FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Semestrální práce z předmětu KIV/UPS

Online puzzle

Patrik Harag

harag@students.zcu.cz

(A15B0034P)

2. prosince 2017

Obsah

Kapitola 1

Úvod

1.1 Zadání

Online puzzle, kde hráči společnými silami skládají puzzle. Dílky bude možné volně přesouvat po herní ploše, a to jednotlivě i po skupinách. Bude zahrnovat server v jazyce C a klienta v Javě.

1.2 Zásady vypracování

- Úlohu naprogramujte v programovacím jazyku C/C++ anebo Java. Pokud se jedná o úlohu server/klient, pak klient bude v Javě a server v C/C++.
- Komunikace bude realizována textovým nešifrovaným protokolem nad TCP protokolem.
- Výstupy serveru budou v alfanumerické podobě, klient může komunikovat i v grafice (není podmínkou).
- Server řešte pod operačním systémem Linux, klient může běžet pod OS Windows XP. Emulátory typu Cygwin nebudou podporovány.
- Realizujte konkurentní (paralelní) servery. Server musí být schopen obsluhovat požadavky více klientů souběžně.
- Součástí programu bude trasování komunikace, dovolující zachytit proces komunikace na úrovni aplikačního protokolu a zápis trasování do souboru.
- Každý program bude doplněn o zpracování statistických údajů (přenesený počet bytů, přenesený počet zpráv, počet navázaných spojení, počet přenosů zrušených pro chybu, doba běhu apod.).
- Zdrojové kódy organizujte tak, aby od sebe byly odděleny části volání komunikačních funkcí, které jste vytvořili na základě zadání, od částí určených k demonstraci funkčnosti vašeho řešení (grafické rozhraní).

Kapitola 2

Programátorská dokumentace

2.1 Server

Použité nástroje Server byl vyvinut v C 99.

Základní návrh Program pracuje ve dvou vláknech. Hlavní vlákno po inicializaci prostředí přijímá nová spojení. Druhé vlákno obsluhuje klienty a vyhodnocuje stav her. Toto řešení se ukázalo jako nejlepší z následujících důvodů:

- jednoduchost – v podstatě celá herní logika v jednom vlákně,
- na straně serveru se neprovádí žádné výpočetně náročné operace,
- cílový server stejně nebude mít více fyzických procesorových jader (VPS).

Když hlavní vlákno přijme spojení, vytvoří novou *session* (více o *session* viz dále) a čeká na další spojení. Druhé vlákno postupně prochází všechny *session* a reaguje na jejich zprávy. Každý socket je nastaven jako neblokující.

Ukládání a načítání Je prováděno automaticky při startu a ukončení programu. Jako formát pro uložení je využit samotný síťový protokol. Načtení je realizováno formou simulace spojení. To znamená, že serveru je předána série zpráv, které zpracuje stejným způsobem jako při běžném chodu a tím uvede server do požadovaného stavu.

Testy Server je testován integračními testy s využitím klientského API.

2.1.1 Seznam zdrojových souborů

- *main.c* – obsahuje vstupní funkci, zpracovává parametry a spouští server.
- *buffer.c/h* – pomocná struktura používaná hlavně pro dočasné ukládání zpráv (po přijetí, před odesláním).
- *utils.c/h* – obsahuje různé pomocné funkce.

- *server.c/h* – obsahuje základní logiku serveru.
- *session.c/h* – obsahuje strukturu pro ukládání dat o jednom spojení + funkce které s nimi pracují.
- *session_pool.c/h* – spravuje všechny *session* na jednom místě.
- *game.c/h* – obsahuje strukturu pro ukládání dat o jedné hře + funkce které s nimi pracují.
- *game_pool.c/h* – spravuje všechny *game* na jednom místě.
- *shared.c/h* – spravuje sdílený herní stav, zastřešuje *session_pool* a *game_pool*. Ukládá a načítá herní stav do souboru.
- *controller.c/h* – zpracovává přijaté zprávy.
- *protocol.h* – obsahuje konstanty síťového protokolu.
- *stats.c/h* – spravuje statistiky.

