



SVEUČILIŠTE U ZAGREBU



Fakultet
elektrotehnike i
računarstva

Diplomski studij

Računarstvo

Akademska godina
2022/2023

Umrežene igre

Mrežna jezgra igrnih sustava



Sadržaj

- Uvod
- Oficijelna sučelja za razvoj višekorisničkih umreženih igara
- Sučelja za razvoj višekorisničkih umreženih igara razvijena od trećih strana
- Photon Unity Networking 2

Uvod

- Distribuirani sustavi su teški za razvoj i održavanje
- Kako bi se olakšao razvoj i održavanje ovakvih sustava koriste se specijalizirana programska sučelja
- Programska sučelja za razvoj višekorisničkih umreženih igara koriste se za olakšavanje razvoja višekorisničkih umreženih igara (ako se u prezentaciji spominje samo termin sučelje misli se na ovaj termin)
 - **Niža razina** omogućuje **direktno korištenje mrežnih protokola** i specifičnih funkcionalnosti te zahtijevaju dobro razumijevanje funkcioniranja računalne mreže
 - **Viša razina** sakriva kompleksnost funkcioniranja mreže **gotovim rješenjima** i omogućava programerima da se **fokusiraju na izazove u razvoju samih igara**
 - **Srednja razina** kombinira funkcionalnosti niže i više razine
- U okviru ovog predavanja fokus na programskim sučeljima za Unity pogonski sustav

Netcode

- „Netcode“ – termin kojim se generalno opisuje implementacija mrežnog dijela videoigre
- Koriste ga i razvijatelji i igrači
- Može se implementirati od najnižeg nivoa (programska sučelja niske razine koja rade na razini uspostave pojedine priključnice) sve do najvišeg nivoa (programska sučelja visoke razine koja sakrivaju gotovo sve aspekte umreženost od razvijatelja igre)
- Razine nisu diskretne već postoji ih više, različita programska sučelja i biblioteke mogu nuditi manje ili više funkcionalnosti s više ili niže razine
- Srednja razina omogućava pristup i upravljanje mrežnim procesima, dok istovremeno nudi određenu razinu apstrakcije radi lakšeg razumijevanja i manjeg opterećenja prilikom izrade igre

Popis sučelja za razvoj u Unity sustavu

- Dugi niz godina programsko sučelje za višekorisničke igre unutar Unity sustava je bilo zapostavljeno u razvoju (UNET)
- Zbog toga je Unity već dugo u procesu izrade novih biblioteka za podršku višekorisničkom igranju (MLAPI, HLAPI...)
- Netcode for GameObjects objavljena 2021. godine te je ista još u razvoju (razvijen iz MLAPI-ja)
- Rezultat toga su veliki broj razvijenih i dobro podržanih sučelja za razvoj koje su razvile druge tvrtke ili zajednica razvijatelja otvorenog koda
- Oficijelna sučelja (razvijena od strane Unity Technologies)
 - Transport – sučelje niske razine
 - Netcode for GameObjects – sučelje srednje/visoke razine (razvijeno iz MLAPI-ja)
 - Netcode for Entities – sučelje srednje/visoke razine razvijeno za Entity Component System (programski sustav koji se temelji na podatkovno orientiranom programskom složaju za Unity)
 - Multiplayer tools – dodatni alati za profiliranje, detektiranje grešaka, praćenje pozicije na klijentu i poslužitelju i slično...
- Sučelja trećih strana
 - Mirror – Open Source Networking for Unity – sučelje visoke razine otvorenog koda temeljeno na UNET-u
 - Forge Networking – sučelje niske razine
 - DarkRift 2 – sučelje niske/srednje razine
 - Fish-Networking – sučelje visoke razine
 - Networking and Serialization Tools (TNet 3) – sučelje niske razine
 - Photon Quantum – determinističko sučelje visoke razine
 - Photon Fusion – sučelje visoke razine
 - Photon Unity Networking 2 – sučelje visoke razine



Faktori razvoja višekorisničkih umreženih igara

- Ključni faktori koji se trebaju uzeti u obzir prilikom razvoja višekorisničkih igara te njihovog životnog ciklusa
- Različiti mrežna sučelja trebaju implementirati podskup ili cijeli skup ovih funkcionalnosti
- Tolerancija na kašnjenje
 - Različite igre imaju različite zahtjeve
 - Mogućnost procesiranja poruka i događaja koji imaju različite zahtjeve
- Broj igrača po sjednici
 - Definira ga tip igre
 - Definira ga i arhitektura (C-S ili P2P)
 - Veći izazovi što je više igrača
- Skalabilnost sinkronizacija stanja
 - Ne moraju svi klijenti biti kompletno sinkronizirani sa svim drugima, pogotovo onima na koje ne mogu direktno utjecati (područje interesa)
 - Postoje razne tehnike s kojima se može utjecati na skalabilnost (posebno predavanje)
- Modularnost izbora
 - Ako se odabere određena arhitektura ili tehnologija, koliko je moguće promijeniti odluku kasnije?
 - Primjerice P2P prebacivanje na C-S arhitekturu

