Kapitola 1

Uživatelská dokumentace

Po zkompilování jsou vytvořeny dvě aplikace - `Client.exe` a `Server.exe`. Nejdříve je potřeba spustit server, poté lze pustit klienta. Po jeho spuštění se zobrazí černé okno a konzole kam uživatel zadá IP adresu serveru(”127.0.0.1.”pokud běží server na stejném PC). Poté by měl být připojen do hry.

Hráč ovládá svůj tank pomocí kláves WASD a střílí levým tlačítkem myši. Pod každým tankem je vidět aktulní množství štítů(modře) a životů(červeně). Po zásahu se nejdříve ubírají štíty, poté životy a při 0 životech hráč umírá, objeví se na novém místě a započítá se mu smrt do tabulky. Ta je k dispozici při držení klávesy TAB. Pokud nejsou štíty úplně vyčerpané, tak se postupně obnovují. Na mapě jsou na zelených blocích dispozici štíty - modré ”O”které obnoví i vyčerpaný štít. Černé bloky značí neprůchozí stěny a šedé bloky místa, kde se oživí mrtví hráči.

Kapitola 2

Programátorská dokumentace

Projekt se skládá z celkem 4 částí:

1. Server aplikace
2. Client aplikace
3. Shared knihovna
4. Engine knihovna

2.1 Engine

Engine je knihovna, která v sobě obsahuje logiku samotné hry. Přehled menších tříd:

- `World` sdružuje všechny objekty ve hře do herního světa.
- `Arena` reprezentuje arénu pro hráče jako 2D tabulku.
- `Player` připojený hráč.
- `EngineCommand` hiearchie tříd, které slouží jako příkazy pro ovládání hry.

Hlavní třídou této knihovny je `Engine`. Jejím účelem je vytvořit API pro ovládání herního světa - `World`. `Engine` obsahuje herní fyziku, především řešení kolizí a jejich následků. Větší část logiky je implementována pomocí hiearchie tříd odvozených od `EngineCommand`. Tyto příkazy se chovají jako funkce, které engine může volat. Tento přístup byl zvolen z toho důvodu, že tímto jsou k dispozici všechny "akce", které mění svět, obsahují v sobě potřebná data ke změn2 a dají se proto jednoduše poslat přes sít a držet klienta a server v synchronizovaném stavu. Například provedení příkazu `PlayerFireCmd` provede vystřelení střely od hráče, tento příkaz je generován po kliknutí myší. Ovšem všechny příkazy jsou "spekulativní", což znamená, že pokud je nelze provést, tak nehlásí žádnou chybu. To je z důvodu, že po síti nemusí přijít ve správném pořadí a mohlo by se stát, že přijde příkaz na změnu pozice již odpojeného hráče, v tom případě se příkaz tiše ignoruje.

2.2 Client

Je aplikace určená pro koncového uživatele, která využívá `Engine` a komunikaci se serverem k vytvoření hry, o grafickou část se stará knihovna OpenTK využívající OpenGL.

`Window` je třída, která vytváří grafické okno, obsahuje herní smyčku a `Main()` funkci. V této smyčce běží instance třídy `Game`. Ta tvoří hru, která je implementována jako stavový automat pro jednotlivé stavy hry. Momentálně existují `MenuState` a `PlayingState`. První slouží k připojení k serveru, ve druhé třídě pak program tráví většinu času, protože obsahuje samotnou hru.

Dále je zde skupina menších tříd:

- `Input` dává ostatním částem hry k dispozici vstupy uživatele.
- `FontManager` načítá grafický font a umí kreslit text.
- `ShaderProgram` objektové API pro práci s OpenGL shadery.
- `ObjModel` je třída schopná načítat a kreslit .obj modely.
- `[XXX]Renderer` třídy vykreslují jednotlivé části herního světa sdružené do třídy `Renderer`.

Zmíněná `PlayingState` obsahuje samotný engine, implemetuje obsah hrací smyčky a pomocí `Renderer` kreslí veškerou grafiku. Důležitým prvkem je `ServerManager`, což je třída poskytující API ke komunikaci se serverem. Tato třída bude popsána později.

2.3 Server

Server je aplikace, na které skutečně běží hra. Obsahuje pouze dvě třídy a to `Program` obsahující `Main()` a `ClientsManager`, což je druhá polovina komunikace mezi klientem a serverem. Konkrétně se stará o komunikaci s klienty a podrobněji je popsána dále.