2.1.2 Důležité datové struktury

Session

Tato datová struktura, která se nachází v souboru *session.h*, ukládá informace o jednom spojení. Může představovat i hráče, ovšem po přerušení spojení a následovném opětovném navázání bude vytvořena zcela nová instance.

Přístup k instancím této struktury musí být řízen, jelikož k nim přistupují obě vlákna.

```
typedef struct _Session {

    int id;
    SessionStatus status;

    int socket_fd;
    unsigned long long last_activity;
    unsigned long long last_ping;
    Buffer to_send;
    int corrupted_messages;

    char name[SESSION_PLAYER_MAX_NAME_LENGTH + 1];
    Game* game;

} Session;
```

Game

Struktura ze souboru *game.h*, která obsahuje stav jedné hry.

```
typedef struct _Game {
    int id;
    unsigned int w, h;
    Piece** pieces;
    bool finished;
} Game;

typedef struct _Piece {
    int id;
    int x, y;
} Piece;
```

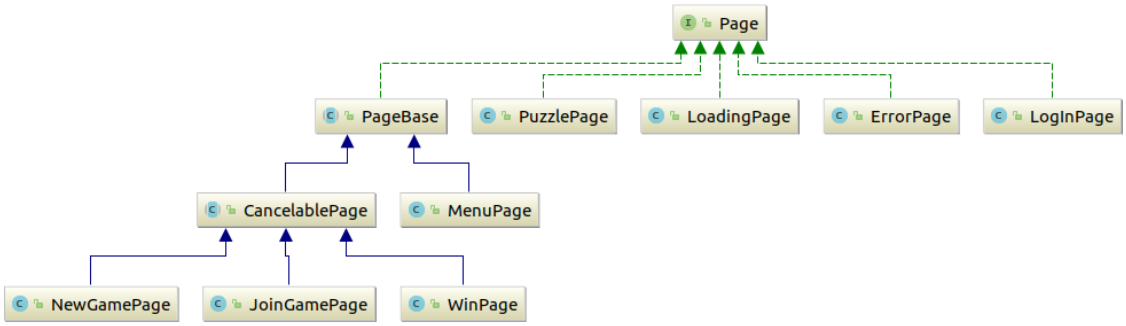
2.2 Klient

Klient je realizován v programovacím jazyce Java 8. Pro uživatelské rozhraní a je použit framework JavaFX.

2.2.1 Balíček cz.harag.puzzle.app

Obsahuje aplikační část – vstupní třídu, která zpracovává vstup z příkazové řádky a grafické uživatelské rozhraní.

Důležitou úlohu zde hraje rozhraní *Page*, které představuje jednu stránku. Existuje například stránka pro přihlášení, vytvoření hry, stránka s chybou atd.



Obrázek 2.1: Diagram tříd – rozhraní *Page*.

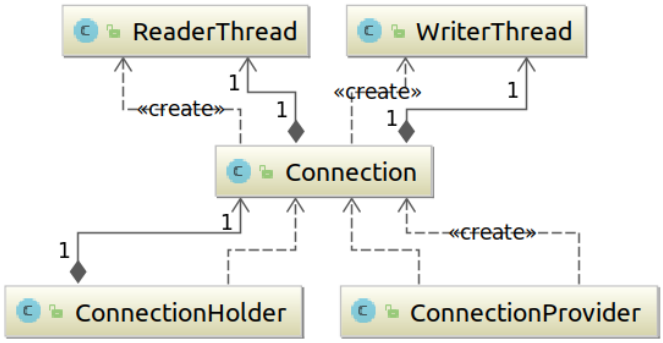
2.2.2 Balíček cz.harag.puzzle.net

Obsahuje síťovou část.

