

EduFarma - Dokumentacja Projektu

1. Opis Projektu

1.1 Logowanie Do Systemu

1.1.1 Nauczyciel

1.1.2 Gracz

1.2 Server

1.2.1 Komunikacja z serwerem

1.2.2 Architektura

1.2.3 Serwer Smartfox

1.2.4 Decyzje Zakupowe Realizowane Przez Server Smartfox

1.2.5 Komunikacja

1.2.6 Technologia wykonania bazy danych

1.2.7 Struktura bazy danych

1.3 Klient Gry

1.3.1 Technologia

2. Specyfikacja Techniczna

2.1 Serwer - Wymagania Techniczne

2.2 Klient - Wymagania Techniczne

2.3 Instalacja Systemu

3. Rozgrywka

3.1 Scenariusz Gry

3.2 Rozgrywka

3.1.2 Początek Rozgrywki

3.2.3 Sprzedaż Produktów

3.2.4 Runda Kupowania

3.3 Nauczyciel

3.3.1 Zadania Nauczyciela

4. Zawartość Płyty Z Plikami Źródłowymi Gry

4.1 Assets

4.2 Dev

4.3 Documents

1. Opis Projektu

1.1 Logowanie Do Systemu

1.1.1 Nauczyciel

Aby zalogować się do systemu jako nauczyciel należy zalogować się w serwisie:
www.edukariera.com.pl

1.1.2 Gracz

Aby zalogować się do systemu jako gracz (uczeń) należy zalogować się w serwisie:
www.edukariera.com.pl

1.2 Server

1.2.1 Komunikacja z serwerem

Baza danych komunikuje się z serwerem SmartFox poprzez sterownik JDBC - Java DataBase Connectivity.

1.2.2 Architektura

Aplikacja wykonana jest w architekturze klient-serwer.

Klient dostępny jest na portalu www.edukariera.com.pl i przeznaczony jest dla gracza oraz nauczyciela. Aplikacja serwerowa, z którą łączą się wszystkie aplikacje klienckie-SmartFox, zainstalowana jest na serwerze OVH.pl

1.2.3 Serwer Smartfox

Aplikacja SmartFox jest centralnym węzłem, do którego podłączają się wszyscy użytkownicy gry. Synchronizuje grę oraz pośredniczy w wymianie danych pomiędzy poszczególnymi graczami.

Wykonuje skrypty zapisane w języku ActionScript odpowiedzialne za sztuczną inteligencję zarządzającą zakupami produktów od poszczególnych graczy.

1.2.4 Decyzje Zakupowe Realizowane Przez Server Smartfox

W zaprogramowanych odstępach czasowych aplikacja SmartFox dla każdej sesji gry osobno wykonuje skrypty mające na celu wybór oraz zakup produktów o poszczególnych graczy.

Skrypty opierają się na algorytmach ekonomicznych, a ich zadaniem jest wyłonienie najbardziej opłacalnego zakupu. Podstawowym kryterium jest to potencjalny zysk, jaki wygeneruje wirtualny sklep zarządzany przez sztuczną inteligencję jeśli zakupi dany produkt po danej cenie w danej ilości. Najbardziej opłacalne są oferty z jak najniższą ceną jednostkową i jednocześnie zawierające największą ilość danego produktu. Dzięki temu wirtualny sklep oszczędza na kosztach transportu, jednocześnie zarabiając najwięcej przy ustalonej wielkości marży.

Zastosowany algorytm ekonomiczny wylicza zysk sklepu na podstawie wzoru:

$\text{Zysk} = \text{Ilość Produktu} \times (\text{Cena Sprzedaży} - \text{Cena Zakupu})$.

Cena zakupu od gracza to cena, po której sklep skupuje produkt od gracza.

Cena sprzedaży to teoretyczna cena, po której sklep sprzedaje produkty wcześniej kupione od graczy.

