Izveštaj analize projekta

O projektu

- Projekat koji će biti analiziran je *Side scroller*, koji se može naći na sledećoj adresi: https://gitlab.com/matf-bg-ac-rs/course-rs/projects-2020-2021/28-side-scroller.
- Primena alata biće izvršena nad main granom, i commit-om čiji je heš: 97026cfbc3002017bfd9424e3fed3d946bdeddaa
- Side scroller predstavlja igricu 2D karaktera, inspirisana video igrama poput *Megaman* i *Super Mario*, gde igrač treba da stigne do kraja nivo-a, pobedi neprijatelje, i spasi princezu!
- Implementacija projekta je urađena uz pomoć **C++** i **Qt** okruženja.

Valgrind - Memcheck

- Prvi alat koji je bio korišćen pri analizi projekta jeste *Memcheck*, alat programa *Valgrind*, koji vrši detekciju memorijskih grešaka datog programa.
- Build system koji je bio korišćen za projekat je **qmake**, zato da bismo izgradili projekat, pozivamo sledeće komande:

```
qmake ../side-scroller/side-scroller.pro
make
```

- Bitno je napomenuti da u *Makefile-u* su dodate opcije **-g -O0**, kako bi se kod preveo u debug mode-u, i bez optimizacija.
- Ako bismo pokrenuli analizu memorije nad trenutnim programom, *memcheck* bi previše pažnje posvetio Qt funkcijama, što nije poželjno (želimo da vidimo da li naš kod napisan pravi probleme sa memorijom). Iz tih razloga, mock-ujemo main funkciju, tako da izgleda ovako:

```
int main(int argc, char *argv[]) {
    QApplication app(argc, argv);
//  Game::instance()->show();
//  return app.exec();

    Game* g = Game::instance();

    g->start();

    for (int i = 0; i < 10; ++i) {
        g->advance();
    }

    g->gameover();

    delete g;

    app.exit(0);
    return 0;

}
```

tj. želimo da testiramo samo rad najveće klase programa, klase **Game**.

- Pokrenuta je analiza memorije, pozivanjem sledeće komande:

```
valgrind --leak-check=full --track-origins=yes --log-
file=valgrind_output_game.txt  ../28-side-scroller/build/side-scroller
```

Izlaz je preusmeren u fajl **valgrind_output_game.txt**, i korišćene su dodatne opcije *--leak-check=full* i *--track-origins=yes* za detaljnije opise.

