



UNIVERZITET U NOVOM SADU  
FAKULTET TEHNIČKIH NAUKA



## Predmet : Multiprocesorski sistemi

Napisati paralelni softver koji implementira igricu Space Defender

# 1. Uvod

Zadatak ovog projekta je implementacija paralelnog softvera koji realizuje igru **Space Defender**. Igrač upravlja svemirskim brodom koji se nalazi na dnu ekrana i ima mogućnost kretanja levo i desno, kao i ispaljivanja metaka. Sa gornje ivice ekrana kontinuirano padaju asteroidi koje je potrebno pogoditi pre nego što stignu do dna.

Pogodak metka uništava asteroid i povećava rezultat za jedan poen, nakon čega se generiše novi asteroid na vrhu ekrana, na nasumičnoj horizontalnoj poziciji. U svakom trenutku je aktivno najmanje dva asteroida. Igra se završava pobedom kada rezultat dostigne ili premaši zadatu vrednost (*target*), odnosno porazom kada bilo koji asteroid dodirne donju ivicu ekrana.

Program je realizovan u programskom jeziku **C++**, uz korišćenje **OpenCV** biblioteke za grafički prikaz i obradu korisničkog ulaza. Paralelizacija je ostvarena korišćenjem **POSIX Threads (pthreads)**, pri čemu su pojedini delovi logike igre izvršavani u posebnim nitima.

## 2. Kontrole i parametri

### 2.1 Kontrole

- Kretanje levo/desno: **A** / **D** ili strelice
- Pucanje: **W** ili **SPACE**
- Restart runde: **R**
- Izlaz iz programa: **ESC** ili **Q**

### 2.2 Parametri komandne linije

- **--asteroids N** — broj aktivnih asteroida (minimum 2)
- **--target K** — ciljani broj poena za pobedu
- **--width W** — širina prozora u pikselima
- **--height H** — visina prozora u pikselima

Ukoliko se parametri ne navedu, koriste se podrazumevane vrednosti.

## 3. Glavne strukture podataka

U implementaciji su korišćene sledeće strukture podataka:

### Ship

{ x, y, w, step }

Predstavlja stanje broda, odnosno njegovu poziciju na ekranu i korak kretanja pri pomeranju levo ili desno.

### Bullet

{ x, y, v, active }

Predstavlja projektil. Brzina metka je izražena u pikselima po sekundi i kretanje je zasnovano na realnom vremenu (dt). Polje active označava da li je metak trenutno aktivan.

### Asteroid

{ x, y, r, v, alive }

Predstavlja asteroid sa pozicijom, radijusom i brzinom padanja izraženom u pikselima po sekundi. Polje alive označava da li je asteroid aktivan.

## InputState

```
{ move_impulse: atomic<int>, fire_impulse: atomic<bool> }
```

Struktura koja služi za prenos impulsa korisničkog ulaza između niti. Korišćenjem atomika omogućeno je bezbedno čitanje i pisanje bez dodatnog zaključavanja.

## GameState

Struktura koja sadrži kompletno deljeno stanje igre:

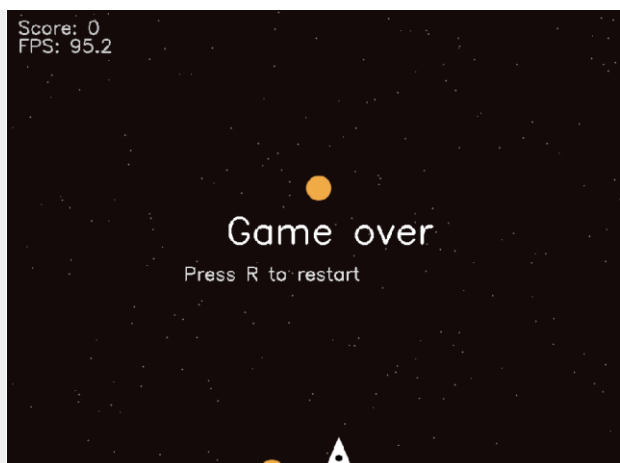
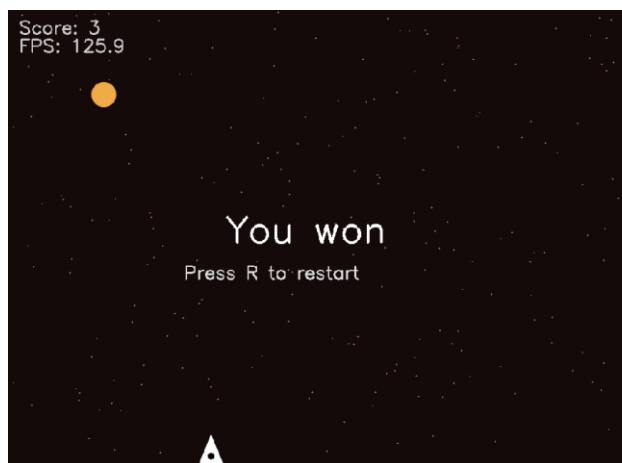
- dimenzije prozora (W, H),
- ciljani rezultat (*targetScore*),
- vektore asteroida i metaka,
- stanje broda,
- trenutni rezultat,
- indikatore završetka igre (*win*, *over*) i indikator rada programa (*running*),
- generator slučajnih brojeva,
- globalni mutex (`pthread_mutex_t`) za sinhronizaciju.