Tabela poniżej przedstawia zapotrzebowanie (nie jest nieograniczone) oraz teoretyczne ceny sprzedaży poszczególnych sklepów:

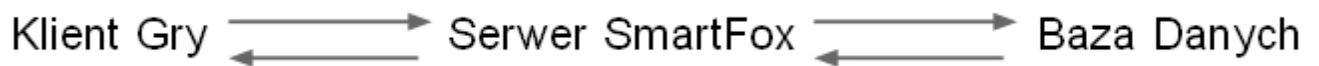
BUILDING	Product	Demand 1	Price 1	Demand 2	Price 2	Demand 3	Price 3
Orchard	Fruits	360	39	288	42	180	45
JuiceFactory	Juice	144	238	116	255	72	272
JamFactory	Jam	72	510	58	544	36	578
Vegfield	Fieldvegs	440	33	352	35	220	38
Vegfactory	Processedvegs	176	175	140	188	88	200
Frozefactory	Frozenvegs	88	525	70	560	44	595
Greenhouse	Housevegs	340	52	272	56	170	60
Canfactory	Canvegs	136	266	108	285	68	304
Saladfactory	Salad	68	615	54	656	34	697
Chickencoop	Eggs	600	65	480	70	300	75
Pigsty	Pigs	30	195	24	210	16	225
Pigslaughter	Pigmeat	76	490	60	525	38	560
Pigbutchery	Pigsausage	60	975	48	1040	30	1105
Gallantryfactory	Gallantry	30	1318	24	1403	16	1488
Brojler	Chickens	100	78	80	84	50	90
Chickensloughter	Chickenmeat	140	364	112	390	70	416
Chickenbutchery	Chickensausage	80	690	64	736	40	782
Cowshed	Milk	400	20	320	21	200	23
Butterfactory	Butter	160	91	128	98	80	104
Cheesfactory	Cheese	80	195	64	208	40	221
Field	Wheat	100	7	80	7	50	8
Windmill	Flour	150	70	120	75	76	80
Bakery	Bread	150	195	120	208	76	221
Noodlefactory	Noodle	130	372	104	396	66	420
Sunflowerfield	Sunseeds	170	16	136	17	86	18
Embossfactory	Oil	222	140	178	150	112	160
Hardenfactory	Margarine	136	360	108	384	68	408

Sztuczna inteligencja zarządza w sumie 3 “typami” sklepów, różniącymi się asortymentem i maksymalną ceną, po której są skłonne kupić dany produkt (więc wielkością swojej marży).

Serwer SmartFox realizuje skrypty sztucznej inteligencji w czasie rundy kupowania- wtedy to rozgrywka jest pauzowana, od graczy pobierane są dane na temat ofert sprzedaży, które następnie porównywane są przez serwer. Skrypty wybierają najlepsze oferty za pomocą algorytmu opisanego powyżej i "pobierają" z rynku odpowiednie towary, jednocześnie przesyłając na konto gracza odpowiednie kwoty.

1.2.5 Komunikacja

Cała komunikacja przebiega według schematu:



Klient gracza oraz nauczyciela łączy się jedynie z serwerem SmartFox. SmartFox natomiast komunikuje się z ze wszystkimi graczami oraz nauczycielami zapewniając wymianę danych w obrębie poszczególnych klas i pokoi, a następnie zapisuje odpowiednie dane w bazie danych.

Serwer SmartFox kontroluje jednocześnie wszystkie toczące się gry zapewniając rozdzielenie komunikacji na wszystkie pokoje (klasy). Jest także odpowiedzialny za mechanikę logowania się poszczególnych nauczycieli oraz użytkowników.

1.2.6 Technologia wykonania bazy danych

Relacyjna baza danych wykonana jest w technologii MySQL.

