

Fakultet elektrotehnike, strojarstva i brodogradnje, Split

RAČUNALNA GRAFIKA

Gameplay 2D/3D

Projektna dokumentacija

Karlo Pavlović

Sadržaj:

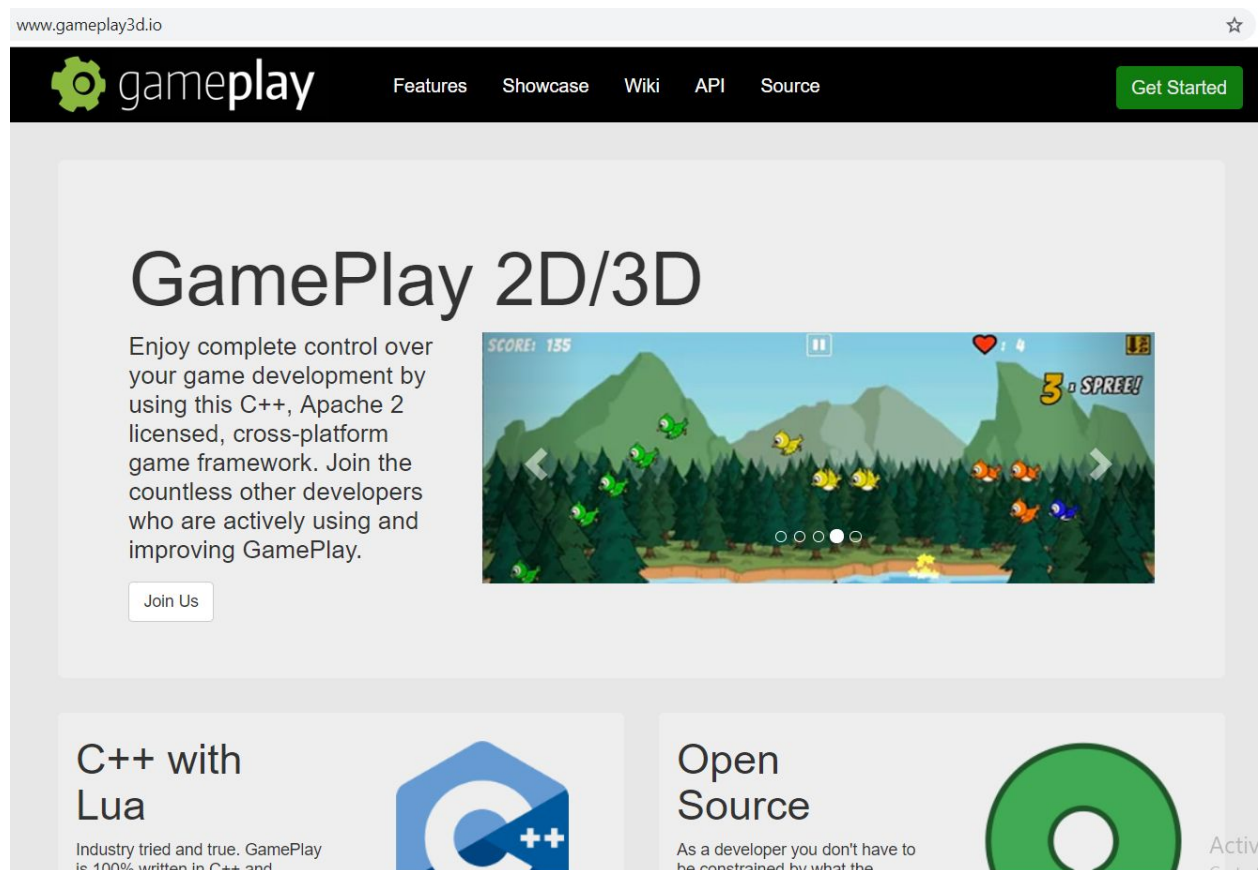
1. Uvod
2. Gameplay 2D/3D
3. Postavljanje projektnog okruženja
4. Izrada novog projekta
5. Projektno okruženje
6. Izrada modela i kamere
7. Kretanje igrača
8. Kretanje neprijatelja
9. Izrada projektila
10. Detekcija sudara
11. Kraj igre i rezultat
12. Pokretanje igre
13. Osvrt na GamePlay alat

1. Uvod :

U okviru ovog seminarskog rada izrađena je desktop igrice Space Invaders. Tema igrice potječe od prvih igara napravljenih uz pomoć računalne grafike. Igrač upravlja svemirskim brodom i uništava nadolazeće neprijatelje. Cilj igre je uništiti što više neprijatelja, a sposobnost igrača je zapisana njegovim krajnjim rezultatom ("Score"). Cilj cjelokupnog projekta je kroz izradu igre procijeniti sposobnosti gameplay3d alata.