Třídy:

- *Connection* – zprostředkuje posílání a příjímání zpráv.
- *ConnectionHolder* – obaluje *Connection*, poskytuje vyšší abstrakci, udržuje spojení se serverem.
- *ConnectionProvider* – vytváří nové instance *Connection*.
- *WriterThread* – vlákno, které zapisuje data do socketu.
- *ReaderThread* – vlákno, které čte data ze socketu.
- *MessageConsumer* – konzumer, který čeká na určitou odpověď serveru.

Dále obsahuje balíček *protocol*, který obsahuje třídy reprezentující různé odpovědi serveru - např.: *LogInResponse*, *GameListResponse*, atd.



Obrázek 2.2: Závislosti důležitých tříd.

2.2.3 Ukázka práce se třídou Connection

Následující příklad ukazuje práci se třídou Connection a zároveň způsob, jakým jsou realizovány integrační testy.

```
@Test(timeout = DEFAULT_TIMEOUT)
public void testLogOut() throws Exception {
    try (Connection connection = ConnectionProvider.connect()) {

        Future<LogInResponse> inRes = connection.sendLogIn("Test");
        assertThat(inRes.get(), is(LogInResponse.OK));

        Future<LogOutResponse> outRes = connection.sendLogOut();
        assertThat(outRes.get(), is(LogOutResponse.OK));
    }
}
```

2.3 Protokol

Formát Jedna zpráva se skládá z názvu instrukce a obsahu. Název instrukce má vždy 3 znaky. Od obsahu není ničím oddělen.

Jednotlivé zprávy jsou od sebe odděleny svislou čarou: |.

Instrukce vysílané klientem

- *BYE* – vysílá klient nebo server, když se rozhodne ukončit spojení. Server jako poslední instrukci posílá také *BYE*.
- *LIN <login>* – přihlášení uživatele.
  Server odpovídá *LIN <návratový kód>*.
  - OK = 0
  - ALREADY_LOGGED = 1
  - NAME_TOO_SHORT = 2
  - NAME_TOO_LONG = 3
  - UNSUPPORTED_CHARS = 4
  - NAME_ALREADY_IN_USE = 5
- *LOF* – odhlášení uživatele.
  Server odpovídá *LOF <návratový kód>*.
  - OK = 0
  - IN_GAME = 1
- *GLI* – požadavek na seznam her.
  Server odpovídá *GLI* $<id_1>,<výška_1>,<šířka_1>;<id_2>\ldots$
- *GPL <id>* – požadavek na seznam hráčů ve hře.
  Server odpovídá *GPL* $<jméno_1>,<jméno_2>\ldots$
- *GNW <šířka>,<výška>* – požadavek na vytvoření nové hry.
  Server odpovídá *GNW <návratový kód>*.
  - id nové hry > -1
  - WRONG_FORMAT = -1
  - WRONG_SIZE = -2
  - NO_PERMISSIONS = -3
- *GJO <id>* – požadavek na připojení do hry.
  Server odpovídá *GJO <návratový kód>*.
  - OK = 0
  - CANNOT_JOIN = 1

  - NO_PERMISSIONS = 2

- *GST <id>* – požadavek na herní stav určité hry.
  Server odpovídá seznamem dílků, po řádcích: *GST* $<x_1>,<y_1>;<x_2>\ldots$ nebo *GST error*

- *GAC <id dílku*$_1$*>,*$<x_1>,<y_1>$*;<id dílku*$_2$*>* – herní akce.
  Server odpovídá *GAC <návratový kód>*.

