+

ВИСОКА ШКОЛА ЕЛЕКТРОТЕХНИКЕ И РАЧУНАРСТВАСТРУКОВНИХ СТУДИЈА

**Манић Давид**

**Реализација мрежне верзије рачунарске игре**

**,,Чистач мина“**

**применом Unity развојног окружења**

**- завршни рад –**



Београд, месец 20\_\_.

Кандидат: **Манић Давид**

Број индекса: **НРТ-77/14**

Студијски програм: **Нове рачунарксе технологије**

Тема: **Реализације мрежне верзије рачунарске игре ,,Чистач мина“ применом Unity развојног окружења**

Основни задаци:

**1. Технике умрежавања у играма. Серверска ауторизација.**

**2. Развој асета, логике игре и међусобно повезовање.**

**3. Публикација и тестирање.**

Ментор:

Београд, месец 20\_\_ годинe.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

м/др XxxxxXxxxxxx, проф./виши предавач/предавач ВИШЕР

**Резиме:**

Овај рад се бави развојем **видео** игре у Unity развојном окружењу. Као и методама реализације умрежавања у видео играма које се израђују у unity развојном окружењу. Један од кључних циљева је и остваривање серверског ауторитета, ради превенције могућности варања и избегавања приступа поверљивим подацима којим играч не би требао да има приступ. Овај рад се такође бави израдом Photon plugin сервера који пружа могућност реализације горе наведених захтева и израдом више начина контроле тока игре, као и методама тестирања игара.

**Кључне речи:** : Unity, Серверски ауторитет, Photon on-premise, Тестирање игара.

**ABSTRACT:**

This assignment follows development of game in Unity engine. Contains research of networking options for games that are developed inside of Unity engine. One of the key requirements are server authority for cheat prevention and avoiding exposing valuable data to player that he should not have access to, also network solution should provide easy scaling and development. This assignment also covers making of Photon server and custom plugins for multiple game logic flows. There is also some coverage of some methods for game testing.

**Key words**: Unity, Server authority, Photon on-premise, Game testing.

**САДРЖАЈ:**

[1. УВОД 2](#_Toc56717799)

[2. Технике умрежавања у играма и серверска ауторизациј 3](#_Toc56717800)

[2.1. Unity Networking 3](#_Toc56717801)

[2.2. Photon Unity Networking 3](#_Toc56717802)

[2.3. Master Server Framework 4](#_Toc56717803)

[2.4. Forge 4](#_Toc56717804)

[2.5. Dark Rift 5](#_Toc56717805)

[2.6. SpatialOS 5](#_Toc56717806)

[2.7. Photon plugins 5](#_Toc56717807)

[2.8. Сажетак 6](#_Toc56717808)

[3. Развој асета, логике игре и међусобно повезивање 7](#_Toc56717809)

[3.1. израда асета 7](#_Toc56717810)

[3.2. Израда Unity апликације 2](#_Toc56717811)

[3.2.1. Израда Главног менија 2](#_Toc56717812)

[3.3. Израда мултиплејер сцене 3](#_Toc56717813)

[3.4. Поставка сервера 4](#_Toc56717814)

[3.5. Израда сервера и логике игре 5](#_Toc56717815)

[3.5.1. Израда минималног plugin-a 5](#_Toc56717816)

[3.5.3. Израда MineSweepers сервера 7](#_Toc56717817)

[4. Публикација и тестирањ 8](#_Toc56717818)

[4.1. Microsoft azure 8](#_Toc56717819)

[4.2. Amazon web services 8](#_Toc56717820)

[Креирање Amazon EC2 сервиса 8](#_Toc56717821)

[4.3. Тестирање 10](#_Toc56717822)

[Тестирање Minesweepers-a 11](#_Toc56717823)

[1. ЗАКЉУЧАК 15](#_Toc56717824)

[2. ИНДЕКС ПОЈМОВА 17](#_Toc56717825)

[3. ЛИТЕРАТУРА 20](#_Toc56717826)

[4. Прилози 22](#_Toc56717827)

[5. изјава о академскoј честитости 24](#_Toc56717828)

# УВОД

Од самог почетка развоја видео игара, нисмо могли ни да замислимо њихов данашњи ниво и могућности као и међусобну повезаност. Експоненцијални раст *мултиплејер* игара је допринео развоју великог броја субкултура и окупља велики број фанова фасцинираних неком игром. Такође смо видели и игре које је сама заједница одржавала у животу својим доприносима, израдом *асета*, мапа, *модова*, као и целих *DLC*-ова. Игре као што су Counter-Strike 1.6, Warcraft 3, Skyrim, GTA и доста других су практично преплављене садржајем који су сами играчи направили. Исто тако растом *мултиплејер* игара се развио и *Е-спорт*. Једноставно ништа не може да се упореди са међусобним такмичењем са вашим пријатељима као и са људима широм света. Међународна такмичења која привлаче пажњу више милиона гледалаца као и више милионске награде, развиле су читаву индустрију која почиње да надмашује остале традиционалне спортове. Једно је сигурно, *мултиплејер* као тип игре постаје све учесталији и неопходнији за сваку игру. Потреба људи да се осете као део заједнице и да буду остварени у некој од игара у односу на остале саиграче, пружа невероватан осећај за милионе играча.