1.2.7 Struktura bazy danych

```
TABLE `class`  
  `id` bigint(20) NOT NULL,  
  `name` varchar(50) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,  
  `school_id` bigint(20) NOT NULL,  
  `teacher_id` bigint(20) NOT NULL,  
  `room_name` varchar(50) NOT NULL DEFAULT "",  
  `tutorial` tinyint(1) NOT NULL DEFAULT '1',  
  `max_users` int(11) NOT NULL,  
PRIMARY KEY (`id`)  
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
TABLE `finances_history`  
  `ID` int(11) NOT NULL AUTO_INCREMENT,  
  `user_id` int(11) NOT NULL,  
  `round_number` int(11) NOT NULL,  
  `money` int(11) NOT NULL DEFAULT '0',  
  `income` int(11) NOT NULL DEFAULT '0',  
  `loan` int(11) NOT NULL DEFAULT '0',  
  `buildings_cost` int(11) NOT NULL DEFAULT '0',  
  `upgrades_cost` int(11) NOT NULL DEFAULT '0',  
  `expand_cost` int(11) NOT NULL DEFAULT '0',  
  `production_cost` int(11) NOT NULL DEFAULT '0',  
  `paid_loan_rate` int(11) NOT NULL DEFAULT '0',  
PRIMARY KEY (`ID`)  
ENGINE=InnoDB AUTO_INCREMENT=150 DEFAULT CHARSET=latin2;
```

```
TABLE `offer_history`  
  `ID` int(11) NOT NULL AUTO_INCREMENT,  
  `round_number` int(11) NOT NULL,  
  `room_id` int(11) NOT NULL,  
  `user_id` int(11) NOT NULL,  
  `product_id` int(11) NOT NULL,  
  `offer_amount` int(11) NOT NULL DEFAULT '0',  
  `offer_price` int(11) NOT NULL DEFAULT '0',  
  `bought_amount` int(11) NOT NULL DEFAULT '0',  
PRIMARY KEY (`ID`)  
ENGINE=MyISAM AUTO_INCREMENT=100 DEFAULT CHARSET=latin2;
```

```
TABLE `products_in_progress`  
  `ID` int(11) NOT NULL AUTO_INCREMENT,  
  `owner` int(11) NOT NULL,  
  `building_id` int(11) NOT NULL,  
  `product_id` int(11) NOT NULL,  
  `amount` int(11) NOT NULL,  
  `progress` int(11) NOT NULL DEFAULT '0',  
PRIMARY KEY (`ID`)  
ENGINE=MyISAM AUTO_INCREMENT=149 DEFAULT CHARSET=latin2;
```

```
TABLE `products_in_warehouse`  
  `ID` int(11) NOT NULL AUTO_INCREMENT,  
  `owner` int(11) NOT NULL,  
  `product_id` int(11) NOT NULL,  
  `type` varchar(20) NOT NULL,  
  `amount` int(11) NOT NULL,  
PRIMARY KEY (`ID`)  
ENGINE=MyISAM AUTO_INCREMENT=136 DEFAULT CHARSET=latin2;
```

```
TABLE `products_on_marketplace`  
  `ID` int(11) NOT NULL AUTO_INCREMENT,  
  `owner` int(11) NOT NULL,  
  `product_id` int(11) NOT NULL,  
  `amount` int(11) NOT NULL,  
  `price` int(11) NOT NULL,  
PRIMARY KEY (`ID`)  
ENGINE=MyISAM AUTO_INCREMENT=139 DEFAULT CHARSET=latin2;
```

```
TABLE `rounds`  
  `ID` int(11) NOT NULL AUTO_INCREMENT,  
  `round_number` int(11) NOT NULL,  
  `room_id` int(11) NOT NULL,  
PRIMARY KEY (`ID`)  
ENGINE=MyISAM AUTO_INCREMENT=12 DEFAULT CHARSET=latin2;
```

```
TABLE `school`  
  `id` bigint(20) NOT NULL,  
  `name` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,  
  `city` varchar(128) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,  
PRIMARY KEY (`id`)  
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
TABLE `student`  
  `id` bigint(20) NOT NULL AUTO_INCREMENT,  
  `name` varchar(50) NOT NULL,  
  `surname` varchar(50) NOT NULL,  
  `email` varchar(255) NOT NULL,  
  `class_id` bigint(20) NOT NULL,  
  `cash` int(11) NOT NULL DEFAULT '0',  
  `loan` int(11) NOT NULL DEFAULT '0',  
  `loan_rate` int(11) NOT NULL DEFAULT '0',  
  `loan_length` int(11) NOT NULL DEFAULT '0',  
  `field_size_x` smallint(3) NOT NULL,  
  `field_size_y` smallint(3) NOT NULL,  
  `accepted` tinyint(1) NOT NULL DEFAULT '0',  
PRIMARY KEY (`id`)  
ENGINE=InnoDB AUTO_INCREMENT=1065 DEFAULT CHARSET=utf8;
```

```
TABLE `teacher`  
  `id` bigint(20) NOT NULL,  
  `name` varchar(50) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,  
  `surname` varchar(50) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,  
  `email` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,  
  `school_id` bigint(20) NOT NULL,  
PRIMARY KEY (`id`)  
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
TABLE `user_has_buildings`  
  `ID` int(5) NOT NULL AUTO_INCREMENT,  
  `type_id` tinyint(2) NOT NULL,  
  `x` tinyint(2) NOT NULL,  
  `y` tinyint(2) NOT NULL,  
  `level` tinyint(1) NOT NULL,  
  `owner` int(5) NOT NULL,  
  `upgrade_progress` int(11) NOT NULL DEFAULT '0',  
PRIMARY KEY (`ID`)  
ENGINE=MyISAM AUTO_INCREMENT=125 DEFAULT CHARSET=latin2;
```