- Sadržaj fajl-a:

```
==12013== Memcheck, a memory error detector
==12013== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==12013== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright
info
==12013== Command: ../28-side-scroller/build/side-scroller
==12013== Parent PID: 2621
==12013==
==12013==
==12013== HEAP SUMMARY:
==12013==     in use at exit: 234,076,404 bytes in 19,618 blocks
==12013==   total heap usage: 206,144 allocs, 186,526 frees, 266,224,429
bytes allocated
==12013==
==12013== 288 (256 direct, 32 indirect) bytes in 1 blocks are definitely
lost in loss record 697 of 1,050
==12013==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-
gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
```

```
==12013==    by 0x920C2F4: ??? (in /usr/lib/x86_64-linux-
gnu/libfontconfig.so.1.12.0)
==12013==    by 0x920C9B8: ??? (in /usr/lib/x86_64-linux-
gnu/libfontconfig.so.1.12.0)
==12013==    by 0x920DFDC: ??? (in /usr/lib/x86_64-linux-
gnu/libfontconfig.so.1.12.0)
==12013==    by 0x921506C: ??? (in /usr/lib/x86_64-linux-
gnu/libfontconfig.so.1.12.0)
==12013==    by 0x97EC9D9: ??? (in /usr/lib/x86_64-linux-
gnu/libexpat.so.1.6.11)
==12013==    by 0x97ED6AF: ??? (in /usr/lib/x86_64-linux-
gnu/libexpat.so.1.6.11)
==12013==    by 0x97EAB82: ??? (in /usr/lib/x86_64-linux-
gnu/libexpat.so.1.6.11)
==12013==    by 0x97EC04D: ??? (in /usr/lib/x86_64-linux-
gnu/libexpat.so.1.6.11)
==12013==    by 0x97EFDBF: XML_ParseBuffer (in /usr/lib/x86_64-linux-
gnu/libexpat.so.1.6.11)
==12013==    by 0x9212F42: ??? (in /usr/lib/x86_64-linux-
gnu/libfontconfig.so.1.12.0)
==12013==    by 0x921337B: ??? (in /usr/lib/x86_64-linux-
gnu/libfontconfig.so.1.12.0)
==12013==
==12013== 3,812 (16 direct, 3,796 indirect) bytes in 1 blocks are
definitely lost in loss record 922 of 1,050
==12013==    at 0x483BE63: operator new(unsigned long) (in /usr/lib/x86_64-
linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==12013==    by 0x11ACC6: Game::Game(QGraphicsView*) (in
/home/user/Desktop/2023_Analysis_side-scroller/28-side-scroller/build/side-
scroller)
==12013==    by 0x11AE16: Game::instance() (in
/home/user/Desktop/2023_Analysis_side-scroller/28-side-scroller/build/side-
scroller)
==12013==    by 0x114FCE: main (in /home/user/Desktop/2023_Analysis_side-
scroller/28-side-scroller/build/side-scroller)
==12013==
==12013== 2,359,296 bytes in 1 blocks are possibly lost in loss record
1,049 of 1,050
==12013==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-
gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==12013==    by 0x506D6AF: QImageData::create(QSize const&, QImage::Format)
(in /usr/lib/x86_64-linux-gnu/libQt5Gui.so.5.12.8)
==12013==    by 0x506D7DE: QImage::QImage(QSize const&, QImage::Format) (in
/usr/lib/x86_64-linux-gnu/libQt5Gui.so.5.12.8)
==12013==    by 0x506D81C: QImage::QImage(int, int, QImage::Format) (in
/usr/lib/x86_64-linux-gnu/libQt5Gui.so.5.12.8)
==12013==    by 0x5070F05: QImage::convertToFormat_helper(QImage::Format,
QFlags<Qt::ImageConversionFlag>) const (in /usr/lib/x86_64-linux-
gnu/libQt5Gui.so.5.12.8)
==12013==    by 0x50B2066:
QRasterPlatformPixmap::createPixmapForImage(QImage,
QFlags<Qt::ImageConversionFlag>) (in /usr/lib/x86_64-linux-
gnu/libQt5Gui.so.5.12.8)
==12013==    by 0x50B235D: QRasterPlatformPixmap::fromImage(QImage const&,
```

```
QFlags<Qt::ImageConversionFlag>) (in /usr/lib/x86_64-linux-
gnu/libQt5Gui.so.5.12.8)
==12013==    by 0x50B1356: QPlatformPixmap::fromFile(QString const&, char
const*, QFlags<Qt::ImageConversionFlag>) (in /usr/lib/x86_64-linux-
gnu/libQt5Gui.so.5.12.8)
==12013==    by 0x50AAB36: QPixmap::load(QString const&, char const*,
QFlags<Qt::ImageConversionFlag>) (in /usr/lib/x86_64-linux-
gnu/libQt5Gui.so.5.12.8)
==12013==    by 0x11AD01: Game::Game(QGraphicsView*) (in
/home/user/Desktop/2023_Analysis_side-scroller/28-side-scroller/build/side-
scroller)
==12013==    by 0x11AE16: Game::instance() (in
/home/user/Desktop/2023_Analysis_side-scroller/28-side-scroller/build/side-
scroller)
==12013==    by 0x114FCE: main (in /home/user/Desktop/2023_Analysis_side-
scroller/28-side-scroller/build/side-scroller)
==12013==
==12013== LEAK SUMMARY:
==12013==    definitely lost: 272 bytes in 2 blocks
==12013==    indirectly lost: 3,828 bytes in 8 blocks
==12013==      possibly lost: 2,359,296 bytes in 1 blocks
==12013==    still reachable: 231,713,008 bytes in 19,607 blocks
==12013==         suppressed: 0 bytes in 0 blocks
==12013== Reachable blocks (those to which a pointer was found) are not
shown.
==12013== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==12013==
==12013== For lists of detected and suppressed errors, rerun with: -s
==12013== ERROR SUMMARY: 3 errors from 3 contexts (suppressed: 0 from 0)
```

- Nakon analiziranja izveštaja datog pokretanjem *memcheck-a*, možemo da zaključimo nešto jako bitno. Naime, iz greške na koju nam ukazuje alat:

```
==12013== 3,812 (16 direct, 3,796 indirect) bytes in 1 blocks are
definitely lost in loss record 922 of 1,050
==12013==    at 0x483BE63: operator new(unsigned long) (in /usr/lib/x86_64-
linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==12013==    by 0x11ACC6: Game::Game(QGraphicsView*) (in
/home/user/Desktop/2023_Analysis_side-scroller/28-side-scroller/build/side-
scroller)
==12013==    by 0x11AE16: Game::instance() (in
/home/user/Desktop/2023_Analysis_side-scroller/28-side-scroller/build/side-
scroller)
==12013==    by 0x114FCE: main (in /home/user/Desktop/2023_Analysis_side-
scroller/28-side-scroller/build/side-scroller)
```

Dolazimo do zaključka da memorija zauzeta, nikada nije bila oslobođena, što je čudno, pošto je u mock-u main funkcije dodata linija koja bi trebala da oslobodi memoriju koju je zauzela Main klasa, što nas dovodi do zaključka da postoji problem sa *destruktorom* klase, što i jeste bio problem. **Skoro nijedna od klasa u**

programu nije imala ispisan destrutor! Detaljna analiza koda nas dovodi do zaključka da bi sledeći destrutori za klase *Main* i *Button* trebali da nam pomognu da se rešimo curanja memorije:

```
Game::~Game()
{

    delete health_bar;
    delete player;
    delete pause_screen;
    delete death_screen;

    scene->clear();
    delete scene;
}
```

```
Button::~Button()
{
    delete text;
}
```