2. GamePlay 2D/3D

GamePlay 2D/3D je besplatan open source alat za izradu igara namijenjenih za više platformi. Platforme koje su podržane su: Windows, MacOSX, Linux, iOS, Android. Alat se temelji na C++ jeziku u kombinaciji s Lua skriptnim jezikom. Razvoj GamePlay aplikacija je podržan na Windows, Mac i Linux operacijskim sustavima. Za razvoj na Windows platformi, koja je korištena za izradu ovog projekta, predviđeno je korištenje Visual Studio razvojnog okruženja.



Slika 1. Naslovna stranica GamePlay alata Link: <http://www.gameplay3d.io>

3. Postavljanje projektnog okruženja

Instrukcije za postavljanje projektnog okruženja nalaze se na stranici:
<https://github.com/gameplay3d/GamePlay/wiki/Visual-Studio-Setup>

Postupak je opisan u sljedećim koracima:

1. Preuzeti GamePlay s github repozitorija: <https://github.com/gameplay3d/GamePlay>
2. Pokrenuti install.bat iz glavnog direktorija
3. Instalirati Visual Studio 2015
4. Instalirati Direct SDK
(<http://www.microsoft.com/en-ca/download/details.aspx?id=6812>)
5. Pokrenuti gameplay.sln
6. Build > Build Solution
7. Izabrati projekt za pokretanje, desni klik: Set as StartUp Project
8. Pokrenuti projekt: Debug > Start Debugging

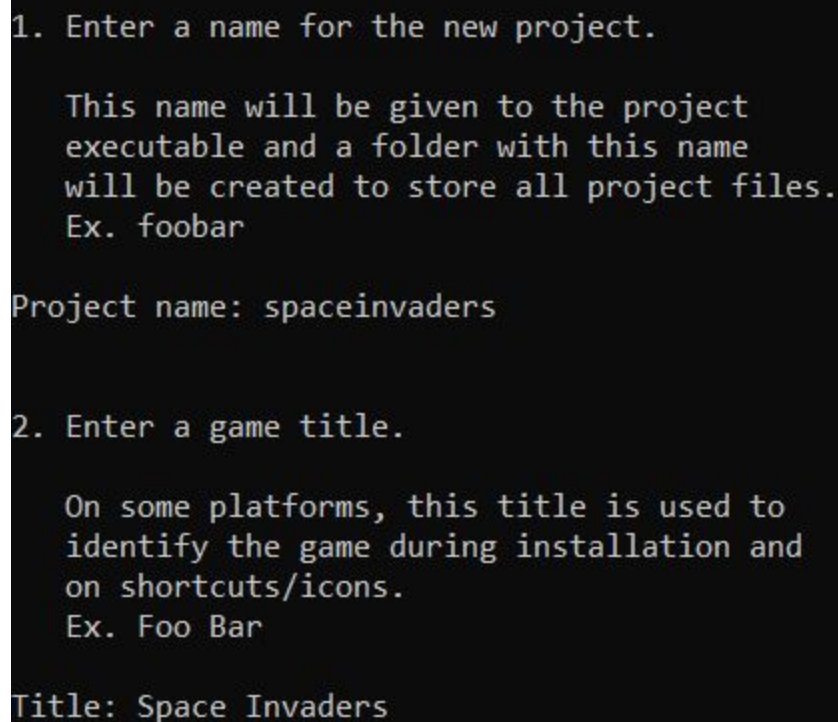
4. Izrada novog projekta

Postupak izrade novog projekta opisan je na stranici:

<https://github.com/gameplay3d/GamePlay/wiki/Creating-a-new-project>

Postupak je opisan u sljedećim koracima:

1. U glavnom direktoriju pokrenuti: newproject.bat
2. Upisati ime projekta
3. Upisati naslov projekta
4. Upisati jedinstvenu oznaku projekta
5. Upisati ime glavne klase
6. Upisati projektni put
7. Unutar Visual Studia, desni klik na solution: Add>Existing project>odabrati projekt



```
1. Enter a name for the new project.

This name will be given to the project
executable and a folder with this name
will be created to store all project files.
Ex. foobar

Project name: spaceinvaders

2. Enter a game title.

On some platforms, this title is used to
identify the game during installation and
on shortcuts/icons.
Ex. Foo Bar

Title: Space Invaders
```

Slika 2. Izrada novog projekta - 1. Dio

3. Enter a unique identifier for your project.

This should be a human readable package name, containing at least two words separated by a period.

Ex. com.example.foobar

Unique ID: com.project.spaceinvaders

4. Enter your game's main class name.

Your initial game header and source file will be given this name and a class with this name will be created in these files.