  - OK = 0
  - NO_PERMISSIONS = 1
  - WRONG_FORMAT = 2
  - WRONG_PIECE = 3

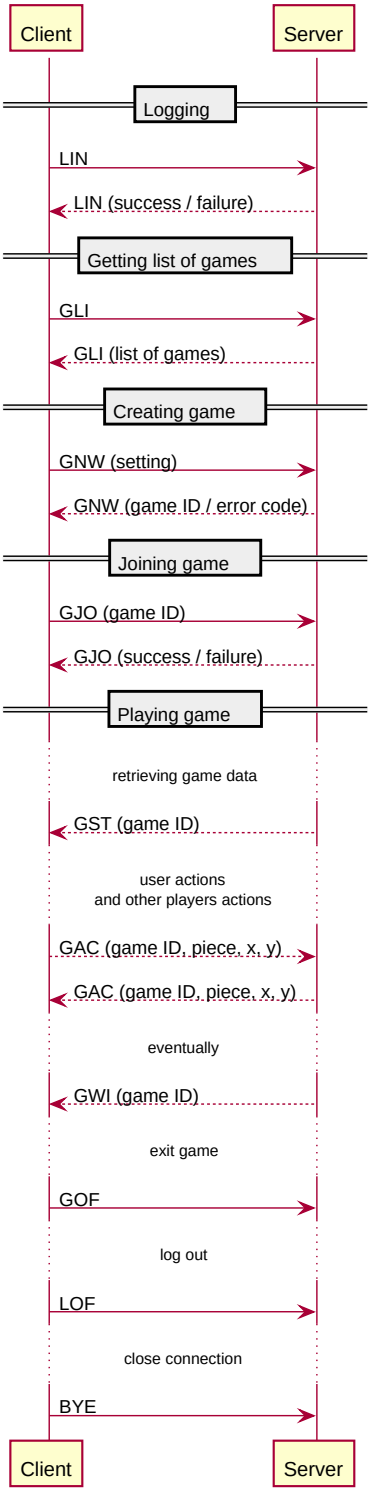
- *GOF* – odchod ze hry.

Instrukce vysílané serverem

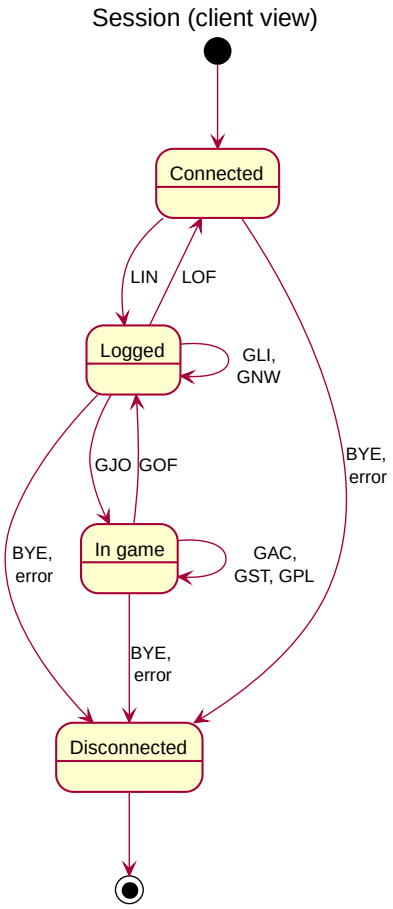
- *PIN* – pro ověření spojení a validního klienta, vysílá server, klient odpovídá také *PIN*.

- *GUP <id dílku*$_1$*>,*$<x_1>,<y_1>$*;<id dílku*$_2$*>* – aktualizace herního stavu, seznam změněných dílků.

- *GWI* – rozesílá server hráčům při úspěšném dokončení hry.



Obrázek 2.3: Sekvenční diagram.



Obrázek 2.4: Stavový diagram.

Kapitola 3

Uživatelská dokumentace

3.1 Pravidla hry

Každý hráč má po přihlášení možnost připojit se ke hře nebo založit novou. Počet her je však omezen na 8. Po dokončení hry se jeden slot uvolní. Dokončení hry znamená složení dílků tak, aby do sebe zapadaly. K jedné hře se může připojit libovolný počet hráčů. Hráči se mohou v libovolnou chvíli odpojit i připojit.

3.2 Server

Sestavení Pro sestavení je vyžadován GNU Linux. Sestavení proběhne po zadání příkazu *make* v kořenovém adresáři serveru.

Spuštění Pro spuštění je vyžadován GNU Linux. Spuštění proběhne po zadání příkazu *./server-dist port*. Pokud nebude port zadán, bude použit port s číslem 8076.

