Зборник радова Факултета техничких наука, Нови Сад

UDK: 621.38
DOI: https://doi.org/10.24867/22BE12Komazec

РАЗВИЈАЊЕ МУЛТИПЛАТФОРМСКЕ ВИДЕО ИГРЕ СА ПОДРШКОМ ЗА ИГРАЊЕ ПРЕКО МРЕЖЕ

DEVELOPING MULTIPLATFORM VIDEO GAME THAT CONTAINS SUPPORT TO PLAY ONLINE

Александар Комазец, Предраг Теодоровић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом чланку је описана реализација и функционисање мултиплатформске игре са подршком за играње преко мреже, као и независне серверске апликације. Омогућено је покретање апликације на оперативним системима Windows 10 OS, Linux OS, као и Android OS.*

Кључне речи: *Libgdx, Node, Box2D, MongoDB, Mongoose*

Abstract – *This article describes realization and functioning of multiplatform video game that contains support to play online. Here is provided to run the application on operating systems like Windows 10 OS, Linux OS, and Android OS.*

Keywords: *Libgdx, Node, Box2D, MongoDB, Mongoose*

1. УВОД

У овом раду ће бити представљена структура онлајн платформерске видео игре као и сам развој видео игре и сви коришћени алати.

Игра се састоји од две независне компоненте (Клијент и сервер). Клијент је развијан коришћењем библиотека које припадају радном оквиру Libgdx [1], док је серверска апликација развијана помоћу Node runetime окружења.

2. АЛАТИ, ЕКСТЕРНЕ БИБЛИОТЕКЕ, РАДНИ ОКВИРИ

Као што је већ поменуто у уводу, игра се састоји од клијента, сервера и базе података, тако да ће сви коришћени алати и библиотеке бити груписани у те три групе.

2.1 Клијент

2.1.1 “Libgdx” радни оквир

“Libgdx” радни оквир поседује подршку за развијање видео игара на различитим платформама и системима попут Windows, Mac, Linux, Android, iOS, and HTML5.

2.1.2 “Box2D” енџин

Box2D је енџин (енг. *engine*) за симулирање физике у дводимензионалном простору. Box2D енџин програмерима пружа апликативни програмерски интефејс помоћу кога је могуће креирати свет са реалистичном физиком.

2.1.3 “Ashley” библипотека

Ashley је библиотека заснована на ECS [2] (eng. *Entity Component system*) архитектуралном шаблону који је се обично креира искључиво при развоју видео игара. ECS софтверски шаблон се састоји из компоненти (То су модули који представљају контејнере за податке), ентитета који представља скуп компоненти, као и система (Систем представља модул који је задужен за једну фунционалност, на пример рендеровање слике или детекцију контакта између тела)

2.2 Сервер

Сервер развијан за потребе видео игре писан је у програмском језику JavaScript.

2.2.1 “Node.js”

Node.js је асинхроно run-time окружење које се заснива на V8 [3] енџину који је имплементиран у претраживач “Google Chrome” и служи за превођење JavaScript кода у машински код, тако да је могуће поред веб апликација развијати и системске апликације користећи главни JavaScript фајл из *Node* пројекта. У овом случају, *Node* је кориштен за развијање серверске апликације.

2.2.2 “Express.js”

Express.js[4] представља радни оквир базиран на већ постојећем Node.js. Express.js проширује могућности постојећег веб сервер модула који је састави део Ноде-а.

2.2.3 “Socket.io-server”

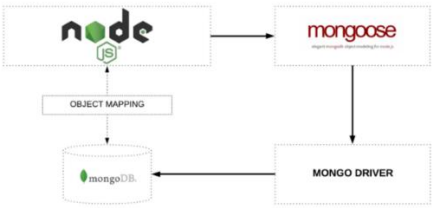
Socket.io [5] представља библиотеку за бидирекциону комуникацију између клијента и сервера засновану на догађајима користећи WebSockets протокол.

2.2.4 “Mongoose” i “MongoDB”

MongoDB [6] је NoSQL база података докумената без шеме. Подаци који се чувају у MongoDB бази података су представљени у JSON формату, а структура ових докумената може варирати на начин да се један објекат у бази састоји од свих понуђених поља док други објекат не садржи сва понуђена поља, већ искључиво обавезна поља.

Ово је једна од предности коришћења NoSQL-а јер убрзава развој апликација и смањује сложеност имплементације.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Предраг Теодоровић, доцент.



Слика 1. Мапирање објеката између Node-а и MongoDB помоћу Mongoose-а (преузето из [7])

Mongoose [7] је библиотека за моделовање података објеката (ODM) за MongoDB и Node.js. Управља односима између података, пружа проверу валидности шеме и користи се за превођење између објеката у коду и представљање тих објеката у MongoDB.