## 4. Funkcije

Implementacija koristi sledeće ključne funkcije:

- **hit(const Bullet&, const Asteroid&)**  
Proverava koliziju između metka i asteroida na osnovu udaljenosti njihovih centara.
- **draw\_ship(Mat&, int x, int y, int w)**  
Is crtava brod u obliku trougla na zadatoj poziciji.
- **init\_asteroids(GameState&)**  
Inicijalizuje vektor asteroida sa nasumičnim početnim pozicijama i brzinama.
- **reset\_round(GameState&)**  
Resetuje stanje runde, uključujući brod, asteroide, metke i rezultat.

Kretanje metaka i asteroida zasnovano je na realnom vremenu, pri čemu se pomeraj računa kao proizvod brzine objekta i proteklog vremena ( $dt$ ).



## 5. Tok programa

### 5.1 Inicijalizacija

Na početku programa vrši se parsiranje parametara komandne linije, inicijalizacija OpenCV prozora i inicijalizacija strukture GameState. Nakon toga se generiše početni skup asteroida.

## 5.2 Pokretanje niti

Pokreću se sledeće niti:

- jedna nit za upravljanje brodom (`ship_thread`),
- $N$  niti za asteroide, po jedna nit za svaki asteroid.

## 5.3 Glavna petlja (T0 / main)

U svakoj iteraciji glavne niti izvršava se:

- obrada tastature i upis impulsa u `InputState`,
- ažuriranje metaka i uklanjanje neaktivnih projektila,
- provera kolizija metak–asteroid uz kratko zaključavanje mutex-a,
- ažuriranje rezultata i provera uslova pobede ili poraza,
- iscrtavanje scene i prikaz HUD-a (rezultat, FPS i poruka o završetku igre).

## 5.4 Završetak i čišćenje

Pri izlazu iz igre postavlja se indikator `running = false`. Zatim se čeka završetak svih radnih niti pomoću `pthread_join`, zatvara se OpenCV prozor i oslobađaju se zauzeti resursi.

# 6. Arhitektura i niti

### T0 (main nit)

Zadužena je za obradu korisničkog ulaza, upravljanje mecima, proveru kolizija, logiku pobede i poraza, kao i za iscrtavanje igre.

### Nit za brod

Čita impulse iz `InputState`, pomera brod u okviru granica arene i generiše metke.

### N niti za asteroide

Svaka nit upravlja jednim asteroidom. Nit računa padanje asteroida korišćenjem dt modela, obavlja respawn nakon uništenja i detektuje dodir dna ekrana.

# 7. Deljeni podaci i sinhronizacija

## 7.1 Deljeni podaci

- vektori metaka i asteroida,
- stanje broda,
- rezultat i indikatori završetka igre.

## 7.2 Primitivi za sinhronizaciju

- **Mutex** (`pthread_mutex_t`)  
Koristi se za zaštitu deljenih struktura podataka tokom kolizija, respawn-a i ažuriranja rezultata.
- **Atomici**  
Koriste se za impulse korisničkog ulaza i za kontrolu životnog ciklusa programa.

### 7.3 Pravila pristupa

- Kolizije se obrađuju centralizovano u glavnoj niti radi determinističkog ponašanja.
- Niti za brod i asteroide ne vrše iscertavanje i ne menjaju HUD.
- Pristup deljenim podacima ograničen je na kratke kritične sekcije.