# Технике умрежавања у играма и серверска ауторизација

Unity пружа уграђену подршку за израду *мултиплејер* игара под називом UNET. Такође постоји могућност додавања библиотека или *plugin*-ова за комуникацију и умрежавање. Решења која ће бити обрађена :

1. UNET(Unity networking)
2. PUN(Photon unity networking)
3. Master server framework
4. Forge
5. Dark rift
6. SpatialOS
7. Photon plugins

## Unity Networking

Unity networking или UNet је уграђени део Unity engine-а. Као такав требало би да представља најбољу опцију од свих, али нажалост то није тако. Током година доста се мењао и развијао. У 2018. години донета је одлука за тотално одбацивање тренутног система и успостављање новог, који би требало да буде имплементиран до 2021. у прихватљивом стању за коришћење у развоју игара, пошто је тренутно у стању развоја.  
Нови систем под називом Connected Games, се развија у колаборацији са Google-ом. Заснива се на новом *ЕCS(Entity component system)* систему и *Multi-threding-у* као и коришћење Google cloud сервиса. Нови систем има за циљ да надмаши традиционалне проблеме који произлазе из *P2P* конекције. Уз то план је да у потпуности подржава серверски ауторитет. Тако да нови систем може да понуди доста тога и заиста може да буде револуција у развоју видео игара у Unity engin-у. Једини страх који инжењери имају је прилагођавање новом стилу програмирања који се заснива на *ЕCS, DOTS(Data oriented tech stack)* и *Multi-threading-у.*

## Photon Unity Networking

Photon је лидер у свету као *мултиплејер* платформа, која је погодна за игре свих величина. Photon cloud обезбеђује проширење сервера и повећање броја играча без потребе отварања нових сервера или куповине нових машина за *хостовање* истих. Photon нуди велики број решења:

*Мултиплејер*

* Realtime – Основа којој су надограђене остали производи Bolt, Quantum, PUN, која није везана за Unity и може се слободно користити за надоградњу.
* Bolt – За игре које захтевају велике брзине и мала кашњења, као што су *FPS/TPS*.
* Quantum – За игре које захтевају детерминистичку симулацију.
* PUN, За општу употребу.

Комуникација

* Chat
* Voice

Самостална решења

* Photon server – Пружа могућност самосталног подизања сервер. Као и самосталну израду додатака за сервер помоћу Photon plugin-ova.
* Photon plugin- пружа могућност надоградње Game/Room логике, као и пресретање *event*-a унутар игре, процесуирање акција играча и додавање нових *callback* функција.

## Master Server Framework

Master Server Framework је дизајниран да на лак начин покрене развој *back end*-a. Садржи имплементацију за аутентификацију, динамичку и аутоматску синхронизацију профила играча као и скалабилну и лако проширљиву архитектуру сервера и игре. Добра ствар је што овај plugin подржава серверски ауторитет али има и своје мане. *Game-Server* се израђује у Unity-у, што доприноси великој конзумацију ресурса серверa. Исто тако не подржава *loadbalancing* као ни *matchmaking*. Иза овог производа не стоји велика компанија. Самим тим нема велику подршку, као ни адекватну документацију.

## Forge

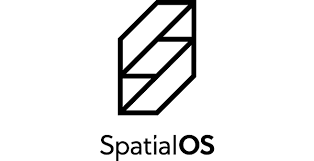
Froge је скуп библиотека за умрежавање. Користи се за развој *мултиплејер* игара и апликација које комуницирају преко интернета или унутар локалне мреже. Такође подржава *cross platform* комуникацију. Могуће је *хостовати* сервере без новчане надокнаде за конкурентне кориснике. Forge ради као Unity инстанца што даје лак начин за тестирање и *дебаговање* кода као и коришћење свих компоненти и алата које Unity може да понуди. Лоша ствар у томе је што то додаје велику потрошњу ресурсе сервера, због засебне инстанце сваке собе. Forge садржи имплементацију за *level-e, achievement-e*, профиле, економију, и табеле.

## Dark Rift



DarkRift подржава серверски ауторитет и лако је проширљив добро је оптимизован и нема ограничења што се тиче конкурентних корисника али што се тиче подршке њу чини само један члан који стоји иза овог пројекта и не подржава *matchmaking* у бесплатној верзији, као и комплетан недостатак loadbalancing-a.

## SpatialOS



SpatialOS помера границе оптимизације сервера као и њених компоненти, пружа невероватне иновације што се тиче *мултиплејер* решења. Могуће је развити игре огромних размерa уз помоћ њихове технологије за раздвајања под називом “Multiserver offloading” и скалирање сервера. Иако подржава још много ствари као што су већ изграђени модули за вашу игру. Ова опција се чини као неадекватна пошто је више намењена за *open world* игре и игре које захтевају велики број играча или *NPC-ева*.

## Photon plugins

Photon plugins као део Photon компаније и њихових решења доста одступа од осталих решења због своје флексибилности, коју остали производи не могу да пруже. PUN, Realtime, Quantum и Bolt су у суштини исти што се тиче развоја игре, само су посебно оптимизовани за одређене типове игара. Photon server се ослобађа уобичајеног начина развоја игре, у коме је један члан собе *мастер* или *хост*, и остали играчи одржавају синхронизацију сцене. Једини начин за превенцију варања је да се клијенти међусобно валидирају, пошто не постоји неки виши ауторитет као што је сервер. Photon server пружа могућност писања кода на самом серверу као и серверски ауторитет, што може бити јако корисно за превенцију варања. Као и безбедност информација, коју играч не би требао да поседује. Могућност креирања више *plugin-а* отвара могућности за креирање различитих врста соба као и различитих врста логика игре, на модуларан начин.