Ex. FooBarGame

Class name: SpaceInvadersGame

5. Enter the project path.

This can be a relative path, absolute path, or empty for the current folder. Note that a project folder named spaceinvaders will also be created inside this folder.

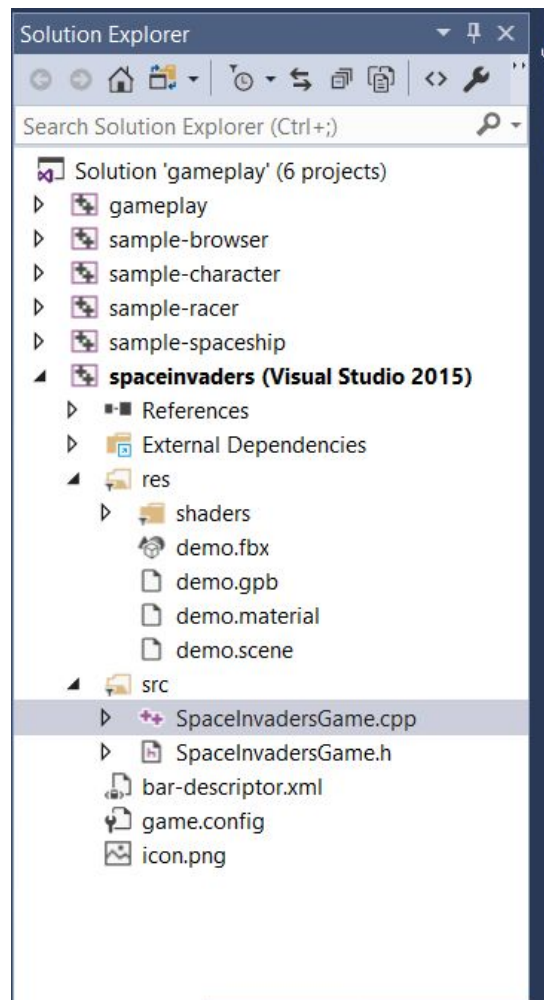
Ex. ./samples

Path: ./project

Slika 3. Izrada novog projekta - 2. Dio

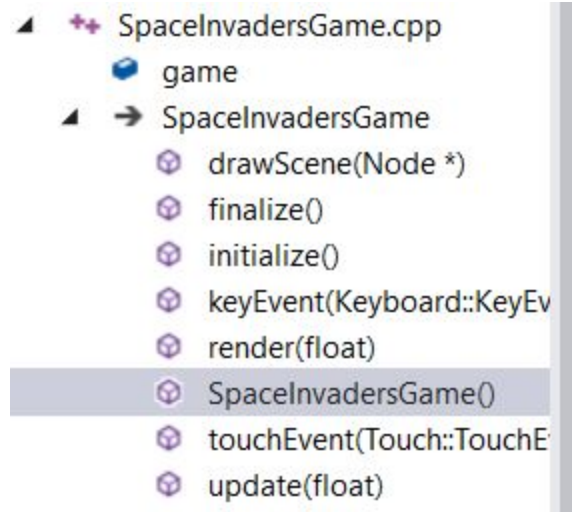
5. Projektno okruženje

Nakon pokretanje skripte za izradu novog projekta izrađeno je novo razvojno okruženje. Dva glavna direktorija su izrađena: src i res. Direktorij res sadrži modele, scene i ostale resurse potrebne za izvođenje projekta. Direktorij src sadži programski kod projekta. Unutar src direktorija izrađena je glavna klasa “SpaceInvadersGame” i njena popratna biblioteka.