Faktori razvoja višekorisničkih umreženih igara

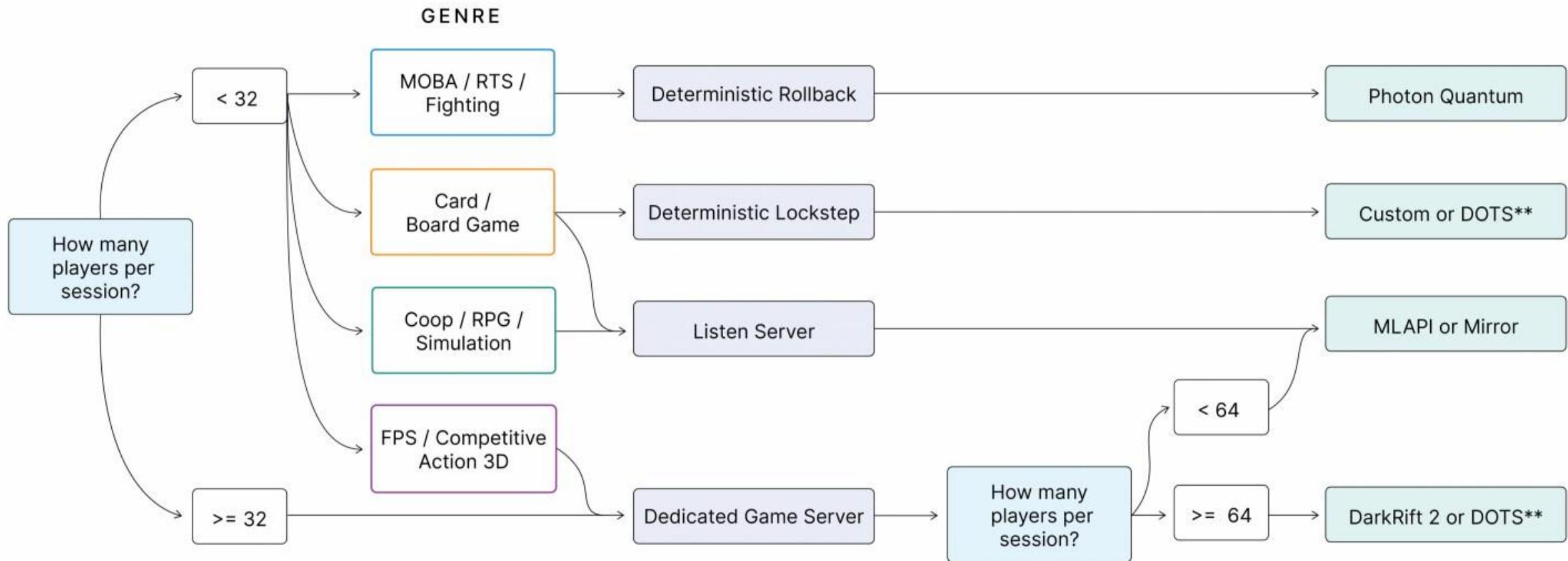
- Preciznost
 - Kolika preciznost je potrebna za određenu igru?
 - Prevelika preciznost može povećati opterećenje mreže i procesora
 - Potrebno prilagoditi samom tipu igre
- Troškovi
 - Kakav model mrežni je održiv?
 - Postoji li dovoljno sredstava za posebne (engl. dedicated) poslužitelje?
 - Može li se optimizirati dodatno mrežni kod igre da se troškovi smanje?
- Složenost izrade
 - Jeden od najsloženijih problema
 - Jesu li potrebna posebna rješenja ili su postojeća dobra (niska ili visoka razina)?
- Varanje
 - Jeden od najvećih problema umreženih igara – igra s puno varanja brzo gubi publiku
 - Protokoli i mehanizmi trebaju imati visoku razinu sigurnosti
- Sve ove faktore potrebno uzeti u obzir prilikom izbora sučelja za razvoj umreženih višekorisničkih igara

Kako odabratи sučelje (iz 2020)?

	Stability/ support	Ease-of- use	Perfor- mance	Scalability	Feature breadth	Cost*	Customer: Expand ↗ recommend for ↘
MLAPI	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	Free	Most client-server games for up to ~64 players that want a stable breadth of mid-level features
DarkRift 2	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	\$100 for source	Games with high perf/scale requirements that want to build on a stable LL layer
Photon PUN	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	\$0.30/PCU	Simple and small (2–8 players) mesh-topology games
Photon Quantum 2.0	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	\$1000/mo + \$0.50/PCU	Games desiring deterministic roll-back, like MOBA games, for up to 32 players
Mirror	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	Free	Stable and proven client-server solution, loved best for its community and ease-of-use

Izvor: Unity blog: <https://blog.unity.com/technology/choosing-the-right-netcode-for-your-game>

Kako odabratи sučelje (iz 2020)?