1.3 Klient Gry

1.3.1 Technologia

Klient gry wykonany został w technologii Adobe Flash.

Do prawidłowego działania aplikacja wymaga instalacji w dowolnej przeglądarce internetowej "wtyczki" (plug-in) Adobe Flash w wersji 10.0 lub wyższej.

2. Specyfikacja Techniczna

2.1 Serwer - Wymagania Techniczne

Serwer Dedykowany
System: Debian 64Bit
Procesor: Intel Xeon lub nowszy
Pamięć RAM: 4GB
Pamięć masowa: 10 GB
Połączenie sieciowe: 1 Gbps

2.2 Klient - Wymagania Techniczne

Komputer klasy PC
System: Windows XP/Vista/7, Linux, MacOS
Przeglądarka internetowa zgodna z Adobe Flash (Firefox/Chrome/IE/Safari)
Procesor: Intel Celeron/ AMD Athlon lub nowszy
Pamięć RAM: 512MB
Pamięć HDD: 10GB

2.3 Instalacja Systemu

Instalacja klienta gry EduFarma:

Aby możliwe było uruchomienie klienta gry EduFarma niezbędna jest wtyczka do przeglądarki internetowej: Adobe Flash Player. Aby zainstalować wtyczkę należy wejść na adres:

<http://get.adobe.com/pl/flashplayer/>

Instalacja części serwerowej gry EduFarma:

Wymagania dotyczące oprogramowania na serwerze:

- php v. >= 5.3 (wraz z pluginem do mysqlmysqli)
- apache 2
- mysql >= 5
- SmartFox 1.6.6

Konfiguracja SmartFox'a:

Rozszerzenia gry należy wgrać do katalogu Server/sfsExtensions w folderze z SmartFox'em.