- Nakon ponovnog pokretanja alata, pri čemu je izlaz preokrenut u fajl **valgrind_output_game_with_destructor.txt**, primetićemo da smo se rešili ovih problema. U izveštaju nagoveštava da postoje još curenja memorije, ali, ako bismo uporedili statistike pre i posle dodavanja *destruktora*, primetićemo da smo sa:
    - 3,828 *indirectly lost* bitova spali na 32
    - 2,359,296 *possibly lost* bitova spali na 0
    - 231,713,008 *still reachable* bitova spali na 448,742
- **Zaključak**: Pri izradi programa, nije bilo velike brige u oslobađanju memorije, ali nakon dodavanja par destruktora, imali smo veliki pad u curenju memorije!

Valgrind - Massif

- *Massif* je alat *Valgrind-a* koji daje informacije o preseku stanja hipa, u toku izvršavanja programa. Pokrenut je korišćenjem komande:

  ```
  valgrind --tool=massif ../../28-side-scroller/build/side-scroller
  ```

- Time dobijamo fajl *massif.out.14753*, samo što čitanje ovog fajla nije lako za čoveka, zato koristimo dodatnu komandu, da bi napravio nama čitljiviji fajl:

  ```
  ms_print massif.out.14753 > massif_graph.txt
  ```

- Tj. preusmeravamo izlaz u novi fajl, *massif_graph.txt*! Graf koji je dobijem datom komandom:

```
--------------------------------------------------------------------------------
Command:            ../../28-side-scroller/build/side-scroller
Massif arguments:   (none)
ms_print arguments: massif.out.14753
--------------------------------------------------------------------------------

    MB
224.2^
:
|     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@:::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
|     @                                                                     :::@:@#@
+--------------------------------------------------------------------------------->Gi
0                                                                                3.349

Number of snapshots: 89
 Detailed snapshots: [2, 5, 11, 13, 15, 16, 36, 47, 53, 55 (peak), 65, 75,
85]

--------------------------------------------------------------------------------
  n        time(i)   total(B)   useful-heap(B) extra-heap(B)    stacks(B)
--------------------------------------------------------------------------------
  0              0          0                0             0            0
  1      8,742,004         40               32             8            0
  2     19,195,457    429,512          392,111        37,401            0
  3     27,344,819    632,392          565,696        66,696            0
  4     36,275,266    708,256          602,445       105,811            0
  5     47,333,858    847,784          707,593       140,191            0
  6     56,041,105    787,696          637,893       149,803            0
  7     63,585,081    850,928          696,274       154,654            0
```

- Iz izveštaja možemo da vidimo da je massif napravio 89 preseka, i izvojio nam je neke od njih. Vrhunac je dostignut u 55, kada je potrošnja hipa bila oko 224.2MB, ali, treba napomenuti da kada se popne do te potrošnje, retko pada. Predpostavka je da program troši toliko veliku količinu hipa, pošto cela mapa je sastavljena od objekata različitih vrsta, pri čemu je svaki objekat predstavljen slikom. Samim tim, mnogo će više memorije trošiti, nego standardni program.

- Ponovo je vršena ista analiza programa, samo što sada pratimo i potrošnju steka. Pozivamo istu komandu, samo što dodajemo opciju *--stack=yes*:

```
valgrind --tool=massif --stack=yes ../../28-side-scroller/build/side-scroller

ms_print massif.out.15379 > massif_graph_2.txt
```

- Rezultat koji dobijamo:

```
--------------------------------------------------------------------------------
Command:            ../../28-side-scroller/build/side-scroller
Massif arguments:   --stacks=yes
ms_print arguments: massif.out.15379
--------------------------------------------------------------------------------


    MB
224.1^                                                                        #
|    @@:@::::::::@::::::::::::::::::::::@:::::::::::::@@:::::::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
|    @ :@:: :: ::@:: :::::: ::: ::::: ::@: :::: ::::: @ ::: :::::@:::@:#:
+----------------------------------------------------------------------->Gi
0                                                                   3.333
```

```
Number of snapshots: 63
 Detailed snapshots: [4, 6, 13, 32, 43, 52, 58, 60 (peak)]

--------------------------------------------------------------------------------
  n        time(i)      total(B)   useful-heap(B) extra-heap(B)  stacks(B)
--------------------------------------------------------------------------------
  0              0             0                0             0         0
  1     60,187,272       805,328          648,617       153,703     3,008
  2    110,938,008     3,577,280        3,401,312       172,368     3,600
  3    183,288,691     3,603,488        3,425,264       172,920     5,304
  4    229,698,589   230,323,976      230,144,950       175,394     3,632
  5    313,548,282   230,497,104      230,318,040       175,472     3,592
  6    369,152,517   230,497,176      230,318,040       175,472     3,664
  7    430,328,999   230,497,104      230,318,040       175,472     3,592
  8    487,827,185   230,498,384      230,318,040       175,472     4,872
  9    559,231,383   230,498,384      230,318,040       175,472     4,872
 10    621,718,728   230,498,384      230,318,040       175,472     4,872
 11    704,754,442   230,497,104      230,318,040       175,472     3,592
 12    767,018,449   230,497,176      230,318,040       175,472     3,664
 13    808,457,781   230,498,376      230,318,040       175,472     4,864
```