## Сажетак

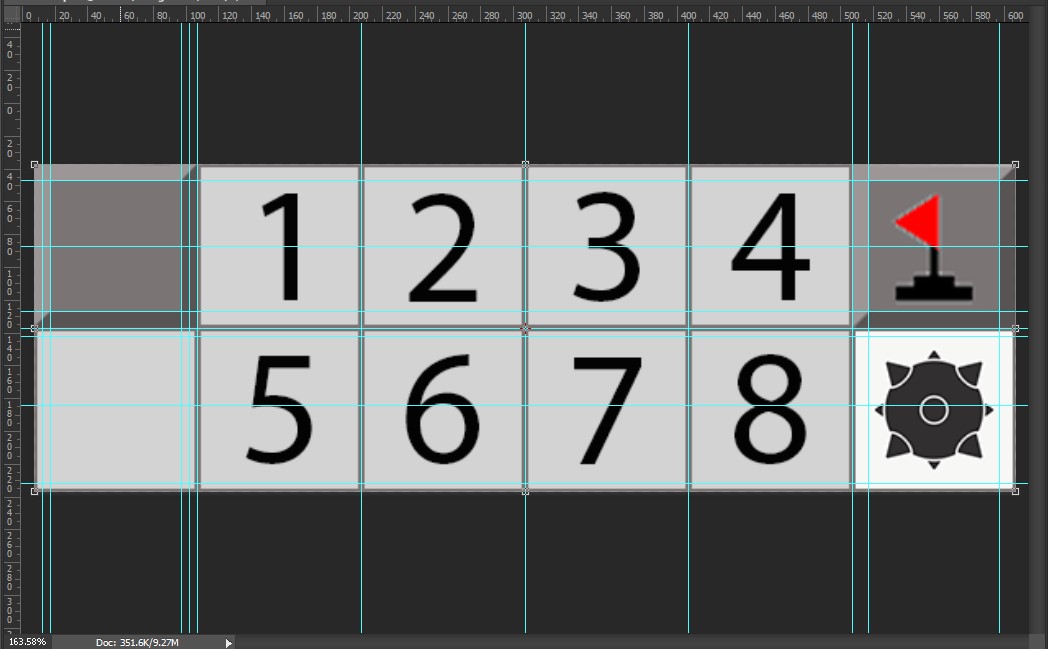
Ово су само неки од начина за умрежавање, постоје још доста њих за које не можемо да кажемо да имају адекватну подршку или да технички пружају лак развој, доста функционалности и стабилност системa. Са тиме што Dark rift, Forge и Master server framework, пружају пристојну подршку и капацитете иако не могу да се пореде са гигантима као што су Unity, Photon или SpatialOS. Из простих разлога зато што су развијени од свега пар чланова и немају подршку у виду инжењера или инфраструктуре коју поседују Unity, Photon или SpatialOS. Пре свега мислећи на *cloud* и *loadbalancing*.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Unity | Photon | SpatialOs | Master server | Forge | Dark Rift | Photon on-premise |
| Server authority | ✓ | X | X | ✓ | ✓ | ✓ | ✓ |
| LoadBalancing | ✓ | ✓ | ✓ | X | X | X | ✓ |
| Headless | ✓ | ✓ | ✓ | X | X | ✓ | ✓ |
| Match making | ✓ | ✓ | ✓ | ✓ | poor | ✓ | ✓ |
| Free CCU | 20 | 20 | Free Development | No limit | No limit | No limit | 100 |

# Развој *асета*, логике игре и међусобно повезивање

## израда *асета*

Што се *асета* тиче игра није захтевна. Потребно је израдити *спрајтове* за таблу, тј поља табле, као и главни мени.



Слика 3.1.1 – Израда поља табле.

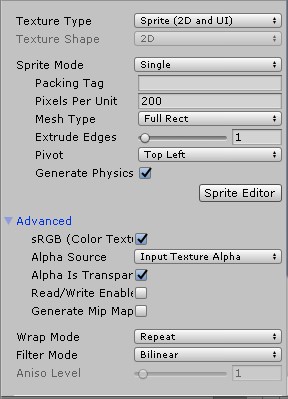
Исто тако можемо користити засебно израђене *спрајтове* за дугмад и *слајдере*

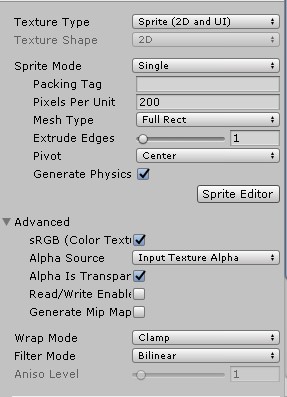
. 

Слика 3.1.2 – *ноб* за клизач

Слика 3.1.3 – Дугме

Исти *спрајт* можемо користити и за позадину главног менија, али је потребно одвојити га како би могао засебно да се манипулише.