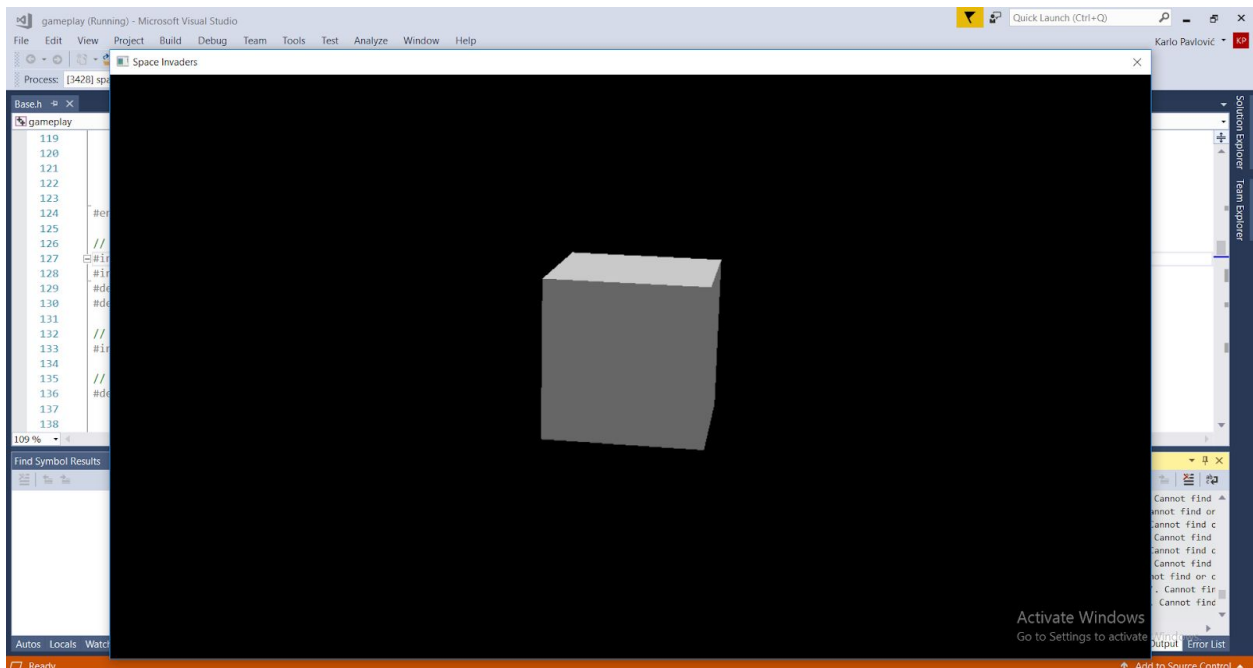


Slika 4. Solution explorer novog projekta

Unutar glavne klase nekoliko osnovnih metoda je napravljeno. Metode initialize() i finalize() su metode koje se pozivaju prilikom pokretanja/gašenja aplikacije. keyEvent i touchEvent su metode koje kontroliraju unos podataka. Metoda drawScene crta sve elemente unutar objekta _scene. Metode update i render služe za programiranje logike i crtanje elemenata.