3. РАЗВОЈ 2D ОНЛАЈН ПЛАТФОРМЕР ВИДЕО ИГРЕ

Назив видео игре на којој се заснива овај рад је “Brick it”. Назив игре је настао тако што се комбинују енглеска речи “Brick” што значи цигла и енглеска реч “Break” што значи поломити.
Цео ниво је израђен од квадратних циглица различитих облика које могу бити уништене од стране играча. Играч има мотивацију да уништава циглице како би пронашао различите награде попут блага или различитих супер моћи.
Игра има могућност самосталног играња, као и могућност играња преко интернета са другим играчима у кооперативном моду са још једним играчем или у “Player vs Player” моду, где су играчи противници.
Игра такође подржава више платформи, то јесте може се играти на персо-налном рачунару или Андроид телефону.

Игра се састоји из три велика система:

•Административни систем који обувхата обраду података који су неопходни за почетак меча (Све класе неопходне за кориснички интерфејс, делегатори екрана и тако даље);

•Играчки (енг. *Gameplay*) систем који обувхата обраду свих података кад меч почне;

•Мрежни систем који обухвата компоненте одговорне за слање и примање података преко мреже.

3.1 Административни систем

Административни систем обухвата све компоненте од кад играч покрене апликацију па до момента када почиње меч.
Административни систем се састоји од:

•Специјализованих платформских покретача (енг. *Android Launcher* и *Desktop Launcher*);

•Game класе (MyGdxGame);

•Група корисничких екрана (eng. *Screens*);

•Менаџера ресурса (eng. *Asset Management Sys-tem*);

•Систем за испис порука (eng. *Logger*);

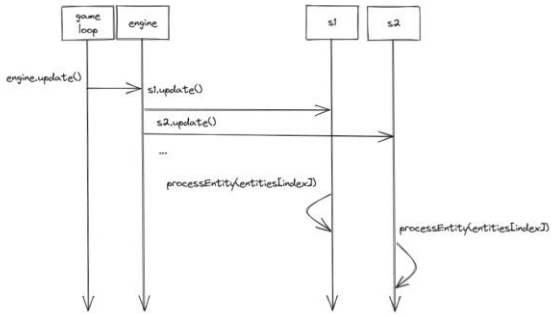
•Конфигурационе класе (GameConfig).

3.2 Играчки (енг. *Gameplay*) систем

Играчки систем сачињавају две велике групе:

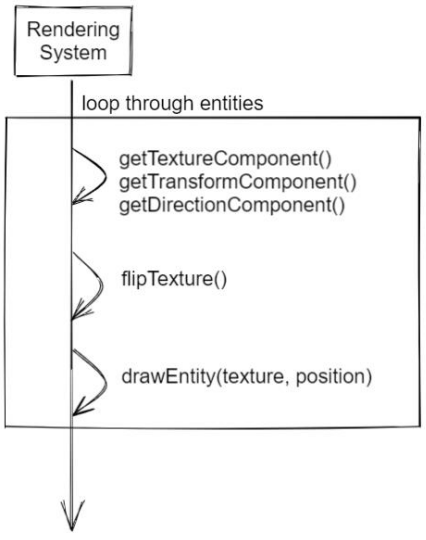
•Модули који се налазе у склопу ECS-а, попут ECS компоненти које представљају контејне-ре за податке као и ECS системи који предста-вљају специјализоване фунцкионалности где су сви системи смештени у један ред и синхроно се извршавају (слика 2.) приликом сваке итерације главне играчке петље;

•Сви остали модули који нису део ECS групе попут MatchTracker класе, BodyCreator, TileMapHandler, InputController класа и осталих. То су фунционални модули који пружају одређени сервис по позиву или асинхроно.



Слика 2. Итерација ентин-а кроз системе

На слици 3 налази се пример једног ECS система под називом “Rendering” систем. Рендеринг систем оперише само са ентитети-ма које је потребно приказати, то јесте који имају текстуру као што је играч или противник.
Рендеринг систем захтева и позицију где се ентитет налази на мапи, а ту информацију добија из Transform компоненте.
Direction компонента садржи информацију о смеру ентитета онда када је потребно заокренути текстуру. Метода за исцртавање прима текстуру и позицију на којој треба да је исцрта. Процедура се понавља за сваки релевантан ентитет.



Слика 3. Фунцкионалност Рендринг система

3.3 Мрежни систем

Мрежни систем видео игре је задужен за слање и примање података преко мреже онда када играч учествује у онлине мечу. ClientHandler је модул који је задужен за слање и примање порука.

3.3.1 Пример комуникације клијент - сервер (Ажурирање позиције играча)

У случају онлајн меча где учествују два играча, сваки клијент у свом свету види себе као локалног играча и преостале играче (У овом случају једног играча) као онлајн играча. Сваки играч је дужан да након промене позиције на мапи пошаље своју позицију серверу и да сервер проследи позицију тог играча преосталим играчима. У наставку ће бити описан процес ажурирања позиције у коме учествују два играча са јединственим ИД-јевима (0 и 1). Претпоставимо да играч 0 промени позицију (слика 4.).

1. Затим DataTransmittingSystem чита податке из бафера за слање података и прави пакет за слање тако што поред података о позицији пакује и *socket* ИД као и назив догађаја.