Konfiguracja bazy danych znajduje się w w pliku z rozszerzeniem o nazwie extDb.as, funkcja connectDB. Po wgraniu rozszerzeń należy zrestartować SmartFox'a

Zmiana ścieżek w aplikacji i konfiguracja

Zmiana adresu smartfox'a w grze: package: pl.fabrykagier.edufarma.communication, klasa: Communicator, metoda: connect, zmienna: address

Zmiana ustawień bazy danych w API: katalog: /api/protected/config/hosts/localhost.php

Konfiguracja środowiska:

1. Instalacje wymaganych pakietów na serwerze
2. Wgranie źródeł PHP do katalogu /var/www
3. Wgranie struktury bazy danych
4. Konfiguracja SmartFox'a
5. Zmiana ścieżek w aplikacji i konfiguracja

3. Rozgrywka

3.1 Scenariusz Gry

Maksymalnie 16 graczy prowadzi własne, niezależne od siebie farmy (jedna farma dla każdego gracza). Mają do dyspozycji narzędzia pozwalające im na budowanie struktur, dzięki którym mogą następnie produkować towary oraz je magazynować. Zmagazynowane towary każdy gracz może wystawić na sprzedaż tworząc ofertę sprzedaży na rynku. Co określoną ilość czasu sztuczna inteligencja sprawdza wszystkie oferty w obrębie jednej gry (klasy), a następnie wybiera te najkorzystniejsze biorąc pod uwagę czynniki ceny i ilości. Za zrealizowane oferty gracz otrzymuje wirtualne środki pieniężne pozwalające na rozwój farmy i produkcję następnych towarów.

3.2 Rozgrywka

Po zalogowaniu na serwisie:

www.edukariera.com.pl

oraz uruchomieniu gry EduFarma gry wyświetli się okno gry.

Pojawienie się ekranu oczekiwania oznacza, że teraz gracz musi czekać na wystartowanie gry przez nauczyciela.

3.1.2 Początek Rozgrywki

Zadaniem gracza jest zbudowanie dobrze prosperującego gospodarstwa rolnego dzięki właściwemu planowaniu produkcji, inwestycjom oraz właściwym zarządzaniem finansami.

Jedynym źródłem dochodu gracza jest sprzedaż wytworzonych produktów przy czym oprócz niego na rynku funkcjonują równocześnie inne, konkurencyjne gospodarstwa.

Gracz rozpoczyna rozgrywkę z pustym obszarem oraz startową ilością “gotówki”. **Na początek** potrzebuje przede wszystkim magazynu odpowiedniego typu, w którym będzie mógł składować swoje produkty. Gdy magazyn będzie już gotowy należy zatroszczyć się o “budynek” produkcyjny, np. sad owocowy. Dzięki niemu gracz może rozpocząć produkcję owoców, które po zebraniu trafią do magazynu.

Budowę gospodarstwa najlepiej jest rozpocząć od **wybudowania podstawowego magazynu**. Aby to zrobić należy kliknąć ikonkę “Tryb Budowania” znajdującą się w lewym dolnym rogu ekranu, a następnie przejść do zakładki “Magazyny”. Następnie należy kliknąć na budynek z zielonym dachem (Magazyn), przesunąć kursor na plansze gospodarstwa i kliknąć w miejscu, z którym ma stać budynek. Po określonej ilości czasu proces budowania zostanie zakończony.

Następnie gracz może **postawić** np. **sad owocowy**, zagon z warzywami czy szklarnię. Są to “budynki” produkcyjne, wytwarzające gotowe do sprzedaży produkty. Stawia się je na planszy w identyczny sposób jak magazyny. Warto zauważyć, że klikając na **niebieską ikonkę “i”** znajdującą się nad każdym budynkiem w zakładce budowania, można sprawdzić podstawowe informacje o danej strukturze.

Gdy budynek produkcyjny jest już gotowy, można **rozpocząć produkcję**. Aby to zrobić należy kliknąć na budynek produkcyjny (np. sad), a następnie kliknąć na zielony przycisk “Start”. Po zakończeniu produkcji nad budynkiem pojawi się ikonka z gotowym produktem. Ponowne kliknięcie na budynek automatycznie przenosi gotowy produkt do odpowiedniego magazynu.

