Hatch Run

Para o projeto final optamos por fazer uma versão da aplicação *Subway Surfers*. Nesta versão (que segue um tema alusivo a *Hatches*).

O objetivo do jogo é atingir o *score* mais alto possível. Os pontos são ganhos através das seguintes formas:

- cada segundo vale 1 ponto;
- cada moeda apanhada vale 10 pontos;
- existe também um *power-up* que duplica o valor das moedas durante um período de tempo;

Desenvolvemos este jogo em *Android*.

Autoras: Beatriz Soares Mendes, up201604253
Joana Sofia Mendes Ramos, up201605017

Set Up / Instalação

A instalação da aplicação baseia-se somente na instalação do apk entregue.

Design Patterns

Próprios à API *Libgdx* e utilizados por nós

- *Observer Pattern*: Utilizada na classe do tipo *Screen Adapter*, que implementa a interface *Screen*: esta classe chama autonomamente o método *render* quando necessário. Cada *screen* criado é um observador. Utilizada de forma a facilitar a gestão de vários *screens* abstraindo-nos da forma como eles são atualizados.
- *Game Loop*: A aplicação é criada, *renderizada* e atualizada de forma abstrata, recorrendo a *Game*, *Screen* e *Listeners*.

Postos em prática por nós

- *Model View Controller*: Escolhido por facilitar os testes à aplicação (aplicados ao módulo *Controller*), assegurar coesão, *low coupling* e facilidade de alterações uma vez que todos os módulos utilizam os mesmos dados. O *Model package* terá as estruturas de dados dos objetos de jogo; O *View package* terás as *views* para esses mesmos objetos (com este *pattern* é também uma vantagem podermos ter várias *views* para os mesmos objetos) e receberá os *inputs* da interação com o utilizador, passando-os ao *Controller* para tratamento; O *Controller package* será responsável pela lógica de jogo, atualizando devidamente o *Model* após tratar o *input* do utilizador (ou a falta dele).
- *Factory*: Utilizado para fabricar botões, uma vez que eram necessários várias vezes.
- *Update*: Usado na *View* e no *Controller*, pois é o utilizado o *delta* para dar update dos vários objetos na mesma frame.
- *Singleton*: Utilizado para garantir que só existiam os objetos estritamente necessários. Foi utilizado para o *GameController* e para o *GameModel.*

Principais Dificuldades

Inicialmente, tínhamos a intenção de usar englobar *socials* no funcionamento do nosso jogo. No entanto, fomo-nos apercebendo que as nossas hipóteses estavam bastante limitadas:

- *Google Play Services* só é possível usar pagando para ter a aplicação na Play Store.
- *API do Facebook*: devido aos problemas de privacidade que a empresa teve, foram recentemente retiradas bastantes permissões.

Outra dificuldade encontrada no desenvolvimento da aplicação, foi a realização de testes unitários para a lógica do jogo devido à utilização da *libgdx*.

Tempo de Desenvolvimento/ Distribuição

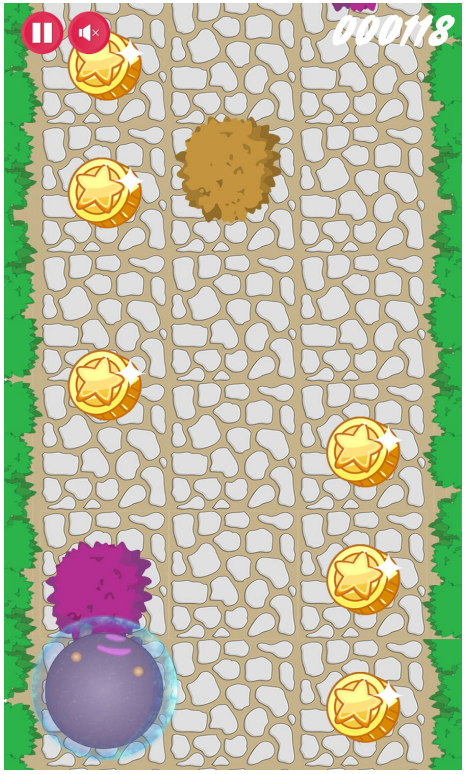
Ambas dedicamos bastante tempo e empenho no desenvolvimento da aplicação, o qual foi iniciado há 1 mês atrás nas aulas práticas, no entanto, nas últimas duas semanas foi quando foi feito um maior progresso na aplicação.

O trabalho foi igualmente dividido, seguindo a seguinte distribuição:

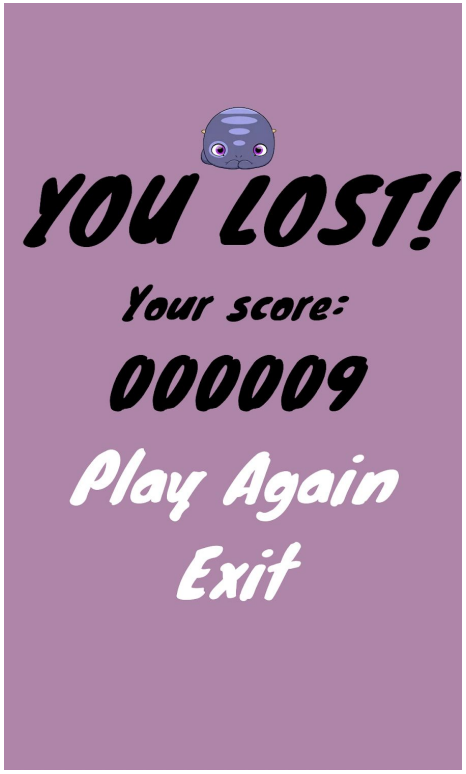
Beatriz Mendes
Lógica principal de jogo
Menus
Log in do facebook
Efeitos de som