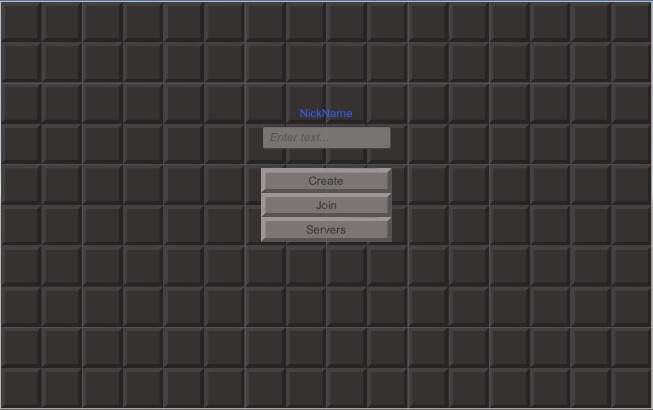
Слика 3.1.4 – Подешавање за позадину

Слика 3.1.5-Подешавање за дугме

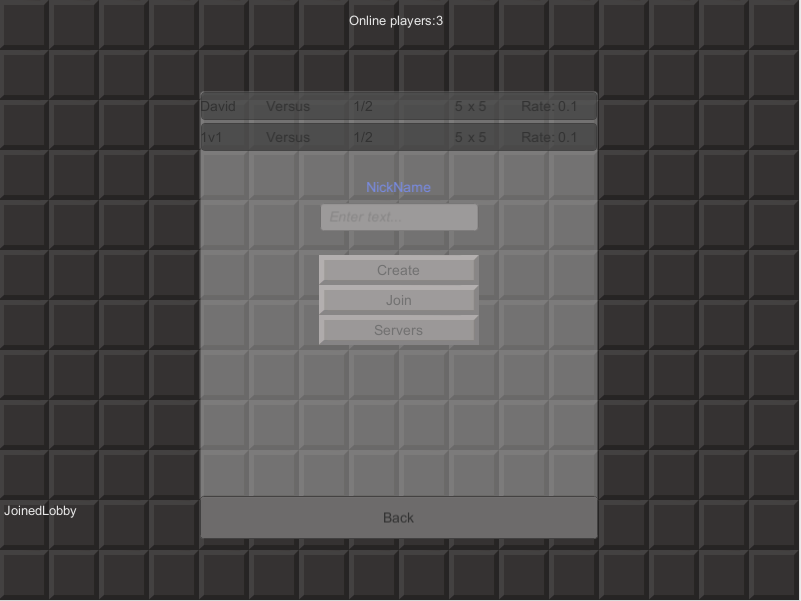
За преостале *асете* кориштене су уграђе компоненте Unity *енџина*.

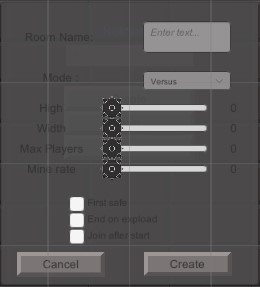
## Израда Unity апликације

### Израда Главног менија



Слика 3.2.1 – Главни Мени





### Израда *мултиплејер* сцене

*Мултиплејер* сцена се састоји од почетног панела тј. менија који се приказује пре игре у коме су приказани играчи који чекају почетак игре као и приликом изласка из игре, табле која приказује минскиа поља као и панела који приказује тренутно стање игре као и табелу. Скрипте које су потребне у позадини су:

NetwrokController - који обавља функцију комуникације између сервера и клијента.

TileFactory - служи за генерисање потребних *спрајтова* и *асета* за игру тј. таблу.

InputController - служи за детекцију уноса команди као и за позиционирање камере и њено померање.

Board - служи као модел класа табле, садржи методе за интеракцију са њом као и њено генерисање.

Menu - модел класа за мену панел, садржи логику са приказивање играча који се тренутно налазе у соби, као и дугме за напуштање игре.

StatusPanel – Служи за приказ тренутног стања игре, као и играча и њихових респективних поена.

## Поставка сервера

* + 1. Преузимање сервера са званичног сајта <https://www.photonengine.com/>
    2. Покретање Photon Control-a. “deploy\bin\_Win64\PhotonControl” након чега ће се појавити *system tray* иконица преко које можемо контролисати сервере.
    3. Преко опције “Game Server IP Config”. Бирамо дали желимо да *хостујемо* на локалној или јавној IP адреси. У случају да *хостујемо* на јавној такође је потребно отворити одговарајуће портове: 4530,4531 за *TCP* и 5055, 5056 за *UDP*. *Хостовање* на локалној IP адреси је практично приликом самог развоја, али не пружа могућност да се други клијенти прикључе. Такође могуће је покренути више клијената на локалној машини ради самог тестирања *мултиплејер* функционалности.  
       Такође постоји могућност да ће бити потребно подесити *Firewall* подешавања.
    4. Покретање сервера. “LoadBalancing(MyCloud)->Start as application”. Сада је *Master* сервер активан. И могуће га је тестирати и проверити да ли постоје неке препреке при конекцији са њим као и са Game сервером.
    5. Тестирање конекције. “LoadBalancing(MyCould)->Run Тestclient”.
    6. Додавање лиценце – Бесплатна верзија без лиценце подржава до 20 истовремених корисника, могуће је проширити капацитет на 100 истовремених корисника креирањем налога на званичном сајту при чему се добија бесплатна лиценца.   
       За веће капацитете потребно је купити званичну лиценцу.  
       Инсталација лиценце се врши копирањем фајла лиценце у *директоријуму* сервера на “deploy\bin\_Win64” и поновним покретањем Photon control-a.

Након успешног постављања сервера потребно је убацили библиотеке за нашу игру и извршити додатна подешавања и повезивања.

Копирати потребне датотеке у:

"deploy\plugins\{pluginName}\{pluginVersion}\bin\"

Минимална поставка мора да садржи бар ова два *фајла*.

"deploy\plugins\{pluginName}\{pluginVersion}\bin\{pluginDllFileName}.dll"

"deploy\plugins\{pluginName}\{pluginVersion}\bin\PhotonHivePlugin.dll"

Ако се приликом израде сервера није користила верзија могуће је изоставити {pluginVersion}.