2.4 Shared

Je podpůrná knihovna, která obsahuje sdílený kód používaný klientem i serverem ke komunikaci mezi sebou po síti. Přehled tříd:

- `Serialization` statická třída obsahující metody pro (de)serializaci základních datových typů.
- `Communication` statická třída poskytuje metody pro asynchronní komunikace přes TCP a UDP.
- `ConnectingStaticData` obsahuje data pro připojení klienta k serveru, konkrétně taková data, která se během hry nemění - momentálně mapa `Arena`.
- `ConnectingDynamicData` druhá část data nutná pro připojení. Naopak obsahuje data, která se mění - aktualní seznam a pozice hráčů, dynamické prvky na mapě (`ShieldPickup`).
- `ServerCommand` hiearchie tříd odpovídající třídám `EngineCommand`, které slouží k zabalení a poslání příkazů přes síť od klienta k serveru.

- `ClientUpdate` je třída, kterou klient pravidelně posílá serveru a obsahuje vstupy klienta.
- `ServerUpdate` třída, která je schopná spolehlivě posílat zprávy(včetně `ServerCommand`) od serveru ke klientovi.

Všechny třídy se umí (de)serializovat. Jejich použití je uvedeno v následujícím popisu komunikace.

2.5 Komunikace serveru a klienta

O veškerou komunikaci se starají již zmíněné 2 třídy - `ServerManager,ClientsManager`. První je součástí Clienta a běží v `PlayingState`. Server komunikuje s klienty prostřednictvím `ClientsManager`. Obě třídy obsahují asynchronní metody pro plynulou komunikaci.

Na serveru běží paralelně tři úlohy - hlavní smyčka, čekání na příchozí spojení a čekání na příchozí `ClientUpdate` od klientů. Hlavní smyčka:

1. Získání nově přijatých `ClientUpdate`, jejich zpracování na `EngineCommand` příkazy a následné vykonání.
2. Zpracování připravených klientů. Což znamená vytvoření a rozeslání dynamických dat. Což je synchronně provedeno zde, protože se stav hry nemění. Z toho důvodu ale musí klienti už nyní poslouchat pro nové příkazy. Neboť jim mohou přijít dříve než dynamická data, která v tu chvíli budou mírně zastaralá.
3. Odstranění nereagujících klientů. Pokud klient delší dobu nepošle žádnou zprávu, tak bude odpojen.
4. Vytvoření všech příkazů o novém stavu hry - nově připojení hráči, střelba, smrt hráčů a jejich nové pozice. Příkazy jsou poté všem klientům rozeslány. Navíc, každý `ServerCommand` mánu sebe má vlastnost `guaranteedExec`, která značí zda je nutné aby klient příkaz vykonal. Přesnější zda se příkaz pošle spolehlivě nebo stačí nespolehlivě. Například nové pozice hráčů se posílají nespolehlivě přes UDP, protože jsou absolutní a jsou posílány opakovaně. Takže při ztracení nedojde k desynchronizaci, pouze ke zpozdění. Naopak odpojení hráču se posílá pouze jednou a musí být tedy doručeno, jinak by došlo k tomu, že klient uvidí nehybné, odpojené hráče. Momentálně se spolehlivě posílají informace o připojení, odpojení, smrti a sebrání štítů. Nespolehlivě se posílají pozice a střelba.

Postup napojení klienta na server:

1. Server pošle klientovi statická data ve formě `ConnectingStaticData`.
2. Po obdržení začne klient asynchronně poslouchat na příkazy od serveru.
3. Toto potvrdí posláním `ACK` paketu.
4. Server označí klienta jako připraveného.
5. Při zpracování připravených klientů server pošle dynamická data ve formě `ConnectingDynamicData`. a zároveň zařadí klienta mezi aktivní hráče.

Smyčka klienta:

1. Zpracování vstupů od uživatele do `ClientUpdate` a poslání serveru.
2. Predikce stavu hry na základě zpracovaných vstupů. Výhoda predikce je, že hra reaguje ihned na akce uživatele, dříve než je stihne potvrdit a poslat server. Ale je potřeba dělat korekci a ukládat odeslané `ClientUpdate`.
3. Získání přijatých příkazů od serveru, jejich vykonání a tím korekce na správný stav hry. To je implemetováno tak, že každý `ClientUpdate` má své číslo, součástí příkazů od serveru je poté poslední zpracovaný `ClientUpdate`. Klient tedy podle příkazů upraví svůj stav hry a případně ještě aplikuje nepotvrzené vstupy od uživatele.

Následující diagram shrnuje běh klienta, serveru a vzájemnou komunikaci