Joana Ramos
Lógica de jogo
Menus
Giroscópio

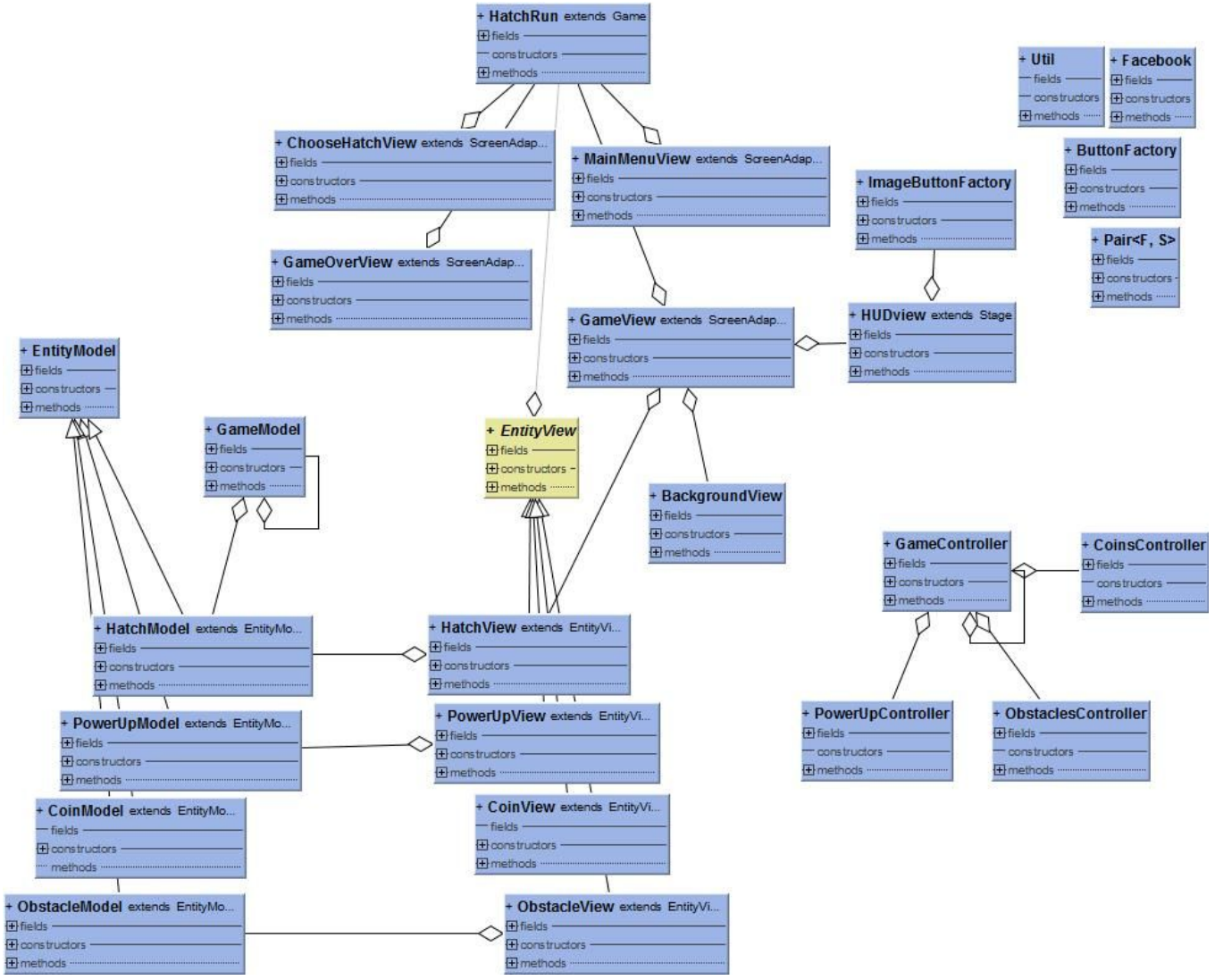
Manual de Utilizador


000118


000164
PAUSED


YOU LOST!
Your score:
000009
Play Again
Exit

UML


+ HatchRun extends Game
fields
constructors
methods
+ Util
fields
constructors
methods
+ Facebook
fields
constructors
methods
+ ChooseHatchView extends ScreenAdap...
fields
constructors
methods
+ MainMenuView extends ScreenAdap...
fields
constructors
methods
+ ButtonFactory
fields
constructors
methods
+ ImageButtonFactory
fields
constructors
methods
+ Pair<F, S>
fields
constructors
methods
+ GameOverView extends ScreenAdap...
fields
constructors
methods
+ GameView extends ScreenAdap...
fields
constructors
methods
+ HUDview extends Stage
fields
constructors
methods
+ EntityModel
fields
constructors
methods
+ GameModel
fields
constructors
methods
+ EntityView
fields
constructors
methods
+ BackgroundView
fields
constructors
methods
+ GameController
fields
constructors
methods
+ CoinsController
fields
constructors
methods
+ HatchModel extends EntityMo...
fields
constructors
methods
+ HatchView extends EntityVi...
fields
constructors
methods
+ PowerUpModel extends EntityMo...
fields
constructors
methods
+ PowerUpView extends EntityVi...
fields
constructors
methods
+ PowerUpController
fields
constructors
methods
+ ObstaclesController
fields
constructors
methods
+ CoinModel extends EntityMo...
fields
constructors
methods
+ CoinView extends EntityVi...
fields
constructors
methods
+ ObstacleModel extends EntityMo...
fields
constructors
methods
+ ObstacleView extends EntityVi...
fields
constructors
methods