Отворити: "{pluginsSdkFolder}\deploy\LoadBalancing\GameServer\bin\Photon.LoadBalancing.dll.config"

И додати:поновно покренути сервер.

## Израда сервера и логике игре

### Израда минималног plugin-a

* 1. Креирање пројекта у Visual Studio:
     1. Тип пројекта “Class Librery(.NET Framework)“.
     2. Framework: “.NET Framework 4”
     3. И одабрати жељено име пројекта, у овом случају “MineSweeperPlugins”.
  2. Додавање потребне библиотеке: "PhotonHivePlugin.dll".  
     Додавање из Visual Studio-а:

1. Десним кликом на "References" у solution explorer-u.
2. "Add Reference.."
3. Изабрати "Browse"
4. Кликнути "Browse" и изабрати "{pluginsSdkFolder}\src-server\Plugins\lib\PhotonHivePlugin.dll"
5. Потврдити са "OK"
   1. Додавање класе plugin-a која порширава класу PluginBase.
   2. Постављање имена *plugin*-a.  
      
   3. Преклапање OnCreateGame функције ради тестирања.



* 1. Прављење Plugin Factory-a:
  2. Додавање нове класе MineSweepersFactory
  3. Битно је да класа буде јавна.
  4. Имплементирати IPluginFactory интерфејс.  
     
     1. Build Solution:

1. Приликом изградње пројекта пожељно је користити „Build Events“, како би се цео процес аутоматизовао током израде сервера.
   1. Visual Studio -> Project -> {SolutionName} Properties -> Build Events
   2. Pre-build event: {ServerDir}\deploy\bin\_Win64\PhotonSocketServer.exe /stop
   3. Post-build event:
2. {ServerDir}\deploy\bin\_Win64\
3. powershell start-process "cmd ""/c PhotonSocketServer.exe /run LoadBalancing""" -windowstyle hidden
   1. Такође у Post-build event можемо додати команду за самостално копиранје датотека које су резултат саграђеног пројекта:
4. xcopy /Y /Q "$(TargetDir)\*.\*" "$(SolutionDir)..\..\..\deploy\Plugins\MyFirstPlugin\bin\"
5. Или изменом Build путање:
6. Visual Studio -> Project -> {SolutionName} Properties -> Build -> Output path

### Израда MineSweepers сервера

Сервер има за циљ да води комплетну симулацију игре док клијенти само врши интеракцију и шаље податке, коју онда сервер прослеђује осталим играчима у случају да је потез валидан. Сваки играч прима потезе осталих играча и игра се симулира и на њиховом рачунару. Сервер се састоји од :

MineSweepersFactory – је обавезни део сваког Photon on premise сервера. И служи са креирање и иницијализацију соба. Свака соба је приморана да користи тј. наследи PluginBase класу која садржи основне функције које се могу прегазити(*override*).

MineSweepersPlugin – Ова апстрактна класа пружа основну имплементацију за оба *мода* игре. Као и основну логику саме игре.

MineSweepersCoop – је *мод* игре у коме нема рачунања резултата или учинка. Намењен је за рекреативну игру или уживање са пријатељима. Може бити практичан за огромне табле и велику густину мина, такве партије могу дуго да потрају и притом буду велики изазов.

MineSweepersVersus – за ралику од *coop* мода у овом *моду* се врши бодовање и суштина је скупити највише бодова. За свако отворено поље или постављену заставицу играч бива награђен одређеном количином бодова. Док у случају ако се игра не завршава при отварању мине (што се одабира приликом креирања собе), играч добија негативне поене.

Board – Ова класа садржи комплетну имплементацију логике игре. Као и паковање података у речнике за слање преко мреже.

Score – Има за задатак да води рачуна о стању поена играча као и да врши синхронизацију и обавештавање о променама резултата.

TurnManager – Има за циљ да врши контролу потеза тј. редослед играња играча. Али приликом тестирања закључено је да ограничавање играча на потезе и принудно чекање док други играч одигра потез одузима од доживљаја игре. И овај део се више не користи, али служи као лак притуп листи играча и манипулацијом са њима.

# Публикација и тестирањ

## Microsoft azure

## Amazon web services

Amazon Web Services (AWS) је најобимнија и широко прихваћена светска *cloud* платформа, која нуди преко 175 услуга из *data* центара широм света. Милиони корисника почевши од брзо растућих *стартуп*-ова, великих компанија, и водећих владиних агенција, користе AWS како би смањили трошкове, постали флексибилнији у раду, и брже уводили новине.

AWS има значајно више услуга, и функција унутар тих услуга него било који други *cloud* провајдер – од инфраструктурних технологија попут рачунања, складиштења, и база података – до нових технологија као што су машинско учење и вештачка интелигенција, *data lake* и аналитике, и *Internet of Тhings(IOT)*. Ове технологије омогућавају брзo, лакo, и економичније померање постојећих апликација на *cloud*, као и изградњу скоро свега што можете да замислите. AWS такође има најдубље функционалности унутар сервиса које су наменски направљене за различите типове апликација тако да ви можете изабрати праву алатку за ваш посао, како би остварили најбољу цену и перформанс.