Ukončení Doporučený způsob ukončení je kombinací *ctrl + c*.

Statistiky Při správném ukončení se statistiky uloží do souboru *stats.txt* v aktuálním adresáři.

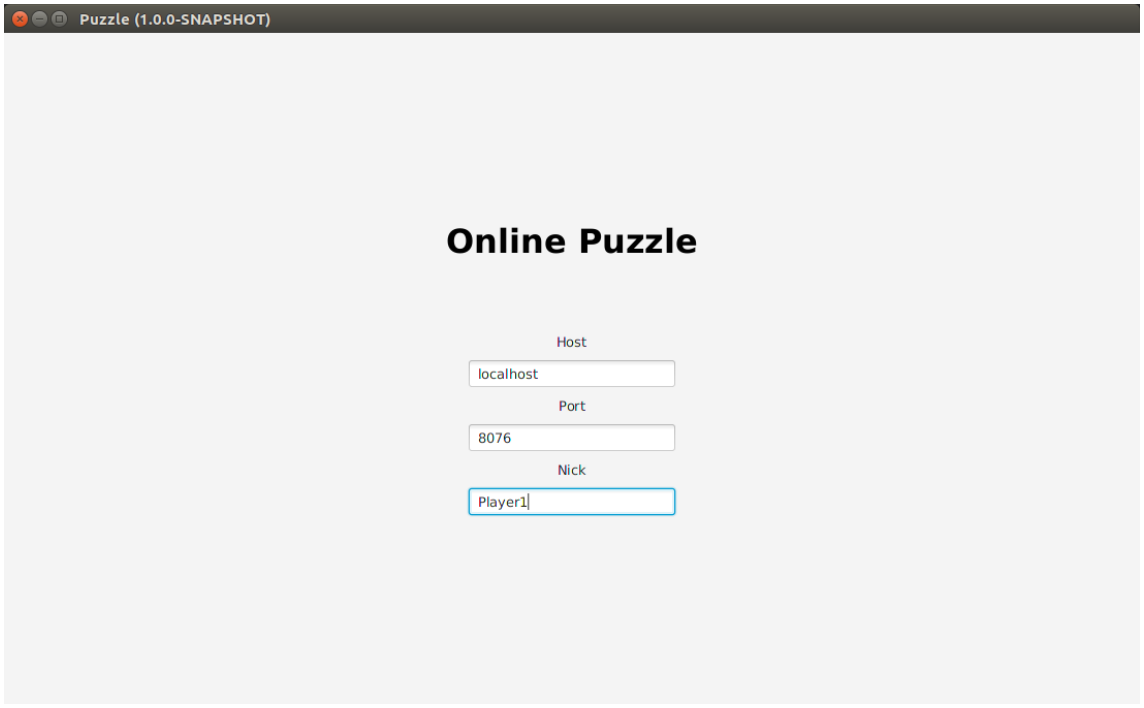
Logování Ve výchozím stavu je logování prováděno na standardní výstup. Pro logování do souboru musíme program spustit následovně: *./server-dist port >log.txt*

