UNIVERZITET U NOVOM SADU

FAKULTET TEHNIČKIH NAUKA

Predmet : Multiprocesorski sistemi

Napisati paralelni softver koji implementira igricu Space Defender.

Asistent : Anja Tanović

Student : Aleksandar Vig

1. Uvod

Potrebno je napraviti paralelni softver koji implementira igru **Space Defender**. Igrac upravlja brodom na dnu ekrana (A/D ili strelice) i puca metke (W/SPACE). Sa vrha padaju asteroidi; pogodak unistava asteroid i donosi 1 poen. Istovremeno pada najmanje 2 asteroida; nakon unistenja jednog, generise se novi na vrhu na nasumicnoj poziciji. Igra se zavrsava pobedom kada score >= target, ili porazom kada asteroid dodirne dno. Koristi se **OpenCV** za prikaz; paralelizacija je uradjena nitima (std::thread/pthreads).

---

2. Kontrole i parametri

2.1 Kontrole

- **Levo/Desno:** A / D ili strelice
- **Pucanj:** W ili SPACE
- **Restart:** R
- **Izlaz:** ESC ili Q

2.2 Parametri komandne linije

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

---

3. Glavne strukture podataka

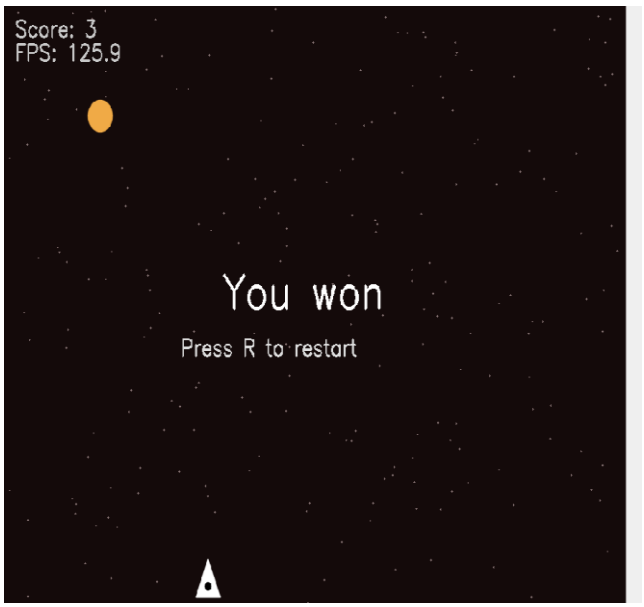
- Ship { x, y, w, step }
- Bullet { x, y, v, active }
- Asteroid { x, y, r, v, alive }
- InputState { move_impulse: atomic<int>, fire_impulse: atomic<bool> }
- GameState { W, H, targetScore, asteroidCount, running: atomic<bool>, game_active: atomic<bool>, win: bool, over: bool, score: int, ship, ast: vector<Asteroid>, bullets: vector<Bullet>, rng/distribucije, mtx: pthread_mutex_t }

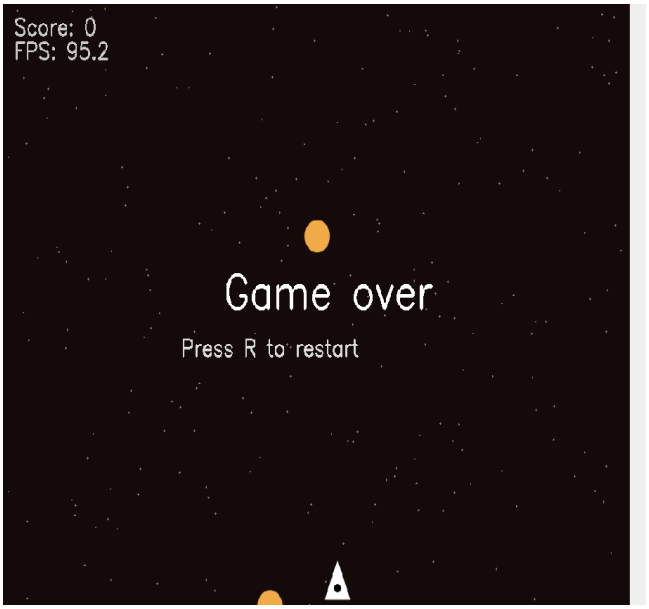
---

4. Funkcije

- **void spawnAsteroid(Asteroid& a, int W)** — postavlja a.x = rand()%W, a.y = 0, a.alive = true.
- **void updateAsteroid(Asteroid& a, int H)** — pomera asteroid nadole; vraca indikator dodira dna.