Amazon Elastic Compute Cloud (Amazon EC2) је веб сервис који пружа рачунаре променљивог капацитета у cloud-у. EC2 је једноставан *веб* сервис који вам омогућава да добијете и подесите могућности у минималан напор. Пружа вам потпуну контролу над вашим машином и њеним ресурсима. Amazon EC2 смањује потребно време за добијање и подизање нових сервера у периоду од свега неколико минута, дозвољавајући вам да брзо смањујете или повећате капацитете ваше капацитете, сразмерно вашим потребама. Велика предност Amazon EC2 је што вам пружа могућност плаћања само ресурса које стварно користите.

### Креирање Amazon EC2 сервиса

* Креирање налога на <https://aws.amazon.com/>,
* Улоговати се на налог.
* Sign in to the Console или My Account - > AWS Management Console
* All services -> Compute -> EC2
* Пре самог креирања битно је да подесимо регион или место на коме ће бити смештена наша машина. То можемо учинити преко падајућег менија у горњем десном углу.
* Step 1: Choose an Amazon Machine Image (AMI)  
  Користићемо бесплатну верзију виртуалне машине тако да ћемо штиклирати “Free tier only”.

Изабрати **Microsoft Windows Server 2019 Base**

* Step 2: Choose an Instance Type  
  Изабрати t2.micro(free tier eligible)
* Сегменти 3,4,5 нису круцијални за рад масине нашег типа тако да се могу прескочити.
* Step 6: Configure Security Group  
  Add Rule:

Custom TCP, Port Range: 4530-4531, Source:0.0.0.0/0  
Custom UDP, Port Range: 5055-5056, Source:0.0.0.0/0

* Review and Launch
* Launch
* Create new key pair. (Битно је сачувати направљени кључа пошто ће бити коришћен за приступање машини)
* Launch Instance
* View Instance
* Сачекати Status Checks

**Подешавање IP адресе**

* Десни клик на машину унутар конзоле.
* Networking -> Manage IP Adresses
* Allocate an Elastic IP
* Изабрати Amazon's pool of IPv4 addresses
* Allocate
* Associate Elastic IP address
* Изабрати инстанцу сервера и кликнути Associate.

**Подешавање сервера**

* Из конзоле селектовати машину и притиснути Connect.
* Download Remote Desktop File.
* Get Password.
* Изабрати датотеку кључа и изабрати Decrypt password.
* Копирати password и покренути преузети RDP фајл.
* Улоговати се на налог администратора помоћу дешифроване шифре.
* Сада је потребно копирати датотеке нашег Photon сервера на виртуалну машину.
* Инсталирати .Net Framework 3.5
* Променити IP адресу Photon сервер са локалне на јавну
* Покренути сервер.

## Тестирање

Тестирање игара, је круцијалан део развоја игара, како би се обезбедио висок квалитет игре и смањио број непријатних искустава са којима корисник може да се суочи. Примарна функција тестирања игара је откривање и документовање софтверских дефеката или нежељених понашања, које најчешће називамо баговима. Тестирање софтвера поготово забавног карактера је високо техничко поље које захтева компјутерску експертизу, аналитичку способност и вештине критичног процењивања. Последњих година поље тестирања игара постаје изузетно напорно како због високих очекивања и стандарда корисника, као и због високих цена и количине ресурса која компаније мора да издвоји да би развила квалитетан производ.

Не постоји стандард за методе тестирања игара, већина методологија су развијени од стране самих програмера, ако и издавача. Методологије се константно усавршавају и разликују се за различите типове игара, Неке методе као што су *Unit testing*, које се користе најчешће у стандардном тестирању софтвера, нису адекватне или довољне приликом тестирања. Тако да поред ове методе можемо користити неке од следећих:

* **Functionality testing**  **(Тестирање функционалности)** је најчешће асоцирано за тестирање игара, јер подразумева играње игре у неком облику. Испитивање функционалности не захтева широко техничко знање. Испитивање функционалности тражи опште проблеме унутар саме игре или њеног корисничког интерфејса, као што су проблеми са стабилношћу, проблеми са механиком игре и поузданост *асета*.
* **Compliance testing (Испитивање усаглашености)** је разлог постојања лабораторија за тестирање игара. Платформе имају строге техничке захтеве које сваки производ мора да испуни.
  + **Sony** - Technical requirements checklist (TRC)
  + **Microsoft -** *Xbox Requirements* (XR)
  + **Nintendo –** Lotcheck

Неки од ових захтева су високо технички и спадају ван опсега тестирања игара, као и придржавање стандарда о порукама грешке, коришћењем података из меморије, такође поступање са заштићеним материјалима и ауторским правима. Довољан је само један прекршај правила или не испуњен услов приликом подношења одобрења за лиценцу да би се игра одбила. Што може продужити време тестирање и поновно чекање за подношење захтева.  
у ову категорију се може уврстити и регулаторна тела као што су  ESRB и PEGI. Која имају за циљ да регулишу непримерен садржа који подносилац мора да пријави, како би се одредила дозвољена старосна група за дати производ.

* **Compatibility testing** (Испитивање компатибилности) је обично потребно за видео игре на PC платформи, обично се тестирање врши при самом крају развоја, јер велики део компатибилности зависи од коначне верзије игре. Често се раде два круга тестирања компатибилности – рано у *бета* верзији како би било довољно времена да се проблем реши, као и касније у *бета* верзији или током самог издавања. Тим за тестирање компатибилности тестира главну функционалност игре на различитим конфигурацијама хардвера. Издавач обично даје списак комерцијално важног хардвера.  
  Испитивање компатибилности осигурава да се игра покреће у различитим конфигурацијама хардвера и софтвера. Хардвер обухвата марке различитих произвођача и разне улазне периферне уређаје као што су *gamepads* и *joystics*.