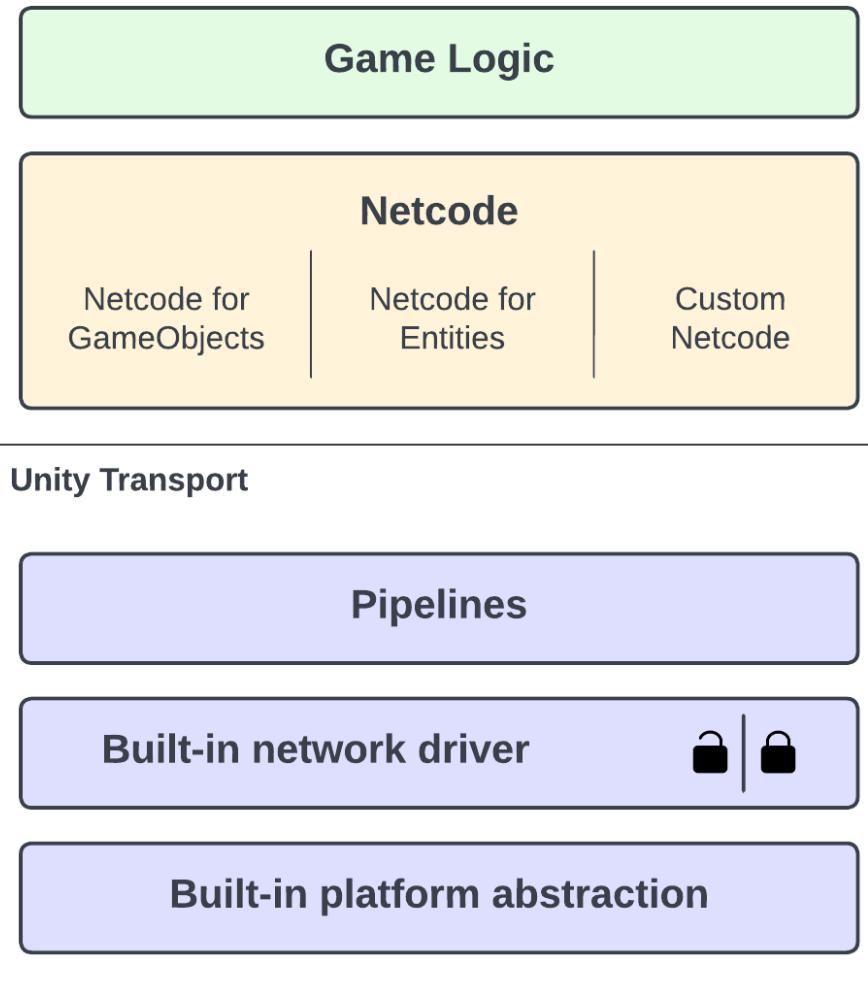
High-level guide to starting to evaluate a netcode solution

Izvor: Unity blog: <https://blog.unity.com/technology/choosing-the-right-netcode-for-your-game>

Oficijelna sučelja

Unity Transport

- Sučelje niže razine
(<https://docs.unity3d.com/Packages/com.unity.transport@2.0/manual/index.html>)
- Omogućuje visoku razinu optimizacija za specifičnu igru
- Koriste ga sučelja više razine (Netcode for GameObjects/Entities)
- Daje apstrakciju veze iznad UDP priključnica (malo viša razina od priključnica)
- Omogućuje
 - Konfiguriranje
 - Spajanje
 - Slanje podataka
 - Primanje podataka
 - Zatvaranje veze
 - Odspajanje
 - Vremensko odspajanje veze (engl. timeout)
- Opcionalne funkcije temeljene na strujanju (engl. pipelines)
 - Pouzdanost
 - Redoslijed dostave paketa
 - Fragmentaciju
 - ...