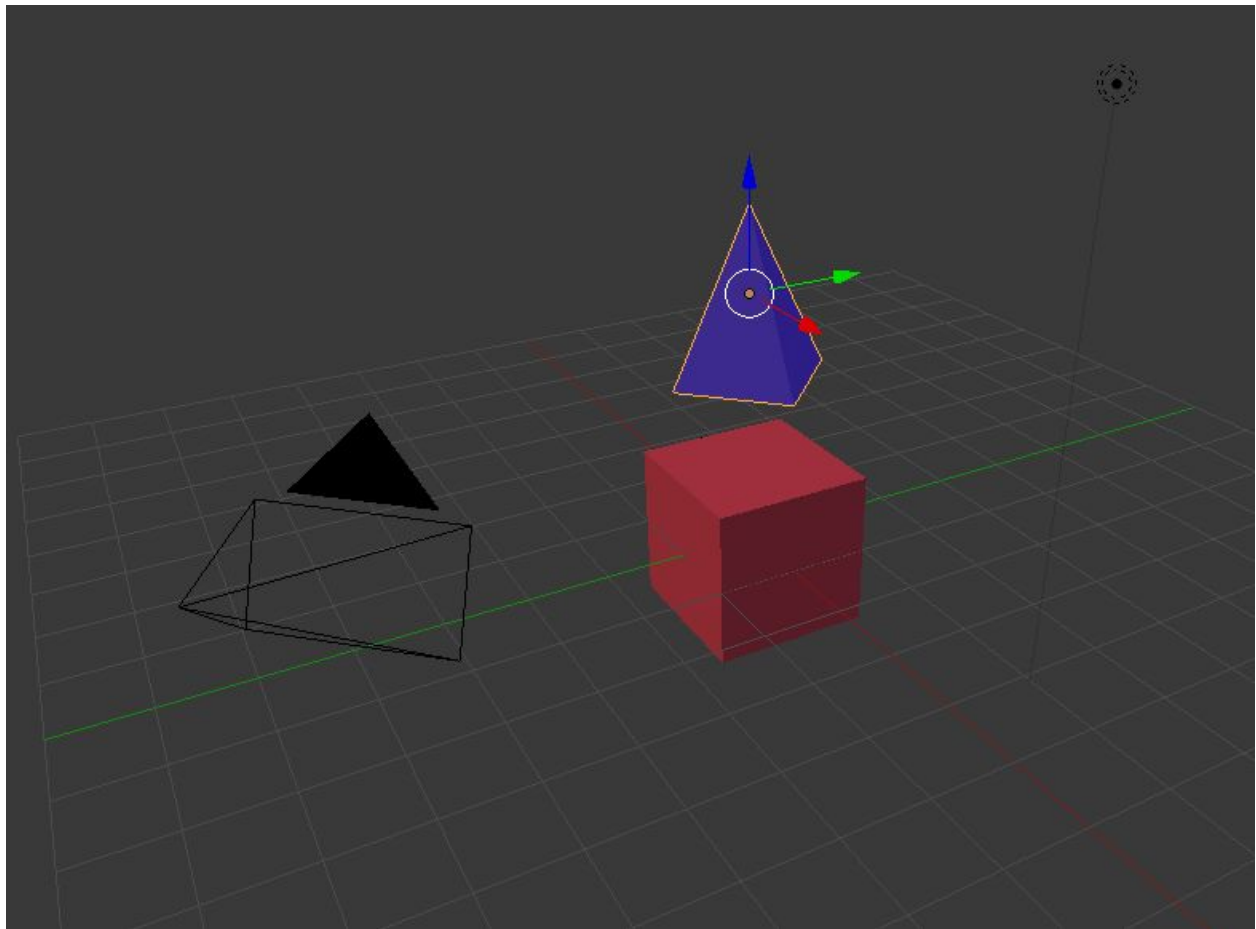
Slika 5. Struktura glavne klase



Slika 6. Prilikom pokretanja novog projekta prikazuje se rotirajuća kocka

6. Izrada modela i kamere

Aplikacija će kositi 2 modela izrađena u blenderu. To su kocka i piramida. Kocka će predstavljati neprijatelje a piramida igrača i projekte.



Slika 7. Izrada elemenata u Blenderu

Nakon izrade potrebno je elemente izvesti iz Blendera u .fbx datoteku. Nakon toga potrebno je koristiti gamepaly-encoder.exe skriptu koja će .fbx datoteku pretvoriti u .gbp datoteku za interno korištenje. Za dohvaćanje modela se koristi demo.scene u kojem su opisani svi node-ovi koji se koriste.



```
demo.scene*  SpaceInvadersGame.cpp  Base.h
1  scene
2  {
3      path = res/scene.gpb
4
5      ambientColor = 1, 1, 1
6
7      node Box
8      {
9          material = res/demo.material#lambert2
10     }
11     node Cone
12     {
13         material = res/demo.material#lambert2
14     }
15
16
17 }
```

Slika 8. Prikaz demo.scene datoteke

Prilikom dodavanja modela potrebno je postaviti njihovu početnu veličinu i položaj.

```
// Get the box model and initialize its material parameter values and bindings
boxNode = _scene->findNode("Box");
playerNode = _scene->findNode("Cone");
boxNode->setScale(1,1,1);
boxNode->setTranslation(0, 0, -30);
playerNode->setScale(1, 1, 1);
playerNode->setTranslation(0, 3, -30);
```

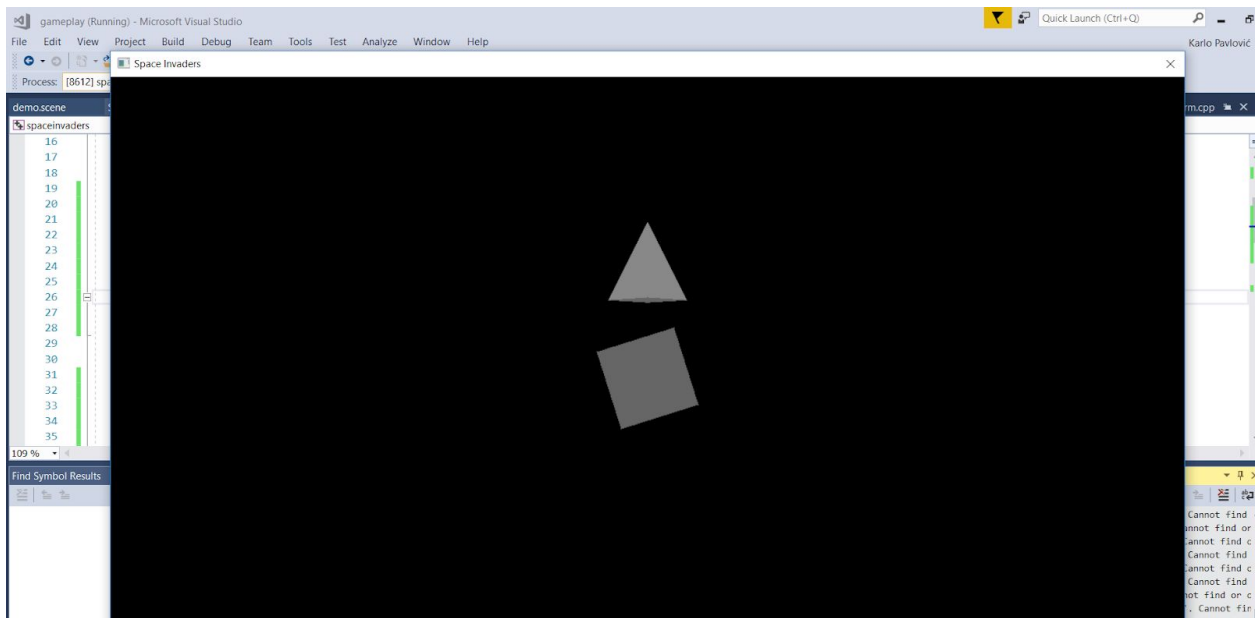
Slika 9. Isječak koda za postavljanje modela