- Kao što možemo da vidimo, zauzeće memorije je slično kao u prošlom primeru, bez mnogo skokova u memoriji. Kada se dospe do 224MB, ne oscilira mnogo od te vredosti. Čak i potrošnja steka uvek ostaje između 3000 i 4000 bajtova.
- **Zaključak**: Hip se relativno odgovorno koristi. Jeste da postoji veliki skok do 224MB, ali to je posledica same prirode programa, i kad se dospe do te velike potrošnje, retko se diže ili spušta.

Unit testing - QtTest

- Napravljen je specijalan projekat u okruženju *QtCreator*, gde će se nalaziti naši **unit testovi** za projekat *SideScroller*.
- Klase za koje su pisani unit testovi:
  - Game
  - Player
  - Enemy
- Testovi koji su bili napisani za klasu **Game**:
  - void **testIfGameIsSingleInstance()**
    - Testira da li je moguće da imamo dve instance klase Game (ne bi trebalo da je moguće, jer se koristi *Singleton design pattern*).
  - void **testIfGameStateIsReadyAtTheBegining**()
    - Testira da li je atribut klase Game *cur_state* na početku izvršavanja jednak **READY**.
  - void **testIfGameStateIsRunningWhenGameStarts**()
    - Testira da li je atribut *cur_state* klase Game jednak **RUNNING** kada je pokrenut metod *start()*.
  - void **testIfWhenGameStartsPlayerIsInitialised**()

        - Testira da li je atribut *player* klase Game inicijalizovan kad je pokrenut metod *start()*.
    - void **testWhenGameIsInMainMenuThereAre3Buttons**()
        - Testira da li kad je pokrenut metod *displayMainMenu()* klase Game su prikazana 3 dugmeta.
    - void **testWhenPlayerDiesStateIsGameOver**()
        - Testira da li kad atribut *player* klase Game pozove sopstven metod *die()*, da je atribut klase Game *cur_state* jednak **GAME_OVER**.
    - void **testWhenTogglePauseGameStateIsPause**()
        - Testira da li kad se pozove metod *tooglePause()* klase Game, da atribut *cur_state* postane **PAUSE**.
        - Treba napomenuti da postoji štamparska greška u imenu metoda. Naime, trebalo bi da se metod zove *togglePause()*.
    - void **testWhenGameStateIsReadyAndButtonPressKeySStateBecomesRunning**()
        - Testira da li kad je atribut klase Game *cur_state* jednak **READY** i pritisne se dugme **S**, *cur_state* postaje **RUNNING**.
    - void **testWhenGameStateIsGameOverAndButtonPressKeyRStateBecomesReady** ()
        - Testira da li kad je atribut klase Game *cur_state* jednak **GAME_OVER** i pritisne se dugme **R**, *cur_state* postaje **READY**.

- Testovi koji su bili napisani za klasu **Player**:

    - void **testIfDefaultXAndYAreCorrect**()
        - Testira da li kad se konstruiše objekat klase Player, da li se postavlja na korektno mesto (tj. da li su mu dobro postavljene *x* i *y* kordinate).
    - void **testIfPlayerLosesCorrectHP**()
        - Testira da li kad se pozove metod *damagePlayer()* klase Player, da li se igraču smanji atribut *health* za toliko.
    - void **testIfPlayerDiesWhenHPNegative**()
        - Testira da li kad atribut *health* klase Player postane <= 0, da li atribut *dying* postaje **true**.
    - void **testIfSpeedIsDoubledWhenRunning**()
        - Testira da li kad se postavi vrednost atributa *running* klase Player na **true**, da li *moving_speed* duplira.
    - void **testIfJumpingIsSetToTrueWhenThePlayerStartsJumping**()
        - Testira da li kad se pozove metod *jumping()* klase Player, atribut *jumping* postane **true**.