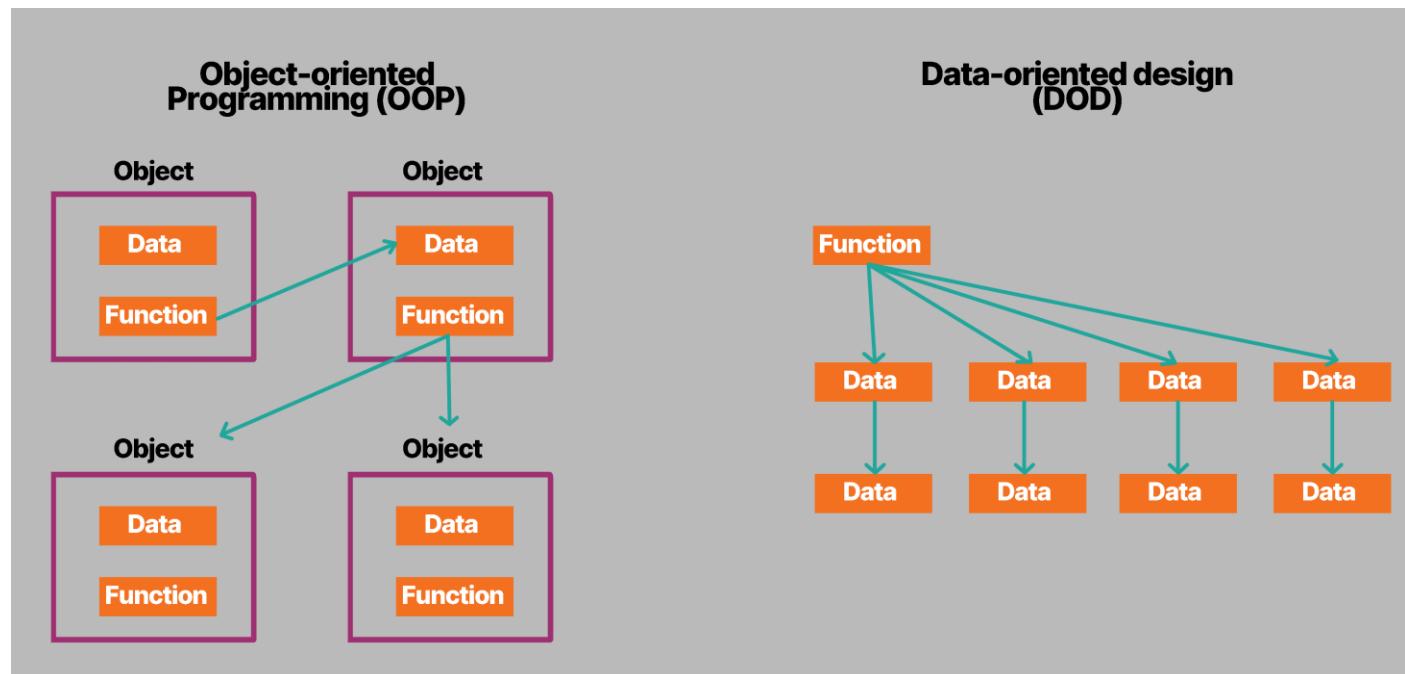


Netcode for Game Objects

- Nova biblioteka razvijena u okviru Unity Technologies (<https://docs-multiplayer.unity3d.com/netcode/current/about>)
- Nalazi se na srednjoj/višoj razini
- Glavne komponente:
 - NetworkObject
 - Komponenta koja omogućuje repliciranje ponašanja određenog objekta (uz NetworkBehaviour)
 - Omogućuje i slanje i primanje RPC-ova
 - Ima jedinstveni identifikator
 - I poslužitelj i klijenti mogu biti vlasnici NetworkObjecta
 - Poslužitelj je zadužen za kreiranje i uništavanje NetworkObjecta
 - NetworkBehaviour
 - Komponenta koja koristi RPC-ove i NetworkVariable kako bi sinkronizirala stanje putem mreže
 - Mora biti prisutna barem jedna komponenta ovog tipa uz NetworkObject kako bi se objekt sinkronizirao putem mreže
 - Apstraktna klasa koja nasljeđuje MonoBehavior i primarno se koristi za implementaciju jedinstvenog mrežnog ponašanja ili logike igre
 - NetworkVariable
 - Varijabla koja se mora definirati unutar NetworkObjecta te kojoj se vrijednost sinkronizira putem mreže
 - Sinkronizira se prilikom inicijalizacije objekta
 - NetworkManager
 - Sadrži sve postavke koje su povezane s umrežavanjem projekta
 - Sadrži i NetworkSceneManager koji omogućuje tranziciju između scena
 - NetworkTransform – omogućuje sinkronizaciju pozicije, rotacije i skale objekta tijekom igre (inicijalno se samo sinkroniziraju kada se kreira NetworkObject)
 - NetworkAnimator – omogućuje sinkronizaciju animacija tijekom umreženog igranja
- **Ako radite novu igru za laboratoriј preporuka je probati Netcode for Game Objects**
- Vodič za kreiranje jednostavne igre: https://www.youtube.com/watch?v=3yuBOB3VrCk&ab_channel=CodeMonkey