Za postavljanje kamere potrebno je postaviti poziciju kamere i njenu rotaciju tako da gleda prema dolje iz točke 0,0,0. Takva pozicija omogućuje virtualno 2D okruženje na zadanoj površini. Udaljenost od kamere jednostavno određujemo unosom negativne vrijednosti u Z koordinatu objekta.

```
// Set the aspect ratio for the scene's camera to match the current resolution
_scene->getActiveCamera()->setAspectRatio(getAspectRatio());
_scene->getActiveCamera()->setFarPlane(1000);

Node* _cameraNode = _scene->findNode("Camera");
_cameraNode->setTranslation(0,0,0);
_cameraNode->setRotation(0,0,0,0);
```

Slika 10. Isječak koda za postavljanje modela



Slika 11. Izgled aplikacije nakon postavljanja modela i kamere.

7. Kretanje igrača

Igrač kontrolira letjelicu (piramidu) uz pomoć miša.
Prvo je potrebno postaviti poziciju igrača.

```
float posY = 0;
float posX = 0;
float leftBorder = -17;
float rightBorder = 17;
float upBorder = -9;
float downBorder = 9;
```

Slika 12. Inicijalizacija varijabli za kontrolu igrača

Varijable posX i posY označavaju poziciju igrača.
Border varijable ograničavaju igrača unutar zadanog ekrana.

```
void SpaceInvadersGame::update(float elapsedTime)
{
    playerNode->setTranslation(posX, posY, -40);
    if (posX < leftBorder)posX = leftBorder;
    if (posX > rightBorder)posX = rightBorder;
    if (posY < upBorder)posY = upBorder;
    if (posY > downBorder)posY = downBorder;
```

Slika 13. Kontrola igrača unutar update funkcije

Zatim je potrebno “dohvatiti” miša pozivanjem: setMouseCaptured(true),
te dodati mouseEvent koji kontrolira pokrete miša.

```
bool SpaceInvadersGame::mouseEvent(Mouse::MouseEvent evt, int x, int y, int wheelDelta)
{
    if (evt == Mouse::MOUSE_MOVE)
    {
        posX += x * .1;
        posY -= y * .1;
    }
    return true;
}
```

Slika 14. mouseEvent funkcija

8. Kretanje neprijatelja

Objekti neprijatelja se spremaju u listu `enemies`. Dodatne varijable `enemySpeed`, `enemyLeftEdge` i `enemyRightEdge` označavaju brzinu kretanja neprijatelja i provjeravaju da li je neprijatelj došao do lijevog ,odnosno desnog, ruba.

```
std::list<Node*> enemys;  
float enemySpeed = .05;  
bool enemyLeftEdge = false;  
bool enemyRightEdge = false;  
Node* boxNode;
```

Slika 15. Inicijalizacija liste neprijatelja

Maksimalan broj neprijatelja je stalan, dvadeset. U metodi `AddEnemys`, Node "Box" se klonira, postavlja mu se pozicija i veličia, te se dodaje u list neprijatelja. Metoda `AddEnemys` se poziva na početku igre te u slučaju da je broj preostalih neprijatelja jednak nula. (`enemys.size() == 0`)

```
Scene* SpaceInvadersGame::AddEnemys(Scene* s) {  
    for (int i = 0; i < 20; i++)  
    {  
        boxNode = s->findNode("Box")->clone();  
        boxNode->setScale(.8, .8, .8);  
        boxNode->setTranslation((i/2*3)-16, (i%2)*3+6, -40);  
        enemys.push_back(boxNode);  
  
        s->addNode(boxNode);  
    }  
    return s;  
}
```