- Testovi koji su bili napisani za klasu **Enemy**:

    - void **testIfWhenEnemyMovesToTheRightItsXValueIsGreater** ()
        - Testira da li kad se proizvoljna instanca klase Enemy pomeri u desnom smeru, povećava mu se *x* kordinata.
    - void **testIfWhenEnemyMovesToTheLeftItsXValueIsLesser**()
        - Testira da li kad se proizvoljna instanca klase Enemy pomeri u levom smeru, smanjuje mu se *x* kordinata.
    - void **testIfWhenEnemySpawnsHeIsFallingAndHisYIsGreaterThanBefore** ()
        - Testira da li dok proizvolja instanca klase Enemy pada, da mu se povećava *y* kordinata.
    - void **testIfWhenEnemyJumpsHisYIsLesserThanBefore**()
        - Testira da li dok proizvolja instanca klase Enemy skače, da mu se smanjuje *y* kordinata.

- Bitno je napomenuti da je bilo potrebno refaktorisanje originalnog koda na par mesta, da bi bilo moguće napisati ove *unit testove*. Kao npr:
    - Pomeriti *enum game_state* da bude globalan.
    - Dodati *getter* i *setter* za atribut *cur_state* u klasi Game:

```
void Game::setCur_state(const game_state &value)
{
  cur_state = value;
}

game_state Game::getCur_state() const
{
  return cur_state;
}
```

    - Dodati su *getteri* za atribute *dying* i *moving_speed* u klasi Entity:

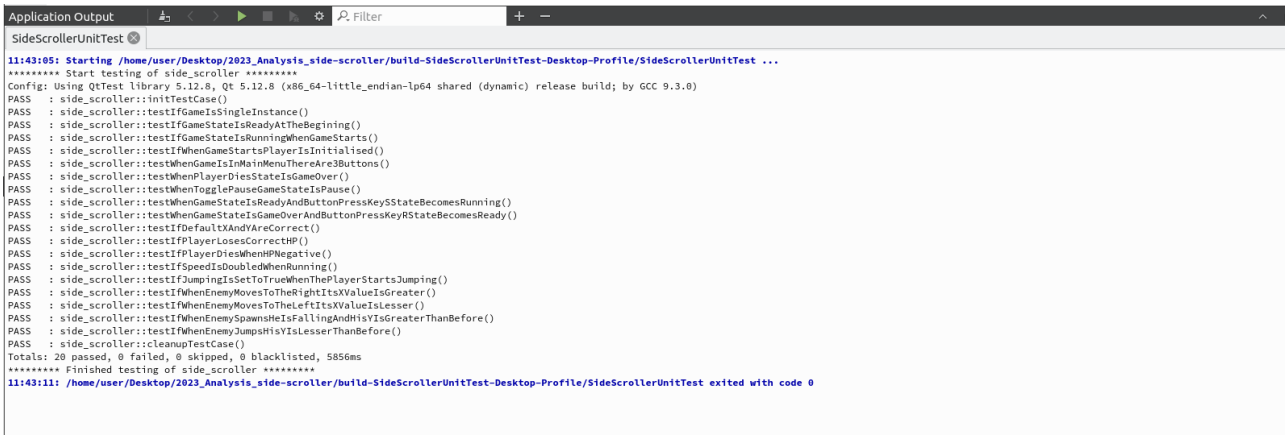
```
bool Entity::getDying() const
{
  return dying;
}

int Entity::getMoving_speed() const
{
  return moving_speed;
}
```

    - Dodati *getteri* i *setteri* u klasi Entity za atribute *jumping* i *falling*:

```
bool Entity::getJumping() const
{
  return jumping;
}

void Entity::setJumping(bool value)
{
  jumping = value;
}

bool Entity::getFalling() const
{
  return falling;
}

void Entity::setFalling(bool value)
{
```

```
    falling = value;
}
```

- Kada pokrenemo projekat, možemo da vidimo da su svi napisati testovi uspešno prošli!



- **Zaključak**: Kod za koji su napisani testovi je dobrog kvaliteta!

Code Coverage - gcov

- Alat *gcov* koristi se za merenje pokrivenosti koda testovima, i koristimo ga nad projektom *SideScroller*.

- Pošto je za build system korišćen *qmake*, potrebno je dodati sledeće linije u .pro fajl:

```
QMAKE_CXXFLAGS += --coverage
QMAKE_LFLAGS += --coverage
```

- Nakon build-a, dobijamo dodatne *.gcno* fajlove, koji se koriste za generisanje samog izveštaja.

- Pokrećeno izvršni fajl projekta, i čim se on završi, i sledeću komandu:

```
lcov --capture --directory . --output-file report.info
```

- Od prethodne komande smo dobili *report.info* fajl, koji nije toliko čitljiv za ljude. Zato pozivamo sledeću liniju:

```
genhtml -o result report.info
```