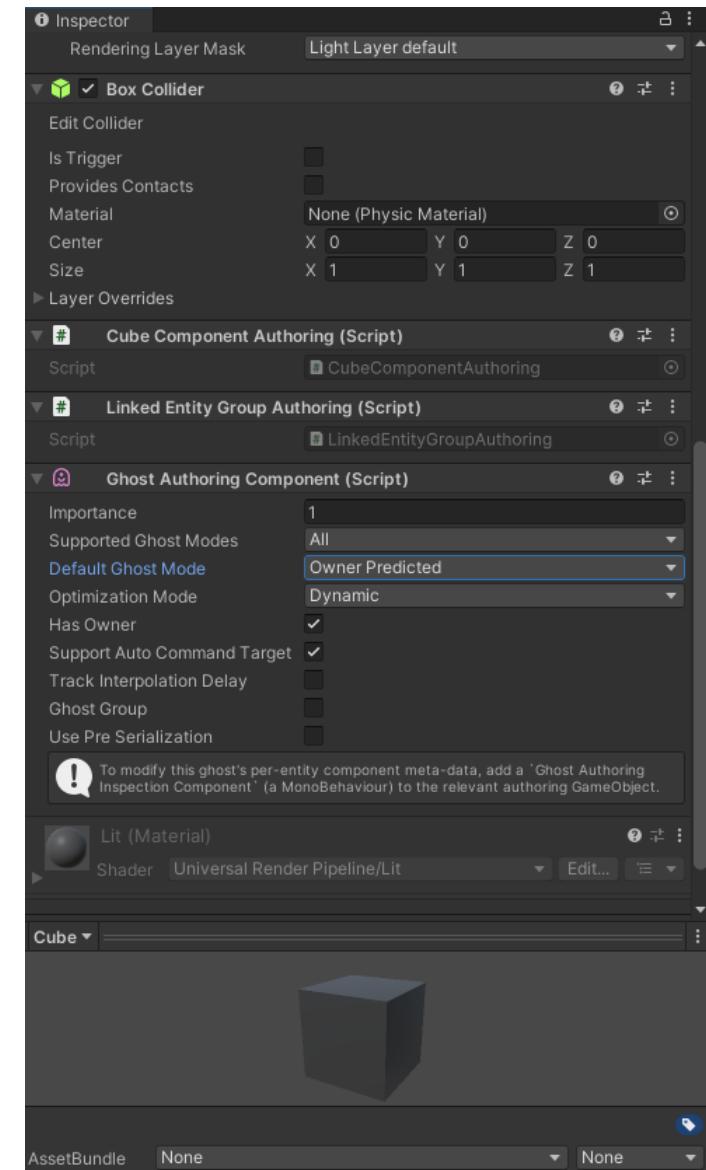
Netcode for Entities

- Paket u razvoju koji je namijenjen igrama temeljenim na Entity Component System (ECS)
<https://docs.unity3d.com/Packages/com.unity.netcode@1.0/manual/index.html>
- ECS je podatkovno orijentirani radni okvir za izradu igara koji omogućuje kreiranje masovnih nivoa, broja neprijatelja i slično koristeći optimizacije na razini organizacije podataka, poslova itd.
- ECS je dio Data Oriented Technology Stack (DOTS) koji je skup tehnologija koji se temelji na optimizaciji rada s podacima
- U okviru ECS-a objekti (GameObjects) su podijeljeni u podatke (variable) te komponente (funkcije)
- Umjesto objekata postavljeni su entiteti koji su automatski optimizirani kako bi ih moglo u sceni biti što više



Netcode for Entities

- Razdvaja logiku poslužitelja i klijenta
- Duh (engl. ghost) je objekt koji simulira poslužitelj, a koji se sinkronizira putem mreže sa svim klijentima
- Klijent prezentira stanje duha, ali ne može ga mijenjati (poslužitelj je vlasnik)
- Kreiranje (engl. spawning) duhova definira poslužitelj, a klijent ako dobije novi ID duha podrazumijeva da je kreiran novi duh
- Ključne komponente
 - GhostComponent – definira entitet kao duha
 - Ghost Authoring Component
 - Komponenta koja definira na koji način se sinkronizira klijentski duh putem mreže
 - Opcije sinkronizacije su: interpolacija, predikcija, oboje
 - Modovi optimizacije su: dinamički (optimizacija veličine kad se mijenja stanje) i statički (nema optimizacije veličine već se samo ne šalju osvježenja kad se ne mijenjaju)
- U pozadini koristi Unity Transport paket te se svaka veza sprema kao entitet
- Entitet u kojem je veza se uništava kada je veza zatvorena
- Sustav ne spaja automatski klijenta i poslužitelja već samo stvara klijentski i poslužiteljski svijet, a sam razvijatelj mora definirati kako se otvaraju kanali komunikacije
- Samo kao informacija, DOTS kao koncept je još u vrlo ranoj razvojnoj fazi, a višekorisnička podrška za isti također