Slika 16. Dodavanje neprijatelja

Za pomicanje neprijatelja unutar update metode, zadužena je for each petlja koja pomiče sve neprijatelje u zadanom smjeru. Ukoliko neki od neprijatelja dođe do ruba ekrana, smjer pomicanja se promijeni za sljedeći poziv petlje.

```
for each (Node* e in enemys)
{
    float x = e->getTranslationX();
    e->setTranslationX(x + enemySpeed);
    if (x > 16 )enemyRightEdge = true;
    if (x < -16)enemyLeftEdge = true;
}
if (enemyLeftEdge)
{
    enemySpeed = .05;
    enemyLeftEdge = false;
}
if (enemyRightEdge)
{
    enemySpeed = -.05;
    enemyRightEdge = false;
}
```

Slika 17. Pomicanje neprijatelja

9. Izrada projektila

Igrač izbacuje projektele pritiskom na lijevi klik miša. Interval pucanja projektila zadan je štopericom koja je direktno povezana za framerate (jedan metas svakih 5 frame-ova). U trenutku ispaljivanja projektila poziva se metoda `CreatePlayerBullet()` koja stvara novi metak. Novi metak (Node) se sprema u `_scene` i listu `playerBullets`.

```
Scene* SpaceInvadersGame::CreatePlayerBullet(Scene* s) {  
    boxNode = s->findNode("Cone")->clone();  
    boxNode->setScale(.1, .1, .1);  
    boxNode->setTranslation(posX, posY, -40);  
  
    playerBullets.push_back(boxNode);  
  
    s->addNode(boxNode);  
    return s;  
}
```

Slika 18. `CreatePlayerBullet` metoda

Putanja metka počinje s pozicijom igrača u trenutku ispaljivanja te se prilikom svakog frame-a pomiče put vrha ekrana. Ukoliko metak prijeđe vrh ekrana, potrebno ga je automatski uništiti.

Neprijateljski metci su napravljeni na sličan način. Razlike u izradi su sljedeće:

- metci se stvaraju pomoću generatora nasumičnih brojeva
- metci se stavljaju u zasebnu listu `enemyBullets`
- putanja neprijateljskih metaka je prema dnu ekrana i njihovo uklanjanje se poziva ukoliko prekorače dno ekrana


```
for (int b = 0; b < playerBullets.size(); b++)
{
    Node* bullet = playerBullets.front();
    playerBullets.pop_front();
    bullet->translateY(.3);
    if (bullet->getTranslationY() > 10) {
        _scene->removeNode(bullet);
        bullet->release();
    }
    else {
        playerBullets.push_back(bullet);
    }
}
```

Slika 19. Pomak igračevih projektila

10. Detekcija sudara

Za detekciju sudara koristi se funkcija `getBoundingSphere()`. Funkcija dohvaća centar objekta njegov radius. Ukoliko je udaljenost od centra objekta do centra metka manja od radiusa objekta, igra detektira sudar. For petlja prolazi kroz sve objekte i projekte i određuje detekciju sudara.

```
for (int e = 0; e < enemys.size(); e++) {
    Node* enemy = enemys.front();
    enemys.pop_front();
    bool colision = false;
    Vector3 center = enemy->getBoundingSphere().center;
    float radius = enemy->getBoundingSphere().radius;

    for (int b = 0; b < playerBullets.size(); b++)
    {

        Node* bullet = playerBullets.front();
        playerBullets.pop_front();

        if (center.distance(bullet->getBoundingSphere().center) < radius*.8) {

            _scene->removeNode(bullet);
            bullet->release();
            colision = true;

        }
        else {
            playerBullets.push_back(bullet);
        }
    }
}
```

Slika 20. Detekcija sudara

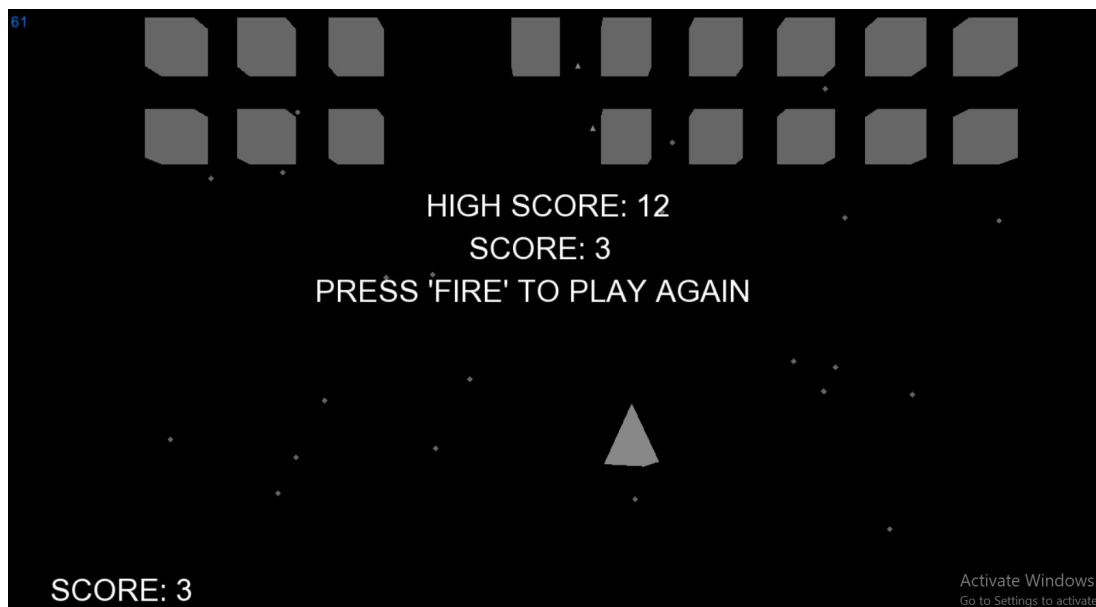
Ukoliko je detektiran sudar, projektil i objekt se uklanjaju, te se 'score' povećava za 1.