Испитивачи такође процењују перформансе и резултати се користе за рекламирање минималне системске спецификације. Проблеме са компатибилношћу или перформансом може исправити програмер, или у случају застарелог хардвера и софтвера може се одустати од подршке.

* **Localization testing** (Тестирање локализације) има за циљ тестирање правилног превода текста у игри. Док сама функционалност текста генерално спада у функционално тестирање, одељење за осигурање квалитета може запослити искључиво тестере који би се бавили само овим делом. Неретко је случај да се испитивачи потичу из саме земље за коју је игра намењена, како би се обезбедио најбољи и најпоузданији квалитет превода.
* **Soak testing** у контексту видео игара представља остављање игре у раду током дужег временског периода у различитим *модовима* рада. Као што су остављање игре у празном ходу, паузирану или у главном менију. Ово тестирање не захтева интеракцију корисника након почетног подешавања. Могу се користити додатни алати како би се аутоматизовала симулација понављајућих радњи, попут кликова мишем или уношења одређених команди. Soaking тестови могу детектовати „цурење меморије“ или грешке приликом заокруживања које се могу манифестовати само после дужег периода времена. Овај тест је један од захтева за тест усаглашености(compliance test).
* **Beta testing (Бета тестирање)** се врши током *бета* развојне фазе. Често се ово односи на прву јавно доступну верзију игре. Јавне *бета* верзије су ефикасне јер хиљаде обожаватеља могу пронаћи грешке које програмери нису тестирали. Притом се може заобићи плаћање компанија које врше тестирање софтвера.
* [**Regression testing**](https://en.wikipedia.org/wiki/Regression_testing) **(Регресијско тестирање)** се врши након што програмери утврде грешку. Одељење за осигуравање квалитета проверава да ли је грешка и даље присутна, затим покреће сличне тестове да види да ли је поправка покварио нешто друго. Та друга фаза се често назива *хало* тестирање. Оно подразумева тестирање у околини саме грешке како би се открила нека друга грешка која је самом изменом могла да настане.
* **Load testing ( Тест оптерећења)**  тестира границе система, као што су број играча на ММО серверима, број активних текстура на екрану или број нити које се покрећу у одређеном програму. За тестирање оптерећења потребна је или велика група тестера или софтвер који опонаша велику активност. Тестирање оптерећења такође мери способност апликације да правилно функционише под оптерећењем.
* **Multiplayer testing (Тестирање за вишe играча)** може подразумевати одвојену екипу за проверу квалитета игре, ако игра има значајне делове за више играча. Ово тестирање је чешће код PC игара. Ово тестирање има за циљ да осигура рад свих метода повезивања(модем, LAN, интернет)

### Тестирање Minesweepers-a

Иако немамо потребе да радимо неке од претходно наведених тестова или их је немогуће применити у нашем случају, постоје функционалности које морамо тестирати. Попут *мултиплејер* теста , функционалног теста, теста оптерећења и *бета* теста.

*Мултиплејер* тест – Предност је у томе што већ користимо широко распрострањену и већ тестирану опцију за умрежавање, Photon unity networking. Наше циљ је да обезбедимо доступност нашег сервера, што значи да отворимо потребне портове и тестирамо саму конекцију. Проблеми могу настати приликом конекције сваког корисника у мрежи или *firewall*-у, али поводом тог питања ми не можемо да предузмемо пуно, и морамо се ослонити на самог корисника да самостално уклони грешке у правилима *firewall*-a и самостално успостави правила мрежног саобраћаја. Било који други вид проблема сем конекције на мастер сервер и game сервер спада у функционално тестирање. Клијент се засебно конектује на *game* и *master* сервер тако да је потребно засебно тестирати ова два случаја.

Функционално тестирање – Иако игра није много комплексна постоје неколико аспеката које морамо тестирати.

* Главног менија – рад главног менија је поприлично једноставан. Простим коришћењем можемо утврдити да ли све његове функције раде оно што би требале. Потребно је више пажње посветити различитим резолуцијама игре. Како би се проверило да ли се сви елементи правилно приказују.
* Креирање собе – Поред провере приказа елемената панела за креирање собе, морамо проверити да ли се сва подешавања правилно преносе на сервер и да ли сам сервер креира собе по наведеним спецификацијама.   
  За овај случај би можда било погодно користити Unit тестове, али проблеми могу настати при самој конекцији која је асинхрона. Такође не можемо проверити све параметре са стране клијента, тако да морамо пратити лог конзолу сервера како би смо утврдили да је све протекло како треба. Пре тога је наравно потребно да документујемо сам процес креирања у конзоли. Као и осталу међусобну комуникацију.
* Прикључивање соби (са започетом игром или не) – Потребно је проверити да ли клијент може да уђе у собу ако су сви услови испуњени као и спречавање у случају да нису, уз прописно враћање у *лоби*.  
  Ако је игра започета и партија подржава прикључивање након старта, треба обезбедити правилну синхронизацију свих играча. Као и постављање стања игре новог клијента као и код осталих играча. Што подразумева прављење табле и слање свих отворених поља или одиграних потеза. Такође потребно је убацити играча у ток игре, и додати га у саму табелу ако игра подразумева мерење резултата, као и остварити синхронизацију резултата са осталим играчима.
* Механике игре – Правила игре су поприлично једноставна, као и њена механика. Игра се састоји од поља која чине таблу.   
  Приликом отварања поља (десни клик) приказује се број мина која окружује то поље или сама мина која експлодира. У случају да нема ни једна у окружењу отварају се сва суседна поља, која даље могу поновно проузроковати отварање још поља.   
  Приликом постављања заставице (десни клик) поље се означава као резервисано за мину, са тим што у неким случајевима игре где је играчу није дозвољено да означи погрешно поље (поље које није мина), поље се отвара и играч добија негативне бодове за погрешан потез, ако игра подржава мерење резултата.   
  У случају да играч притисне десни клик на поље које је отворено и број заставица у околини је једнак броју који се налази у том пољу сва преостала поља која нису отворена се отварају.  
  У случају да игра не подржава грешку приликом отварања мине игра се завршава.  
  Ако игра има заштиту против отварања мине из првог потеза, потребно је обезбедити заштиту у том случају што се може постићи измештањем отворене мине у друго поље и ажурирањем околних поља на обе локације.  
  Такође потребно је проверити да ли зумирање и померање камера функционише у постављеним границама.