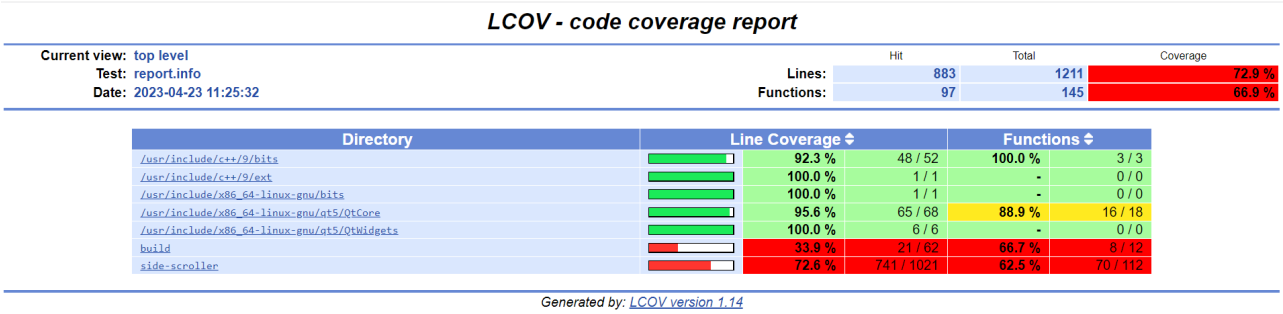
  Koja će nam izgenerisati *.html* fajl, koji nam prikazuje tačno koliko je koda pokriveno testovima.

- Pokretanjem prethodne komande, dobijamo sledeći izlaz:

```
user@irvn:~/Desktop/2023_Analysis_side-scroller/28-side-scroller/build$ genhtml -o result report.info
Reading data file report.info
Found 66 entries.
Found common filename prefix "/home/user/Desktop/2023_Analysis_side-scroller/28-side-scroller"
Writing .css and .png files.
Generating output.
Processing file build/qrc_res.cpp
Processing file build/moc_Game.cpp
Processing file build/moc_Button.cpp
Processing file side-scroller/Bullet.cpp
Processing file side-scroller/EnemyFinalBoss.cpp
Processing file side-scroller/Game.cpp
Processing file side-scroller/EnemySniperJoe.h
Processing file side-scroller/Button.cpp
Processing file side-scroller/Inert.cpp
Processing file side-scroller/main.cpp
Processing file side-scroller/LevelManager.cpp
Processing file side-scroller/HealthBar.h
Processing file side-scroller/Babe.h
Processing file side-scroller/FinalBossBullet1.cpp
Processing file side-scroller/HealthBar.cpp
Processing file side-scroller/Rocket.h
Processing file side-scroller/FinalBossBullet2.h
Processing file side-scroller/Player.h
Processing file side-scroller/EnemySniperJoe.cpp
Processing file side-scroller/Projectile.cpp
Processing file side-scroller/Entity.h
Processing file side-scroller/EnemyPickman.cpp
Processing file side-scroller/SniperJoeBullet.cpp
Processing file side-scroller/EnemyProjectile.cpp
Processing file side-scroller/FinalBossBullet2.cpp
Processing file side-scroller/Entity.cpp
Processing file side-scroller/Projectile.h
Processing file side-scroller/EnemyPickman.h
Processing file side-scroller/FinalBossBullet1.h
Processing file side-scroller/Bullet.h
Processing file side-scroller/EnemyFinalBoss.h
Processing file side-scroller/Object.h
Processing file side-scroller/Block.h
Processing file side-scroller/Block.cpp
Processing file side-scroller/EnemyProjectile.h
Processing file side-scroller/Pickaxe.cpp
Processing file side-scroller/Babe.cpp
Processing file side-scroller/Rocket.cpp
Processing file side-scroller/SniperJoeBullet.h
Processing file side-scroller/SniperJoeBullet.h
Processing file side-scroller/Enemy.h
Processing file side-scroller/Pickaxe.h
Processing file side-scroller/Inert.h
Processing file side-scroller/Enemy.cpp
Processing file side-scroller/Player.cpp
Processing file side-scroller/Object.cpp
Processing file /usr/include/c++/9/bits/basic_string.tcc
Processing file /usr/include/c++/9/bits/move.h
Processing file /usr/include/c++/9/bits/char_traits.h
Processing file /usr/include/c++/9/bits/atomic_base.h
Processing file /usr/include/c++/9/bits/stl_iterator_base_funcs.h
Processing file /usr/include/c++/9/bits/basic_string.h
Processing file /usr/include/c++/9/bits/stl_algobase.h
Processing file /usr/include/c++/9/bits/std_abs.h
Processing file /usr/include/c++/9/ext/new_allocator.h
Processing file /usr/include/x86_64-linux-gnu/bits/string_fortified.h
Processing file /usr/include/x86_64-linux-gnu/qt5/QtCore/qpoint.h
Processing file /usr/include/x86_64-linux-gnu/qt5/QtCore/qarraydata.h
Processing file /usr/include/x86_64-linux-gnu/qt5/QtCore/qrefcount.h
Processing file /usr/include/x86_64-linux-gnu/qt5/QtCore/qlist.h
Processing file /usr/include/x86_64-linux-gnu/qt5/QtCore/qatomic_cxx11.h
Processing file /usr/include/x86_64-linux-gnu/qt5/QtCore/qrect.h
Processing file /usr/include/x86_64-linux-gnu/qt5/QtCore/qflags.h
Processing file /usr/include/x86_64-linux-gnu/qt5/QtCore/qstring.h
Processing file /usr/include/x86_64-linux-gnu/qt5/QtWidgets/qgraphicsview.h
Processing file /usr/include/x86_64-linux-gnu/qt5/QtWidgets/qgraphicsitem.h
Processing file /usr/include/x86_64-linux-gnu/qt5/QtWidgets/qgraphicsscene.h
Writing directory view page.
Overall coverage rate:
  lines......: 72.9% (883 of 1211 lines)
  functions..: 66.9% (97 of 145 functions)
user@irvn:~/Desktop/2023_Analysis_side-scroller/28-side-scroller/build$
```