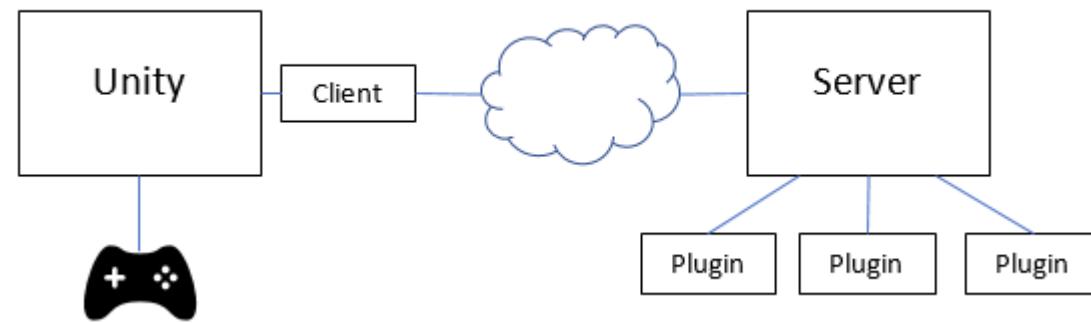
Sučelja trećih strana

Sučelja niske razine

- Networking and Serialization Tools (TNet 3)
<https://assetstore.unity.com/packages/tools/network/networking-and-serialization-tools-tnet-3-56798#description>
 - Cijena od oko 50 EUR
 - Pisan u C# te je dostupan cjelokupni izvorni kod
 - Nema ograničenja na broj korisnika preko 300 korisnika testirano
 - Složena implementacija
- Forge Networking Remastered <https://assetstore.unity.com/packages/tools/network/forge-networking-remastered-38344>
 - Besplatan
 - Pisan u C# te je dostupan cjelokupni izvorni kod
 - Temeljen na C# priključnicama
 - Fokusirano na igre u stvarnom vremenu
 - Podržava komunikaciju između različitih platformi (primjerice iOS i Android)
 - Podržava UDP i RUDP
 - Nije osvježavan posljednjih godinu dana

DarkRift Networking 2

- Dark Rift Networking je sučelje niske/srednje razine
- Dizajniran za različite žanrove igara
- Sustav otvorenog koda
- Koristi višedretvenost
- Na strani servera koristi različite module (engl. plugin) koji se razvijaju kao .NET biblioteke te postavljaju kao DLL-ovi u odgovarajući direktorij
- Besplatna verzija podržava neograničen broj istovremenih igrača, dok plaćena dopušta puni pristup cijelom izvornom kôdu



Mirror networking



- Mirror Networking je mrežno sučelje visoke razine <https://mirror-networking.com/>
- Osmisljeno je tako da je jednostavno za korištenje i temeljeno je na zastarjelom UNet-u
- Poslužitelji u Mirror Networking arhitekturi imaju puni autoritet nad informacijama, te ga čine odličnom opcijom za igre u kojima bi igrači mogli probati varati
- Besplatno za korištenje i otvorenog kôda, što donosi mnoge mogućnosti kod implementacije igre
- Mirror Networking ne nudi poslužitelje na koje se klijenti mogu slobodno spojiti, već programeri sami trebaju postaviti vlastiti poslužitelj i odrediti hoće li se raditi o posvećenom poslužitelju ili o modelu u kojem je klijent poslužitelj
- Mirror se može koristiti za izradu igara manjih opsega (nezavisne igre), ali i velikih, poput velikih mrežnih igara u kojima zajedno igra veliki broj igrača (engl. Massive Multiplayer Online, skr. MMO)
- Zbog toga što je nastao iz UNet tehnologije, sličan mu je i zbog toga ga ljudi koji su prethodno koristili UNet preferira



Fish-Networking (Fish-Net)

- Sučelje više razine <https://fish-networking.gitbook.io/>
- Može koristiti više sučelja niže razine
 - Tugboat (LiteNetLib)
 - FishySteamworks (Steamworks.Net)
 - FishyFacepunch (Facepunch for Steam)
 - Bayou (WebGL)
 - Epic Online Services [EOS]
 - Yak (For offline gameplay)
 - FishyUTP (Unity Transport)
- Relativno novo rješenje (objavljeno krajem 2021. godine)
- Besplatno sučelje otvorenog kôda
- Jednostavno je za korištenje i dolazi kao biblioteka za Unity
- Slično kao Mirror, Fish-Net podržava puni autoritet poslužitelja, ali istovremeno daje mogućnost načina rada klijenta kao poslužitelja
- Kada se radi o skalabilnosti, ovo sučelje nadmašuje Mirror Networking jer za visoke brojke umreženih objekata, vrijeme odziva Mirror poslužitelja raste, dok Fish-Net ima gotovo statične vrijednosti
- Dodatno podržava i kompenzaciju kašnjenja (engl. lag compensation), dinamičku veličinu paketa i AOI sustav (eng. area of interest)
- Glavne komponente
 - NetworkObject – komponenta za sinkronizaciju između klijenta i poslužitelja koja se automatski postavlja na objekt kada se na objekt doda skripta koja nasljeđuje NetworkBehavior klasu
 - NetworkBehaviour – apstraktna klasa koju nasljeđuju druge klase i implementiraju ponašanje objekata
 - NetworkAnimator – omogućuje sinkronizaciju animacija putem mreže
 - NetworkTransform – omogućuje sinkronizaciju pozicije, rotacije i skale putem mreže
 - PredictedObject – bilo kakav objekt na koji utjecaj može imati predikcija na strani klijenta (sam objekt igrača ili lopte koju igrač može udariti)

Photon

- Photon nudi nekoliko osnovnih Proizvoda za izradu višekorisničkih umreženih aplikacija:
 - Quantum
 - Fusion
- Sakrivaju se funkcije mreže od korisnika
- Photon je platforma sadrži mnoge funkcionalnosti potrebne pri izradi višekorisničkih umreženih igara.
- Neke od poznatijih igara koje su razvijene pomoću ovog alata su Shadowgun Legends, LAMO, Thea 2: The Shattering, Pixel Gun 3D i Guns of Icarus: Online.
- Paket Photon Unity Networking je biblioteka namijenjena jednostavnom razvoju višekorisničkih videoigara u pogonskom sustavu Unity.