W grze zawartych jest wiele możliwych **ścieżek rozwoju gospodarstwa**- gracz może skupić się na masowej produkcji owoców, ale może też zainwestować np. w przetwórnienie owoców czy fabrykę soków. Przetworzone produkty są wartościowsze i można je sprzedać z większą marżą, jednak ich produkcja wymaga wybudowania odpowiednich przetwórni.

Należy zawsze pamiętać o obowiązku posiadania **odpowiednich magazynów**- np. w przypadku mięsa wymagane są chłodnie. Bez odpowiednich powierzchni magazynowych nie jest możliwe "zebranie" produktów z budynków wytwórczych. Produkty mogą czekać na zebranie tylko określoną ilość czasu, po którym ulegają zepsuciu.

Każda z **gałęzi produkcji** składa się najczęściej z 3 budynków- podstawowego, przetwórni I stopnia korzystających z produktów podstawowych i przetwórni II stopnia korzystających z produktów pochodzących z przetwórni I stopnia. Należy pamiętać, że im bardziej przetworzony produkt tym większą marżę można na niego nałożyć podczas sprzedaży- jest więc bardziej zyskowny.

Budynki produkcyjne można również **modernizować**- wiąże się to z nakładami finansowymi i czasem potrzebnym na zakończenie modernizacji, ale dzięki niej budynek będzie produkował szybciej.

3.2.3 Sprzedaż Produktów

Produkty znajdujące się w magazynach należy wystawić na rynek w celu ich sprzedaży. Wystawienie na rynek nie oznacza automatycznie, że produkty zostaną sprzedane.

Gracz funkcjonuje na tym samym rynku co pozostali uczestnicy gry w obrębie danej klasy, a wielkość popytu na każdy produkt jest ograniczony. Oznacza to, że jeśli wszyscy gracze wystawią na rynek np. owoce, to tylko części z nich uda się je sprzedać. Gra, działając zgodnie z zasadami wolnego rynku, skupi najpierw owoce wystawione po najniższej cenie. Będzie to robić aż do wyczerpania całej wielkości popytu. Oznacza to, że część owoców nie zostanie kupiona, a gracze nie otrzymają zapłaty za wystawione oferty sprzedaży. W tym przypadku oferty te pozostaną na rynku, przy czym gracz będzie mógł je wycofać i wystawić swoje produkty ponownie, np. po innej cenie.

Gra przeprowadza zakupy co 5 minut- następuje wtedy runda kupowania, odbywająca się w tym samym czasie dla wszystkich graczy. Wtedy też gra porównuje wszystkie oferty i wybiera dla siebie te najbardziej opłacalne.

Aby **wystawić produkt na sprzedaż** należy kliknąć budynek wybranego magazynu bądź ikonkę “Magazyny” znajdującą się u góry ekranu. Następnie wybrać z listy dany produkt i kliknąć przycisk “Wystaw Na Sprzedaż”. W oknie które się wyświetli gracz decyduje, ile sztuk danego produktu chce wystawić oraz po jakiej cenie. W wyborze pomocna jest informacja ile wynosił jednostkowy koszt wytworzenia, znajdująca się u góry okna. Warto dodać, że zarówno ilość jak i cenę można wpisać “z klawiatury”.

Gracz może sprawdzić listę wystawionych przez siebie produktów na sprzedaż poprzez kliknięcie na ikonkę “Rynek” u góry ekranu.

3.2.4 Runda Kupowania

Runda kupowania odbywa się co 5 minut, dokładnie w tym samym momencie dla wszystkich graczy. Dzięki temu możliwe jest jednoczesne porównanie wszystkich ofert uczestników biorących udział we wspólnej grze (klasie).

Po zakończeniu rundy kupowania każdy gracz otrzymuje podsumowanie jej przebiegu, tzn. które produkty z wystawionych na rynek udało mu się sprzedać, a które nie.

Następne okno zawiera informację na temat postępów gracza- jego bilans finansowy oraz wykres słupkowy przedstawiający wielkość kosztów oraz przychodów.