3.3 Klient

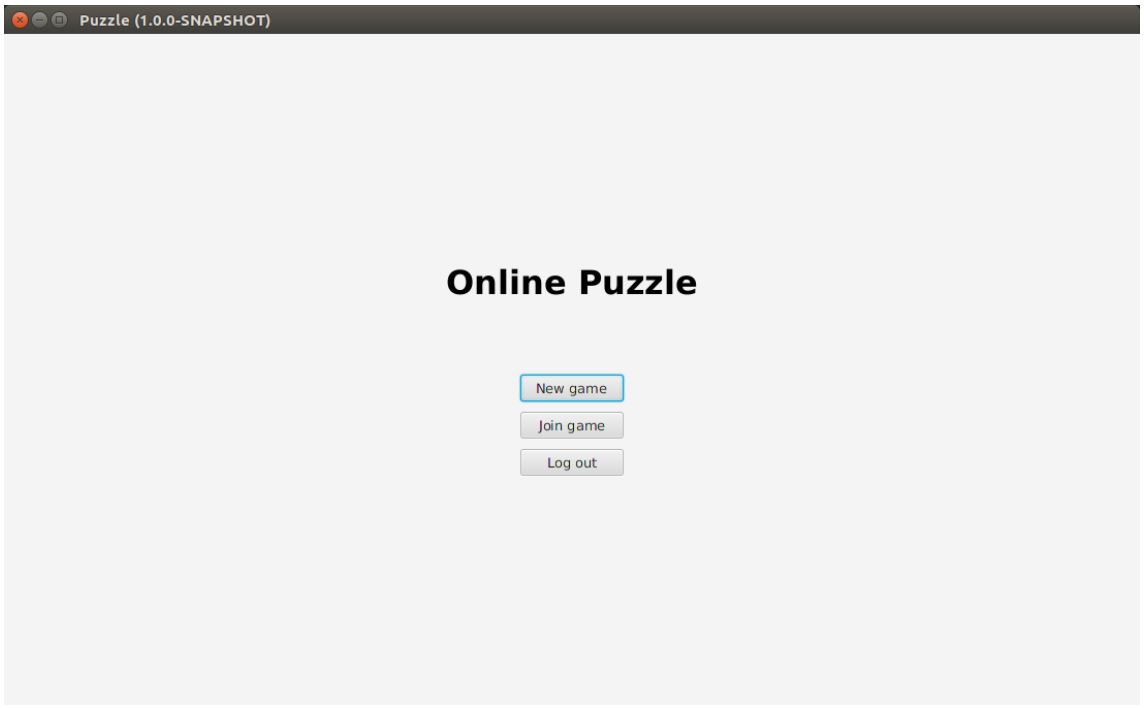
Sestavení Pro sestavení je vyžadován nástroj JDK 8. Sestavení proběhne po zadání *gradlew* nebo *gradlew.bat* do příkazové řádky v kořenovém adresáři klienta.

Spuštění Spuštění vyžaduje JRE 8. Program se spustí zadáním *java -jar program.jar*.

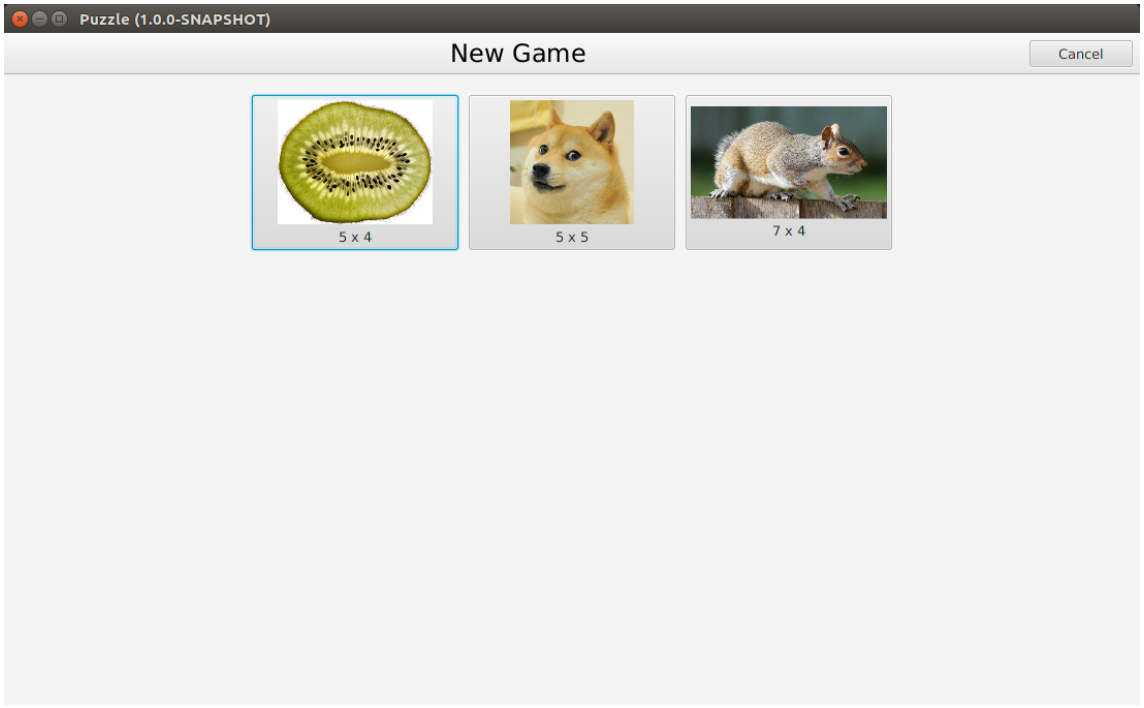
Logování Ve výchozím stavu je logování chyb prováděno na standardní výstup a dále veškeré logování navíc do souboru *log.txt*.