- **void updateShip(Ship& s, InputState keys, int W)** — pomera brod levo/desno u okviru arene.
- **void fireBullet(vector<Bullet>& bullets, const Ship& s)** — aktivira neaktivni metak na polozaju broda.
- **void updateBullets(vector<Bullet>& bullets, int H)** — pomera metke nagore; gasi metke izvan ekrana.
- **bool hit(const Bullet& b, const Asteroid& a)** — kolizija (distanca <= r + poluprecnik metka).
- **int resolveCollisions(vector<Bullet>& bullets, vector<Asteroid>& asteroids)** — gasi pogodjene metke/asteroide, povecava score.
- **void renderFrame(Mat& frame, const Ship&, const vector<Bullet>&, const vector<Asteroid>&, int score)** — iscrta stanje i skor (HUD).
- **bool checkWin(int score, int target)** — pobeda kada score >= target.
- **void showEndScreen(bool win)** — prikazuje "You won" ili "Game over".





5. Tok programa (main petlja)

1. **Inicijalizacija**
   - Otvaranje OpenCV prozora i postavljanje dimenzija (--width, --height).
   - Parsiranje parametara: --asteroids, --target.
   - Inicijalizacija stanja (brod, vektori metaka/asteroida, skor).
2. **Start niti**
   - Pokretanje jedne niti za brod (ship_worker).
   - Pokretanje N niti za asteroide (asteroid_worker, po jedna po asteroidu; minimum 2).
3. **Glavna petlja (T0 / main) — svaka iteracija:**
   - **Tastatura**: citanje tastera (A/D ili strelice; W/SPACE; R; ESC/Q) i upis impulsa u InputState.
   - **Metci**: pomeranje aktivnih metaka nagore; gasenje metaka izvan ekrana.
   - **Kolizije**: kratko zakljucavanje mtx, provera pogodaka metak–asteroid, azuriranje score, respawn unistenih asteroida.
   - **Provera stanja**: pobeda ako score >= target; poraz ako je bilo koji asteroid dotakao dno.
   - **Iscrtavanje**: kratko procitati stanje (snapshot) i nacrtati scenu + HUD; u zavrsetku prikaz "You won" / "Game over".
4. **Zavrsetak**
   - Na ESC/Q, pobedu ili poraz: postaviti running = false.
   - Ispratiti kraj runde (poruka), zatvoriti prozor.
5. **Ciscenje**
   - join svih radnih niti (brod + asteroidi).
   - Oslobadjanje resursa i uredno gasenje.

---

6. Arhitektura i niti

- **T0 (main) —** glavna kontrola: tastatura → InputState, kretanje metaka, kolizije, skor, win/lose, iscrtavanje frejma i zivotni ciklus**.**
- **Nit za brod (ship_worker) —** cita impulse iz InputState, pomera brod unutar arene, generise metke (aktivira slobodan metak na impuls pucnja).
- **N niti za asteroide (asteroid_worker) —** svaka nit upravlja jednim asteroidom: pad (y += v), respawn na vrhu kada je unisten; pri dodiru dna postavlja globalni indikator kraja.

---

7. Deljeni podaci i sinhronizacija

7.1 Deljeni podaci

- **std::vector<Bullet> bullets, std::vector<Asteroid> asteroids, Ship ship**
- **score (int), dimenzije W/H, targetScore, asteroidCount**
- **globalne zastavice: running (atomic<bool>), game_active (atomic<bool>), win/over (bool)**

7.2 Primitivi

- **Globalni mutex: pthread_mutex_t mtx —** stiti pristup deljenim vektorima i skoru tokom kolizija/respawn-a.
- **Atomici: running, game_active i impulsi u InputState — bez locka.**
- **Tajming:** niti koriste kratke uspavane intervale (~8–16 ms) za meko uskladjivanje;

nema barijere po frejmu.

7.3 Pravila pristupa

- Kolizije su centralizovane u main niti (deterministicki ishod, kratak kriticni deo).
- U render fazi koristi se snapshot: nakratko zakljucati mtx, kopirati stanje u lokalne strukture, odmah otkljucati, pa crtati iz kopije.
- Niti asteroida i broda ne crtaju i ne menjaju HUD.