Детектовање завршетка игре – потребно је проверити да ли у сваком сценарију се игра успешно завршава. Иако играч не успе беспрекорно да реши игру, а партија подржава означавање поља заставицом иако нису мина, као и ако партија подржава наставак игре иако играч погрешно отвори мину.

Отворена поља + број мина – број експлодираних мина = висина\*ширина табле

Иако је формула поприлично једноставна грешка се може јавити приликом измене неке од ових варијабли. Такође не води се рачина о броју заставица пошто би то могло изазвати нежељени крај. У случају да број преосталих поља је једнак броју мина није потребно означити пошто се аутоматски подразумева да се ради о минама.

Излазак из игре/собе – Приликом напуштања потребно је проверити да ли се стање партија правилно ажурира. Да ли је играч који је напустио игру избацио из тока игре као и из табеле резултата. И тиме спречило беспотребно чекање у случају да се ради о партији која има предвиђено време за потез сваког играча. Исто тако треба проверити дали TTL период правилно функционише и избацивање играче које су изгубили конекцију.

# ЗАКЉУЧАК

У зависности од одабраног решења можемо у многоме скратити пут развоја видео игра. Коришћењем већ постојећих технологија не само да добијамо на времену него и поузданости система што у пракси доста значи. Велике компаније попут Unity, Photon-a, Googla, Amazona и Microsoft-a, могу много да обезбеде својим сервисима, као што су лака публикација, и *loadbalacing;* који пуно значи за мале студије који немају ни инфраструктуру, ни базу играча да би имали више сервера широм планете. Оно што је сигурно да ће *мултиплејер* наставити да се развија и напредује. Како би померио границе могућности и обезбедио све боље услове за развијање све умреженијих и комплекснијих игара.

# ИНДЕКС ПОЈМОВА

# ЛИТЕРАТУРА

[1] https://assetstore.unity.com/packages/tools/network/master-server-framework-71391

[2] https://github.com/alvyxaz/barebones-masterserver/wiki

[3] https://forum.unity.com/threads/master-server-framework.430570/

[4] https://assetstore.unity.com/packages/tools/network/forge-networking-remastered-38344

[5] https://improbable.io/multiplayer-networking

[6] https://www.photonengine.com/

[7] https://docs.unity3d.com/Manual/UNet.html

[8] https://assetstore.unity.com/packages/tools/network/darkrift-networking-2-95309

[9] https://doc.photonengine.com/en-US/server/current/getting-started/photon-server-in-5min#game\_server\_ip\_config

[10] https://doc.photonengine.com/en-US/server/current/plugins/manual

[11] https://doc.photonengine.com/en-us/server/current/plugins/getting-started

[12] https://en.wikipedia.org/wiki/Game\_testing

# Прилози

# изјава о академскoј честитости

**ИЗЈАВА О АКАДЕМСКОЈ ЧЕСТИТОСТИ**

|  |  |
| --- | --- |
| **Студент (име, име једног родитеља и презиме):** |  |
| **Број индекса:** |  |

Под пуном моралном, материјалном, дисциплинском и кривичном одговорношћу изјављујем да је завршни рад, под насловом:

1. резултат сопственог истраживачког рада;
2. да овaj рад, ни у целини, нити у деловима, нисам пријављиво/ла на другим високошколским установама;
3. да нисам повредио/ла ауторска права, нити злоупотребио/ла интелектуалну својину других лица;
4. да сам рад и мишљења других аутора које сам користио/ла у овом раду назначио/ла или цитирао/ла у складу са Упутством;
5. да су сви радови и мишљења других аутора наведени у списку литературе/референци који је саставни део овог рада, пописани у складу са Упутством;
6. да сам свестан/свесна да је плагијат коришћење туђих радова у било ком облику (као цитата, прафраза, слика, табела, дијаграма, дизајна, планова, фотографија, филма, музике, формула, вебсајтова, компјутерских програма и сл.) без навођења аутора или представљање туђих ауторских дела као мојих, кажњиво по закону (Закон о ауторском и сродним правима), као и других закона и одговарајућих аката Високе школе електротехнике и рачунарства струковних студија у Београду;
7. да је електронска верзија овог рада идентична штампаном примерку овог рада и да пристајем на његово објављивање под условима прописаним актима Високе школе електротехнике и рачунарства струковних студија у Београду;
8. да сам свестан/свесна последица уколико се докаже да је овај рад плагијат.

У Београду, \_\_. \_\_. 201\_. године

Својеручни потпис студента

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_