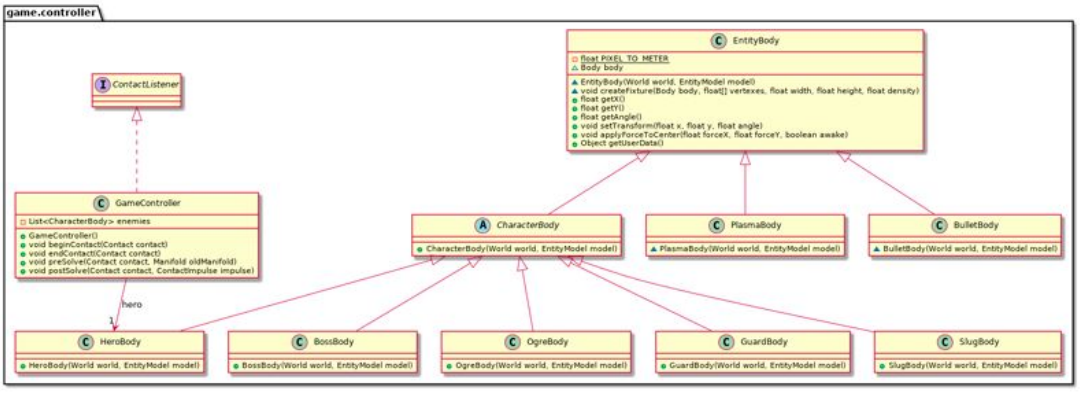
Architecture Design







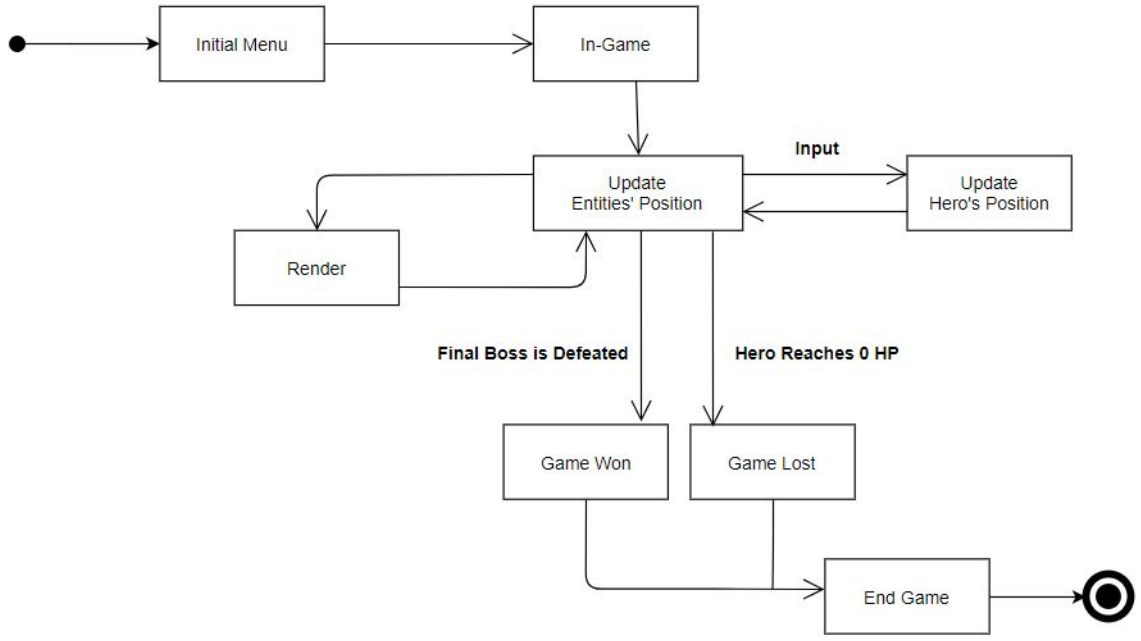
No jogo que estamos a desenvolver recorremos à framework libGDX e como tal damos uso a algumas das classes e interfaces já implementados na mesma, tal como se pode notar pelo UML. Por exemplo recorremos ao classe Body e à interface ContactListener de forma a detectar colisão entre os Personagens e objetos presentes no jogo.

Quase todos os elementos que constituem o jogo são entidades que podem ser ou objectos ou personagens, e o que os distingue é que os últimos estão em constante movimento e são animados, enquanto que os últimos são são inanimados e encontram-se parados, exceto em caso de colisão.