Pozostałe zakładki okna podsumowania pozwalają na sprawdzenie archiwum sprzedaży gracza oraz w jakich ilościach sprzedawane były poszczególne produkty w czasie poprzednich rund kupowania.

3.3 Nauczyciel

3.3.1 Zadania Nauczyciela

Nauczyciel jest moderatorem gry, mogącym przyjmować nowych graczy, rozpoczynać oraz zatrzymywać grę. Pozwala mu na to klient gry w wersji nauczycielskiej- jest to osobna wersja aplikacji przeznaczona tylko dla nauczyciela. Dzięki odpowiednim przyciskom, nauczyciel wysyła komendy kontrolujące grę (np. start, stop) do serwera SmartFox, który następnie wysyła te komendy do aplikacji wszystkich graczy znajdujących się w danej klasie.

Po zalogowaniu do gry jako nauczyciel okno pokoju przedstawia wszystkich graczy (użytkowników), którzy do niego dołączyli (domyślnie- wszystkich uczniów grających w danej klasie).

Kiedy w pokoju będą znajdować się już wszyscy gracze (uczniowie), nauczyciel może rozpocząć rozgrywkę klikając na przycisk “Start” znajdujący się u dołu ekranu.

Rozpoczęcie rozgrywki powoduje, że wszyscy gracze w tym samym czasie zostają przeniesieni z ekranu oczekiwania do właściwej gry.

Nauczyciel może zatrzymać grę w każdym momencie klikając "Stop". Stan gry zostanie zapisany i zachowany. Wznawiając grę w późniejszym czasie wszyscy gracze rozpoczną grę od momentu, w którym została ona poprzednio zatrzymana.

Zatrzymanie gry powoduje wyświetlenie wyników- gracze zostają posortowani w zależności od stanu konta jakim dysponowali w momencie zatrzymania gry. Dzięki temu możliwe jest sprawdzenie postępów graczy po każdej sesji.

4. Zawartość Płyty Z Plikami Źródłowymi Gry

4.1 Assets

EduFarma_Pliki\EduFarma_Sources\trunk\assets\bmp

Folder, zawierający wszystkie assety graficzne (bitmapy) użyte w grze.

EduFarma_Pliki\EduFarma_Sources\trunk\assets\bmp\placeable

Folder zawierający bitmapy budynków.

EduFarma_Pliki\EduFarma_Sources\trunk\assets\bmp\buttons

Folder zawierający bitmapy przycisków.

EduFarma_Pliki\EduFarma_Sources\trunk\assets\bmp\products

Folder zawierający bitmapy Produktów

4.2 Dev

EduFarma_Pliki\EduFarma_Sources\trunk\dev\ai

Folder zawierający pliki sztucznej inteligencji gry.

EduFarma_Pliki\EduFarma_Sources\trunk\dev\flash

Folder zawierający pliki źródłowe gry.

EduFarma_Pliki\EduFarma_Sources\trunk\dev\flash\as

Folder zawierający kod źródłowy gry.

EduFarma_Pliki\EduFarma_Sources\trunk\dev\flash\FLA

Folder zawierający pliki FLA gry (w nich umieszczona jest grafika oraz dźwięki gry)

EduFarma_Pliki\EduFarma_Sources\trunk\dev\flash\SWC

Folder zawierający przekompilowane pliki FLA, czyli pliki SWC.

EduFarma_Pliki\EduFarma_Sources\trunk\dev\php

Folder zawierający pliki PHP.

EduFarma_Pliki\EduFarma_Sources\trunk\dev\SmartFox

Folder zawierający pliki serwera SmartFox

4.3 Documents

EduFarma_Pliki\EduFarma_Sources\trunk\documents

Folder zawierający dokumentację projektu.

EduFarma_Pliki\EduFarma_Sources\trunk\documents\uml

Folder zawierający diagramy UML.