11. Kraj igre i rezultat

Rezultat se mjeri brojem uništenih neprijatelja (score). Ukoliko je igrač pogođen igra završava. Ukoliko je rezultat na kraju igre veći od prethodnog najvećeg rezultata, taj rezultat postaje najveći rezultat(HIGH SCORE). Tekst je zapisan uz pomoć Font objekta unutar render metode.

```
_font->start();
char fps[32];
sprintf(fps, "%d", getFrameRate());
_font->drawText(fps, 5, 5, Vector4(0, 0.5f, 1, 1), 20);
char scoreText[32];
sprintf(scoreText, "SCORE: %d", score);
_font->drawText(scoreText, 50, getHeight() - 60, Vector4(1, 1, 1, 1), 40);
if (gameEnd)
{
    char highScoreText[32];
    sprintf(highScoreText, "HIGH SCORE: %d", highScore);
    _font->drawText(highScoreText, getWidth() / 2 - 150, getHeight()/2 - 150, Vector4(1, 1, 1, 1), 40);
    _font->drawText(scoreText, getWidth() / 2 - 100, getHeight() / 2-100 , Vector4(1, 1, 1, 1), 40);
    char playAgain[32];
    sprintf(playAgain, "PRESS 'FIRE' TO PLAY AGAIN");
    _font->drawText(playAgain, getWidth() / 2 - 280, getHeight() / 2 - 50, Vector4(1, 1, 1, 1), 40);
}
_font->finish();
```

Slika 21. Postavljanje teksta



Slika 22. Kraj igre

12. Pokretanje igre

Za pokretanje igre `spaceinvaders.exe` aplikaciju potrebno je premjestiti iz Debug direktorija u glavni direktorij projekta. Glavni direktorij projekta nalazi se na putanji: `'\GamePlay-master\project\spaceinvaders'`

Druga opcija je pokretanje uz pomoć Visual Studio lokalnog debugger-a. Potrebno je odabrati željeni projekt i postaviti ga kao StartUp projekt te pokreniti GamePlay solution.

13. Osvrt na GamePlay alat

Gameplay je snažan alat za izradu igara. Temelji se na C++ jeziku što omogućava dobre performanse. Međutim, Gameplay ima mnogo problema. Alat nije dovoljno dobro dokumentiran. Službena dokumentacija je poprilično sažeta, dodiruje samo osnovne pojmove i ne opisuje sve funkcionalnosti koje alat pruža. Iz kratkog iskustva rada s GamePlay-om imao sam velikih poteškoća s pronalaženjem instrukcija za implementaciju i korištenje većine internih funkcija koje GamePlay pruža. Jedini cjeloviti izvor znanja je API koji je napisan stručnim jezikom i težak za čitanje i razumijevanje. Nadalje, API pruža samo izlistane funkcije i njihov opis, ali ne i preporuku kada i zašto koristiti neku funkciju te kako ostvariti neku funkcionalnost.

Google pretraga gotovi svih problema i pitanja koji su se postavili tijekom razvoja nije pronašla korisne rezultate u niti jednom slučaju. Većina rezultata bi bili rješenja za sličan ili identičan problem vezan za Unity. Iz prethodnog zaključujem da, ikao besplatan, GamePlay nije široko primijenjen za razvoj, te da pogotovo nije prikladan za upoznavanje početnika s računalnom grafikom. Uz zahtjevnu i sažetu dokumentaciju, pristup alatu je dodatno otežan radom u C++ jeziku.

Uz sve poteškoće pri izradi aplikacije, smatram da je GamePlay ipak jako dobar alat za izradu igara