**Elektrotehniško-računalniška strokovna šola in gimnazija Ljubljana**

**Vegova ulica 4, 1000 Ljubljana**

**»Čas za obračun«**

**SPECIFIKACIJE**

**za 4. predmet na poklicni maturi**

Moravče, 6. 1. 2025 Avtor: Klemen Skok, R4.A

**Kazalo**

[1. Ideja igre 3](#_Toc187053986)

[2. Princip odjemalec-strežnik 3](#_Toc187053987)

[3. Stran strežnika 3](#_Toc187053988)

[4. Stran odjemalca 4](#_Toc187053989)

[5. Zastavice 4](#_Toc187053990)

[6. Orodja in knjižnice 5](#_Toc187053991)

[7. Viri 5](#_Toc187053992)

**Kazalo slik**

[Slika 1:Shema strežnika 3](#_Toc187052572)

# Ideja igre

Igra poteka po principu “ujemi zastavo” (ang. “*Capture the Flag*”). Dogajanje se odvija na virtualnem hodniku Vegove v prvem nadstropju. Ob zvonenju šolskega zvonca se na hodniku zbereta ekipi: dva profesorja oz. dežurna dijaka na poti po malico na sredini hodnika. Da bo naloga težja, lahko igralec ostale igralce obmetava z različnimi predmeti (npr. svinčniki, krede) in jih tako upočasni. Tisti par, ki največkrat uspe malico odnesti na svojo stran, zmaga.

# Princip odjemalec-strežnik

Igra temelji arhitekturi odjemalec-strežnik (ang. *client-server*), ki prenaša podatke prek omrežja na podlagi protokola UDP, ki omogoča minimalen zamik pri komunikaciji. Strežnik deluje tako, da sprejema UDP sporočila od odjemalcev in glede na vsebino paketov posodobi stanje seje igre. Hkrati tudi vsem odjemalcem v igri periodično pošilja trenutno stanje igre.

# Stran strežnika

A diagram of a server

Description automatically generatedZgradba strežnika je realizirana po sledeči strukturi:

Slika 1:Shema strežnika

Program poteka v treh ločenih nitih. Prva nit preverja, ali je program dobil kakšno novo sporočilo. Iz prejetih sporočil izloči podatke in pošiljatelja, ju zapakira v strukturo UDPmessage, ki jo doda na konec čakalne vrste za obdelavo, ki je realizirana z std::queue (recievedQueue). Iz slednje jih nato strežnik prebira v glavni niti, in z njimi upravlja s povezavo, dodaja oz. odstranjuje odjemalce, ali pa posodablja podatke o igri. podatke vsake igre periodično pošilja v še eno čakalno vrsto (prav tako std:queue, sendQueue), katere podatke v še eni stranski niti pošilja odjemalcem.

Sama vsebina paketa je s strani knjižnice podana kot tabela bajtov (Uint8\*), vendar jo strežnik zapakira v objekt tipa razreda PacketData, ki podatke shrani kot std::vector<Uint8>, kar omogoča preprost dostop do posameznih bajtov in do zastavic, ki so namenjene za strežnik.

Pri tej arhitekturi je ključnega pomena zagotavljanje varnosti podatkov, s katerimi lahko upravlja več niti (ang. *thread safety*). Slednja je zagotovljena s pomočjo razreda std::mutex, ki z zaklepanjem omogoča uporabo določenega podatka le eni niti hkrati. Dodatna varnost je zagotovljena s podatkovnim tipom std::atomic<T>, ki zagotavlja varno branje podatka, tudi če ga druga nit ravno spreminja.

Strežnik bo beležil tudi log dnevnike od delovanju, kamor bo zapisoval pomembnejše dogodke, opozorila in napake med izvajanjem.

# Stran odjemalca

Arhitektura odjemalcev je zelo podobna arhitekture strežnika, le da se v tem primeru glavni razred ne imenuje *Server*, ampak *Game*. Glavna nit pri odjemalcu bere igralčeve vnose in jih (oz. njegovo pozicijo) pošlje strežniku. Hkrati stanje posodobi tudi pri sebi, da se zamik pri komunikaciji čim manj občuti.

Za preprosto komunikacijo bo odjemalec lahko prejel le nekaj različnih formatov podatkov, kot so položaji igralcev (vključno z njim, da se prepreči odstopanja, ki bi nastala pri izgubi paketov), položaji ostalih predmetov, stanje in časovnik igre, rezultat in podobno. Oblike formatov bodo ločene s pomočjo številčenja le-teh.

Uporabniški vmesnik bo prikazoval tlorisni pogled igre, mini zemljevid v kotu, ikone povezanih igralcev, ipd.

# Zastavice

Prvi bajt v podatkovnem delu sporočila bodo vedno predstavljale zastavice, namenjene strežniku.

|  |  |  |
| --- | --- | --- |
| **Ime** | **Vrednost** | **Pomen** |
| ACK | 0x01 | Potrdilo (1 = da) |
| FIN | 0x02 | Zahteva za prekinitev povezave |
| KEEPALIVE | 0x04 | Ohranitev povezave, ko se nič ne dogaja (npr. čakanje na sejo) |
| PUSH | 0x08 | Zahteva podatkov s strani strežnika |
| PULL | 0x10 | Zahteva podatkov s strani odjemalca |
| DATA | 0x20 | Pove, da paket vsebuje podatke |
| - | 0x40 | - |
| - | 0x80 | - |

# Orodja in knjižnice

**Orodja:**

* *Visual Studio Code* – urejevalnik kode
* *CMake –* izvajanje kode in razhroščevanje (odjemalci)
* *Makefile –* izvajanje kode in razhroščevanje (strežnik)
* *GitHub –* shranjevanje projekta, kontrola različic

**Knjižnice:**

* SDL2 – prikaz uporabniškega vmesnika v oknu
* SDL2\_net – komunikacija med strežnikom in odjemalci
* SDL2\_image – prikaz slik v oknu
* spdlog – pisanje log dnevnikov

# Viri

Med razvijanjem in pripravo dokumentacije so bili uporabljeni naslednji viri:

* SDL2 dokumentacija (Citirano 6. 1. 2025). Naslov: <https://wiki.libsdl.org/SDL2/CategoryAPI>
* SDL2\_net dokumentacija (Citirano 6. 1. 2025). Naslov: <https://wiki.libsdl.org/SDL2_net/CategoryAPI>
* C++ dokumentacija *cppreferenc*e*.com* (Citirano 6. 1. 2025). Naslov: <https://en.cppreference.com/w/>
* C++ dokumentacija *cplusplus.com* (Citirano 6. 1. 2025). Naslov: <https://cplusplus.com/reference/>
* spdlog dokumentacija (Citirano 6. 1. 2025). Naslov: <https://github.com/gabime/spdlog?tab=readme-ov-file#spdlog>