Photon Fusion i Quantum

- Fusion je softverski razvojni okvir (engl. Software Development Kit skr. SDK) koji je namijenjen profesionalnim razvijateljima u pogonskom sustavu Unity i nudi vrlo napredne opcije poput predikcije na strani klijenta, interpolacije snimaka, kompenzacije kašnjenja, razliku snimaka i slično
 - Fusion je najnoviji razvojni okvir koji objedinjuje prethodno razvijene Bolt i PUN sustave te ih i podržava.
 - Sustavi razvijeni u PUN-u i Boltu nastavit će raditi u okviru Fusion sustava
- Quantum je deterministički višekorisnički pogonski sustav za višekorisničke videoigre namijenjen za razvoj u Unity pogonskom sustavu te ga najčešće koriste strategijske igre
- Realtime je mrežni pogonski sustav namijenjen za umrežavanje različitih platformi (primjerice za mobilne i PC platforme).
 - Photon Chat i Voice pružaju podršku za pismenu, odnosno govornu komunikaciju između korisnika
 - Photon Server je još jedan dodatni paket koji nudi funkcionalnosti za uspostavu aplikacije na vlastitom poslužitelju uz korištenje od nekih spomenutih osnovnih paketa
 - Zbog jednostavnosti implementacije detaljnije obrađujemo Photon Unity Networking verzije 2 (PUN 2)



Photon Unity Networking (PUN) 2

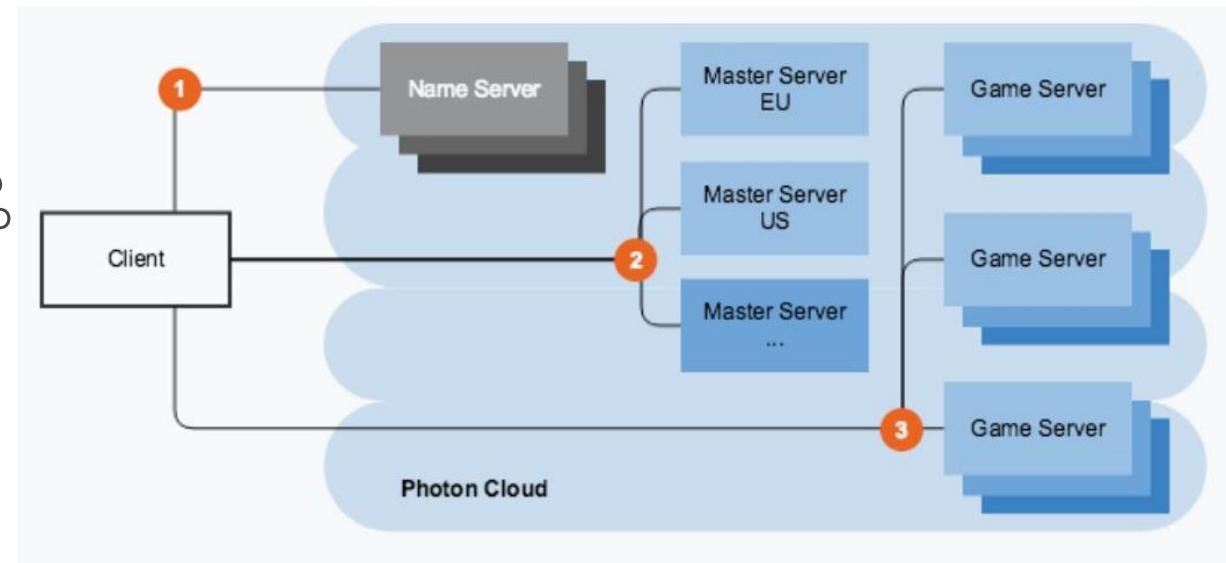
Photon PUN2

- PUN 2 je paket koji pruža podršku pri izgradnji višekorisničkih umreženih igara u pokretačkom sustavu Unity.
- PUN 2 je integriran i usko vezan uz Unity, on nadograđuje Unity Editor s novim funkcionalnostima poput podrške zainstanciranje mrežnih objekata.
- Brza i (opcionalno) pouzdana komunikacija je ostvarena kroz Photon poslužitelje pa igrači nemaju potrebe da se povezuju vezama jedan-na-jedan, tj. koristi se klijent-poslužitelj arhitektura.
- PUN 2 je nudio dvije opcije:
 - PUN 2 Free je besplatna inačica ovog alata i podržava maksimalno 20 konkurentnih korisnika ako igru pokrećemo na njihovim poslužiteljima
 - PUN 2 Plus je plaćena opcija koja je koštala 85 EUR te omogućuje 100 konkurentnih korisnika ako igru pokrećemo na njihovim poslužiteljima