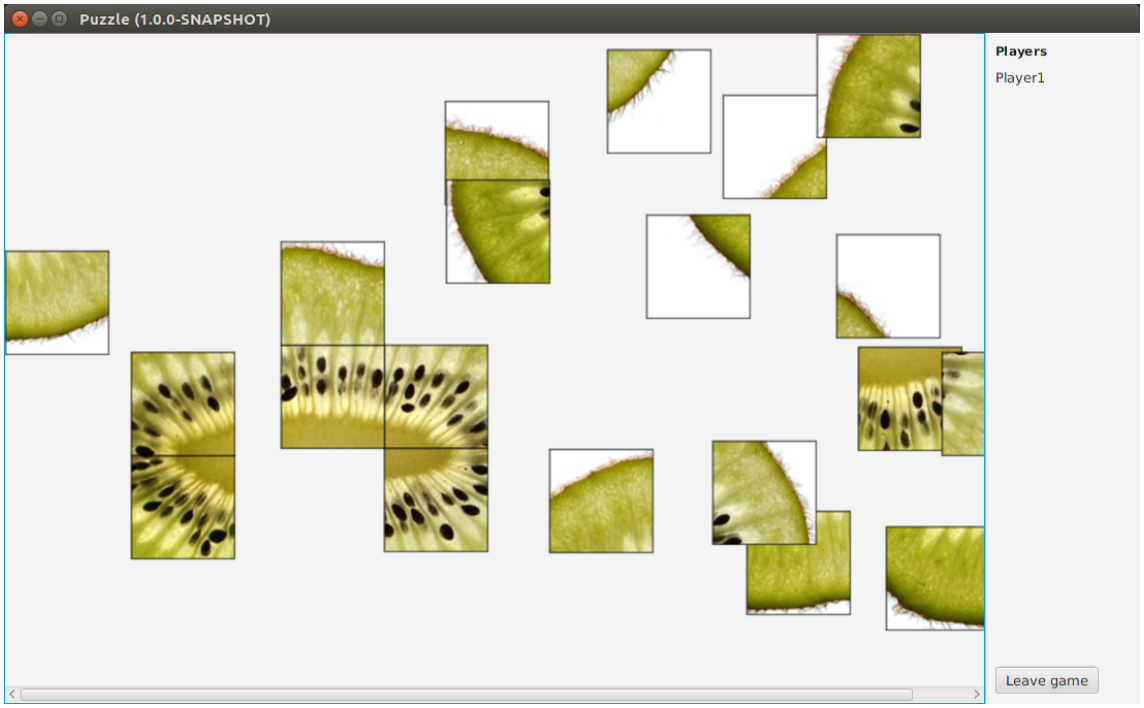
Obrázek 3.1: Stránka s přihlášením.



Obrázek 3.2: Stránka s menu.



Obrázek 3.3: Stránka s vytvořením nové hry.



Obrázek 3.4: Stránka se hrou.