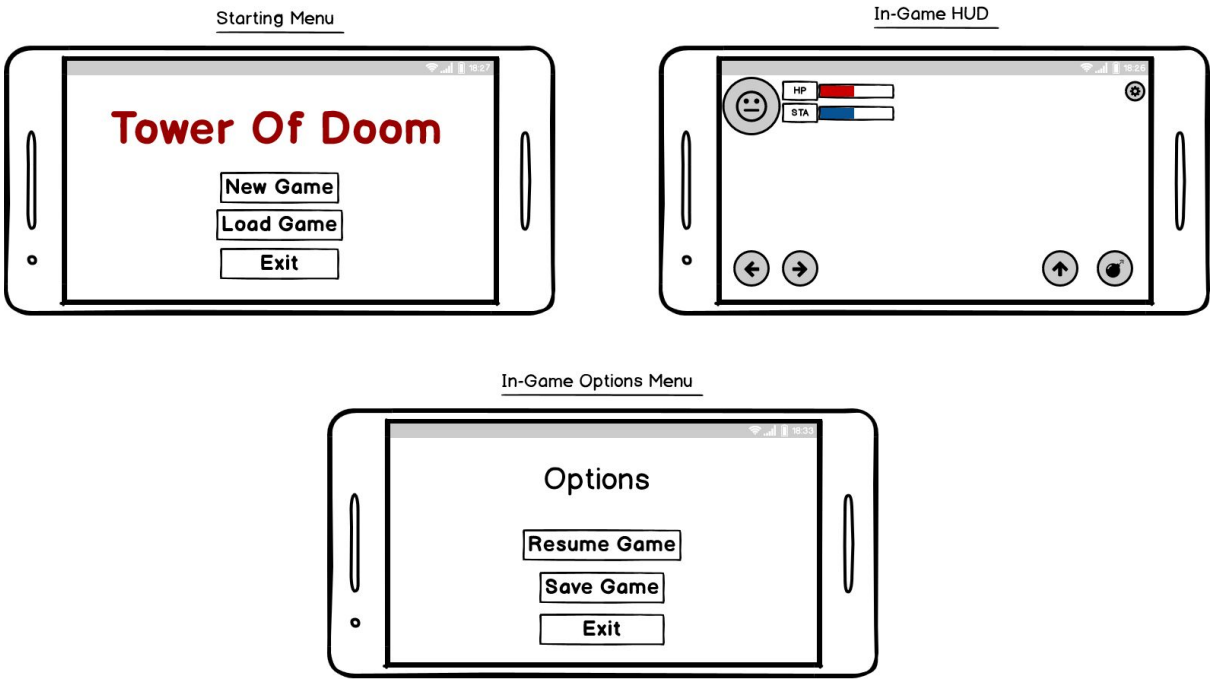
Pelos conhecimentos obtidos nas aulas teóricas, decidimos dividir a informação de cada personagem/objecto em classes de 3 tipos: Model,Body e View. A primeira representa a informação abstrata da entidade, como por exemplo as suas coordenadas na cena de jogo. O Body é utilizado para detectar as interações entre as entidades, nomeadamente as colisões. E o View tem como objetivo representar visualmente a entidade na sua respectiva posição num determinado instante e mostrar o seu estado atual.

De forma a coordenar as classes de cada tipo, existe uma classe geral para cada uma delas. Essas classes são: GameController, GameModel e GameView

Behavioural Aspects



GUI Design



O “Starting Menu” será o menu inicial. Terá a opção de “New Game” que permitirá iniciar um novo jogo. Terá também a opção de “Load Game” que permite ao utilizador continuar o jogo a partir do último save file. A opção “Exit” termina o jogo.

O “In-Game HUD” é um exemplo da HUD que pretendemos implementar. Mostra a HP e Stamina do Hero (canto superior esquerdo). Tem os principais controlos no fundo do ecrã (andar para a esquerda, andar para direita, saltar e lançar plasma ball). No canto superior direito teremos um ícone para o menu de opções “in-game”.

O “In-Game Options Menu” pausa o jogo. Irá ter a opção “Resume Game” para o utilizador voltar ao jogo, a opção “Save Game” que atualiza o save file para o estado atual, e a opção “Exit” que faz com que o jogo termine e volte ao menu inicial (mas não faz “Save Game” automaticamente).

Test Design

Os nossos testes serão apenas focados nas classes que se encontram nos packages Model e Controller.

Lista dos testes:

- Testar se as mudanças de estados do Hero ocorrem quando o programa recebe os inputs corretos, e se o estado não muda com outros inputs;
- Testar se os inputs que deveriam alterar os valores de velocidade do Hero, alteram-nos corretamente;
- Testar se as colisões entre Hero-Enemies são feitas corretamente e se os valores do Hero são devidamente atualizados;
- Testar se quando o Boss é derrotado, o jogo termina (com vitória por parte do Hero);
- Testar se quando o Hero chega a 0 HP, o jogo termina (com derrota por parte do Hero);
- Testar se a HP do Hero diminui quando sofre ataques dos diferentes Enemies;
- Testar se a Stamina do Hero diminui quando atira uma plasma ball;
- Testar se a Stamina do Hero regenera gradualmente em função do tempo;
- Testar se as plasma balls do Hero diminuem a HP dos Enemies quando os atingem;