Tj. odmah u terminalu možemo da vidimo osnovnu statistiku. Naime, *pokrivenost linija koda* je **72.9%**, dok je *pokrivenost funkcija* **66.9%**.

- Ako otvorimo *index.html*, fajl koji je bio izgenerisan prethodnom komandom, u browser-u, možemo da vidimo sledeće:



LCOV - code coverage report

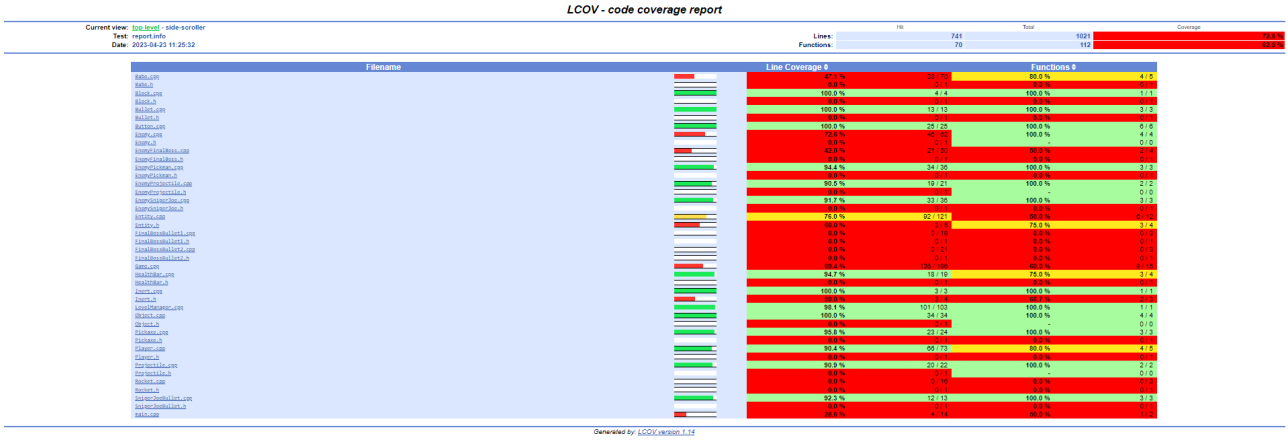
			Hit	Total	Coverage
Current view:	top level	Lines:	883	1211	72.9 %
Test:	report.info	Functions:	97	145	66.9 %
Date:	2023-04-23 11:25:32				

Directory	Line Coverage		Functions	
/usr/include/c++/9/bits	92.3 %	48 / 52	100.0 %	3 / 3
/usr/include/c++/9/ext	100.0 %	1 / 1	-	0 / 0
/usr/include/x86_64-linux-gnu/bits	100.0 %	1 / 1	-	0 / 0
/usr/include/x86_64-linux-gnu/qt5/QtCore	95.6 %	65 / 68	88.9 %	16 / 18
/usr/include/x86_64-linux-gnu/qt5/QtWidgets	100.0 %	6 / 6	-	0 / 0
build	33.9 %	21 / 62	66.7 %	8 / 12
side-scroller	72.6 %	741 / 1021	62.5 %	70 / 112

Generated by: LCOV version 1.14

Tj. tačnu statistiku koliko je koji deo koda prekriven testovima. Nama je najviše od interesa koliko smo testirali sam side-scroller, što ispadne **72.6%** *pokrivenost linija koda*, i **62.5%** *pokrivenost funkcija*. Te brojke nisu za pohvalu, trebale bi da budu obe cifre oko **90%**, ali, treba uzeti u obzir da je sa 20 unit testova podignuta pokrivenost na ovoliko veliki procenat.

