Proekti Dokumentatsioon

# Intro

Proekt on antud meile kolmele, sellepärast esiteks peame jagada projekti:

* Moduliks;
* Etappideks;
* Töömaht tegelaste vahel;

Proekt on koostöö (andmevahetus) Klientide ja Serveri vahel.

## Mudel

Mudel on objekt, mis hojab Kliendi seisundit. Selle projekti jaoks pakkun Mudelit, mis on kirjeldatud Andmemudeli Näidisvorm Punktis.

# Serveri Funktsionaalsus

Server

* Võtab Klientide ühendust; Töötleb Klientide ühenduse katkestamisi;
* **Hoiab Andmemudeli etaloon;**
* Saab Andmed Klientide käest; Töötleb saadud Andmeid; Muudab andmemudeli;
* Saadab muudatud Andmemudeli Klientidele;

Andmemudeli etalooni Serveris hoidmine on eelistav, sest et:

1. Kõik töötlemise loogika on ühes kohas – Serveris; Isegi kui Klient edastab Andmeid ebakorrektselt, ei saa ta rikustada mängu reegleid.
2. Kui mõlemad Kliendid saavad täieliku Andmemudelit(mitte ainult muudatusi), siis Kliendides kui ka Serveris on garanteeritud sünkroniseeritud mängu „pilt“.

Serveri Andmemudel on andmed Player’i seisundi kohta. Sellepärast „asuvad“ Serveris 2 Player objekti.

# Kliendi Funktsionaalsus

Klient

* Tekitab ühendusi Serveriga; Registreerib enda serveris – genereerib endale ID;
* Saab Serverilt jooksvat Andmemudelit;
* Väljastab jooksva Andmemudeli ekraanile; \*\*
* Saab Kasutajalt käsusid ja edastab neid Serveri poole;

Kliendis asuvad ka 2 Player objekti – Klient ise ja tema „Opponent“. Klient ei tee midagi selle objektidega, vaib loeb neid ja väljastab vajaliku infot ekraanile.

\*\*Alguseks tegeme tekstiväljundi. Siis saame kontrollida et Proekt on üldiselt runnable. Tekstiväljundi on ka lihtsam kodeerida ja kontrollida. Kui aega leiame, siis saame ka püüata teha graafilisi moodulit.

# Andmete vahetuse kord

## Reglament

Server

* Pidevalt(periodiga Ns\*\*) saadab Andmemudelit mõlemadele Kliendidele sõltumatu sellest, kas oli Andmemudel muudatud või ei oli.

\*\* Perioodi saame hiljem seadistama nt eraldi failis. Loogiline periood minu arvates on : 0,1~30sek. Perioodiga 0,1sek saame „Real-Time“ mängu, perioodiga 30sek saame „Turn-Based“ variandi. Seda saab Host hiljem ise muudata.

Klient

1. Väljastab mängu „maailma“ ekraanile;
2. Võtab Kasutajalt Käsusid (alguseks käsud tulevad teksti vormis);
3. Edastab Käsud Serveri poole;
4. Ootab vastust serverilt;
5. Kordab punktid 1-4;

## Andmemudeli näidisvorm

Client ------------------> Server:

* Player.ID+Player.Name+ActionCode

\*\* Player.ID on RGV (Random Generated Value), Player.Name’i Kasutaja võib sisestada. Seda väljastatakse mõlemade Kasutajade ekraanidele.

Server <------------------ Client:

* „[Player.ID; Player.HP; Player.LastAction; Player.LastReceivedAction; [Player.CooldownTimer1;....; Player.CooldownTimerN]; comments();]“

\*\* Seda vormi saadame täistekstina. Iga Value on eraldatud semikoloniga ( „; „ - tahtsin seda kirjutada, kuid on raske näha). CooldownTimer’id on eraldi massiivina, sest me ei tea veel, mitu neid tuleb. Comments() on just tekst, kuhu me paneme effekti infod tekstina, sest Klient ei töötle seda infot ise, just väljastab ekraanile Comments()’i näide: „Player took 5 damage. Cooldowns increased by 10%“. Sellega on väga tähtis ka vaadata, et Player.Name, LastAction, LastReceivedAction kui ka comments() EI sisalda semikolonit, sest see murrab massiivi struktuuri.