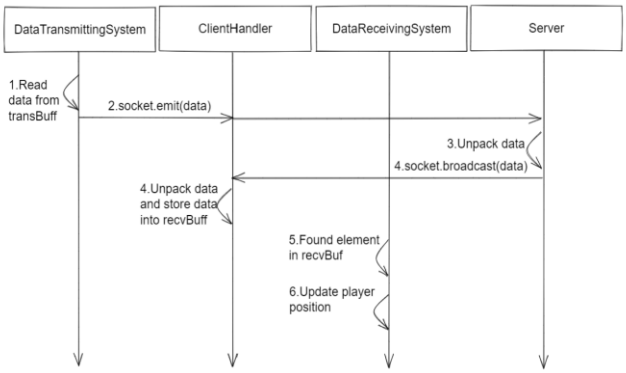
2. Пакет података се емитује користећи *socket*-е (утичнице) из ClientHandler-а;

3. Сервер региструје нови догађај, распакује податке, пакује их и прослеђује податке свим осталим клијентима (Играчу 1) користећи догађај под именом “updatePlayerInputPositionResp”;

4. ClientHandler играча 1 прихвати пристигли догађај, распакује податке, пакује их и смешта у бафер за примљене поруке;

5. DataReceivingSystem играча 1 проверава да ли има садржаја у баферу за примљене поруке;

6. Чита елемент из бафера и ажурира позицију на којој се налази играч 0.



Слика 4. Ажурирање позиције играча

3.4 Серверски систем

Серверска компонента је развијена коришћењем Node runtime окружења. Као веб сервер је кориштен Http сервер користећи Express.js радно окружење. За бидирекциону комуникацију се користе утичнице из библиотеке socket.io. База података која је кориштена је MongoDB. Mongoose библиотека се користи како би се једноставно дефинисале колекције података као и једноставније коришћење упита.

Такође треба имати на уму да је *Node* предвиђен за споре операције попут IO/мрежних операција као на пример слање и примање пакета као читање и писање са екстерног диска и да није намењен за захтевне процесорске операције тако да серверска апликација углавном служи са прослеђивање порука између клијената.

3.4.1 Компоненте серверског система

Сервер се састоји из две компоненте:

•Главни модул која представља улазну тачку за серверску апликацију. Има улогу да се при иницијализацији конектује са MongoDB сервером, иницијализује се http сервер (подеси се ip адреса и порт на којем сервер треба да слуша), и врши иницијализацију TrafficHandler модула. Одговоран је за регистрацију и пријаву клијената као и за слање упита бази података;

•TrafficHandler модул је задужен за комуникацију са клијентима као и за праћење стања соба и играча током меча.

4. ЗАКЉУЧАК

На основу развоја видео игре у овом пројекту, може се закључити да је неопходно познавање више области. За самостални развој видео игре треба бити упознат са технологијама које су актуелне, зато што постојеће технологије знатно убрзавају развој. Бројне елементарне операције (детекција кретања, исцртавање текстура на екрану, ефикасно учитавање ресурса и тако даље) су имплементиране тако да девелопер може да се посвети развијању механике саме игре.

Libgdx радни оквир пружа много могућности при развоју 2D видео игара, док *Node* и *MongoDB* пружају много могућности при развијању серверске апликације.

Архитектура апликације мора бити добро осмишљена како би развијање карактеристика (енг. *Features*) апликације било што једноставније. Добро установљена архитектура доноси униформност при развијању апликације, тако да нови развојни инжењер на пројекту треба да се упозна са шаблонима који се користе и да их користи при даљем развоју апликације. ECS шаблон поред високе ефикасности пружа и униформни развој. Девелопери који развијају нова својства игре користећи ECS шаблон треба да логику новог својства представе ECS системом, а податке представе ECS компонентом.

Такође треба напоменути да се до развоја квалитетног софтвера долази кроз вишеструке итерације.

5. ЛИТЕРАТУРА

[1] Sanchez Cejas Alberto, Marquez Saltares David, “Libgdx Crossplatform Game Development Cookbook”, Packt Publishing, 2014

[2] Härkönen Toni, “Advantages and implementation of entity-component-systems“, Faculty of Information Technology and Communication Sciences, Tampere, 2019

[3] V8.dev, Septembar 2022. [online]. Available: https://v8.dev/

[4] Expressjs.com, Septembar 2022. [online]. Available: https://expressjs.com/
[5] Socket.io, Septembar 2022. [online]. Available: https://socket.io/docs/v4/
[6] Mongodb.com, Septembar 2022. [online]. Available: https://www.mongodb.com/
[7] Freecodecamp.org, Septembar 2022. [online]. Available: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/

Кратка биографија:

Александар Комазец рођен је у Врбасу 1995. год. Дипломски рад на Факултету техничких наука из области Електротехника и рачунарство – Ембедед системи и алгоритми одбранио је 2020.год.
Области интересовања су му развој видео игара коришћењем C++ и C# језика.

контакт:
aleksandarkomazec5@gmail.com