Ako uđemo u pokrivenost samo side-scrollera, vidimo sledeće:



Dati prikaz se poklapa sa unit testovima koje smo pisali. Naime, ako pogledamo pokrivenost testovima klasu *Rocket*, vidimo da ona uopšte nije testirana, što i ima smisla, pošto nismo pisali testove za zadatu klasu, i testovi koji su bili napisani, nisu imali veze sa tom klasom. Dok npr. klasa *EnemySniperJoe* ima pokrivenost koda od **91.7%** i pokrivenost funkcija **100%**, zato što dok smo pisali testove za *Enemy* klasu, instancirali smo objekat te klase. Klasa za koju smo najviše pisali testove, *Game*, ima relativno malu pokrivenost, **69.4%** pokrivenost koda, i **60%** prekrivenost funkcija, ali, treba uzeti u obzira da je klasa Game najveća klasa celog projekta, i da bi zahtevala veliki broj testova da bi se pokrila do kraja.

- **Zaključak**: Sa unit testovima koji su napisani nismo postigli ciljanu pokrivenost koda, ali, podigli smo je na vrlo visok nivo. Daljom nadogradnjom, i pisanjem još testova, pokrivenost će se podići!

Clang statička analiza (scan-build)

- scan-build je alat koji omogućava pokretanje statičkog analizatora nad izvornim kodom, i koji je korišćen da bi se uradila statička analiza projekta *SideScroller*.
- Pri pozivanju **qmake-a**, korišćena je komanda:

```
scan-build --use-c++=g++ qmake ../side-scroller/side-scroller.pro
```

- Dok je pri kompajliranju korišćena komanda:

```
scan-build --use-c++=g++ make
```

Bitno je napomenuti da je dodata opcija *-g*, tj. da je kod kompajliran u debug mode-u.

- Bilo je potrebno više vremena nego uobičajno da bi se izgradio projekat (oko 6-7 minuta). Na kraju analize, data nam je komanda koju treba pustiti sledeću, da bismo videli izveštaj:

```
scan-view /tmp/scan-build-2023-04-23-175726-8268-1
```

- Pokretanjem prethodne komande, dobijamo sledeći izveštaj:

build-scan - scan-build results

User:	user@irvm
Working Directory:	/home/user/Desktop/2023_Analysis_side-scroller/28-side-scroller/build-scan
Command Line:	make
Clang Version:	Ubuntu clang version 10.0.1-++20200708122807+ef32c611aa2-1~exp1~20200707223407.61
Date:	Sun Apr 23 17:57:26 2023

Bug Summary

Bug Type	Quantity	Display?
All Bugs	1	☑
Memory error		
Memory leak	1	☑

Reports

Bug Group	Bug Type ▾	File	Function/Method	Line	Path Length			
Memory error	Memory leak	LevelManager.cpp	load	180	7	View Report	Report Bug	Open File

Kao što se može videti, izveštaj nam javlja jednu grešku u projektu. Naime, u klasi *LevelManager*, u **180** liniji, dešava se *curenje memorije*. Otvaranjem koda na zadatom mestu, vidimo sledeće:

```
51    QList<Block *> blocks;
52    QList<Enem *> enemies;
53
54    Babe *princess = new Babe();
                  4 ← Memory is allocated →
55
56
57    while (!in.atEnd()) {
            5 ← Assuming the condition is false →
        6 ← Loop condition is false. Execution continues on line 180 →
58
59      QString line = in.readLine();
60
61      for (auto c : line) {
62
63        if (c == '#') {
64
65          blocks.append(new Block());
66          blocks.last()->setPixmap(
67            QPixmap(":/images/Textures/block_steel_02.png"));
68          blocks.last()->setPos(block_x_coord, block_y_coord);
69
70        } else if (c == '1') {
71
72          blocks.append(new Block());
73          blocks.last()->setPixmap(
74            QPixmap(":/images/Textures/block_rock_01.png"));
75          blocks.last()->setPos(block_x_coord, block_y_coord);
76
77        } else if (c == '2') {
78
79          blocks.append(new Block());
80          blocks.last()->setPixmap(
81            QPixmap(":/images/Textures/block_rock_02.png"));
82          blocks.last()->setPos(block_x_coord, block_y_coord);
83
84        } else if (c == '3') {
85
170                                 enemies.last()->boundingRect().height());
171       }
172
173       block_x_coord += block_dim;
174     }
175
176     block_y_coord += block_dim;
177     block_x_coord = 0;
178   }
179
180   myfile.close();
          7 ← Potential leak of memory pointed to by 'princess'
181
182 } else
183   std::cerr << "Level not implemented";
184
185 return player;
186 }
```

Kao što vidimo, greška je nastala jer je alocirana memorija za objekat klase *Babe*, ali izgubljena je referenca, i nije oslobođen taj memorijski prostor. Sledećim kratkim izmenama klase *LevelManager*, ispravljamo ovaj problem. Najpre postavimo da *LevelManager* čuva referencu na alociranu memoriju:

```
this->princess = new Babe();
```

Pored toga, kreiramo destruktor za klasu u pitanju, da bi se oslobodila ta memorija:

```

```

```
LevelManager::~LevelManager()
{
  delete princess;
}
```

I time rešavamo zadati problem.

- **Zaključak**: Statičkom analizom koda nije pronađeno puno grešaka u projektu, i one koje su nađene su popravljene.