PUN 2 arhitektura

- Igrači se inicijalno spajaju na Name Server od kojega dobivaju listu raspoloživih poslužitelja
- Nakon toga šalju testni paket svakome iz liste da odrede vrijeme kašnjenja s kraja na kraj
- Korisnik se zatim spaja na poslužitelj koji ima najmanje kašnjenje
- Photon također nudi i opciju da se igrači uvijek spajaju samo na unaprijed definirane poslužitelje što je korisno ako znamo da će se naša videoigra koristiti samo u nekim regijama
- Svaka regija je potpuno odvojena od drugih i sastoji se od Master Servera i Game Servera
- Na Master Server sa spaja nakon Name Servera i tamo se može vidjeti listu postojećih soba te kreirati nova.
- Na Game Serveru se igrač nalazi kada se pridruži nekoj od soba
- MasterClient je poseban tip igrača i može imati više funkcionalnosti od običnog
 - Primjerice, može se implementirati da jedino MasterClient smije pokrenuti igru za igrače koji se nalaze s njim u sobi
 - Igrač koji je kreirao sobu je i ujedno MasterClient



PUN 2 arhitektura – raspored poslužitelja

- Photon poslužitelji su rasprostranjeni po cijelom svijetu kako bi se ostvarilo ujednačavanje opterećenja i smanjilo kašnjenje u mreži.
- Postoji 13 regija na koje su podijeljeni poslužitelji
- Igre razvijene u Hrvatskoj bit će najčešće dodijeljene na poslužitelj u Amsterdamu s oznakom „eu“

Regija	Zemlja	Token
Azija	Singapur	Asia
Australija	Melbourne	Au
Istočna Kanada	Montreal	Cae
Kina	Šangaj	cn
Europa	Amsterdam	eu
Indija	Chennai	in
Japan	Tokio	jp
Rusija	Moskva	ru
Istočna Rusija	Khabarovsk	rue
Južna Amerika	Sao Paolo	sa
Južna Koreja	Seul	kr
Istočne SAD	Washington	us
Zapadne SAD	San Jose	Usw

PUN 2 sobe i predvorja

- Photon je napravljen za igre koje se temelje na sobama, odnosno poslužiteljima za ograničeni broj igrača (primjerice do 10 igrača)
- Igrači u pojedinoj sobi su odvojeni od igrača u drugoj sobi
- Igrači mogu komunicirati samo dok su u istoj sobi
- Igrači se u sobe mogu spajati na razne načine:
 - Random Matchmaking – slučajno raspoređivanje u sobe u kojima se igrač na slučajan način postavlja u jednu od postojećih soba koja zadovoljava uvjete koje je igrač postavio
 - Spajanje u sobu putem njenog identifikatora kada se igrač želi spojiti u specifičnu sobu
 - Odabir sobe iz liste koju dostavlja Master Server
- Kada se koriste predvorja (engl. Lobby) u njima se igrači spajaju prije nego što odaberu sobu
- U predvorjima se prikazuju sve postojeće sobe te se predvorja generiraju na Master Serveru
- U predvorjima se ne može komunicirati između igrača

PUN 2 struktura

- PUN 2 struktura se sastoji od sljedećih skupova funkcionalnosti grupiranih u imenske prostore (engl. namespace):
 - Photon.Pun – najviša razina koja nudi već implementirane funkcionalnosti specifične za rad u Unity alatu poput mrežnih objekata i poziva udaljenih procedura (engl. Remote Procedure Call skr. RPC),
 - Photon.Realtime – srednja razina koja sadrži svu logiku potrebnu za rad s Photon poslužiteljima i
 - Najniža razina koja se sastoji od .dll datoteka i sadrži stvari vezane uz serijalizaciju/deserijalizaciju, protokole i sl.

Photon.Pun

- Osnova za mnoge igre koje koriste Photon je upravo Photon.Pun.
- Neke od značajki tj. funkcionalnosti koje podržava su:
 - Pozivi udaljenih procedura
 - Izvanmrežni (engl. offline) način rada
 - **Komponenta za sinkronizaciju mrežnih objekata (najvažnija komponenta – PhotonView).**
- Photon.Pun omogućava korištenje sučelja MonoBehaviourPunCallbacks, koje omogućava korištenje mnoštva metoda za spajanje korisnika u sobe i slično.
- Omogućava korištenje glavne klase koja omogućava korištenje Photon paketa – **PhotonNetwork**
- Koristeći PhotonNetwork klasu korisnik dobiva pristup mnoštvu metoda

Metoda	Opis (kada se poziva)
bool ConnectUsingSettings()	Povezivanje lokalnog igrača s Photon poslužiteljem koristeći unaprijed postavljene postavke.
void Disconnect()	Odspaja lokalnog igrača od Photon poslužitelja.
bool JoinLobby()	Za pridruživanje lokalnog igrača u predsoblje.
bool CreateRoom(string roomName, RoomOptions roomOptions=null)	Za stvaranje i pridruživanje lokalnog igrača u novu sobu.
void LoadLevel(int levelNumber)	Za učitavanje nove scene.
Svojstvo	Opis (kada se poziva)
ServerSettings PhotonServerSettings	Sadrži serijalizirane informacije o spajanju na poslužitelja
Player LocalPlayer	Sadrži informacije o lokalnom igraču.
bool isMasterClient	Informacija o tome je li lokalni igrač glavni klijent u sobi.

Photon.Realtime

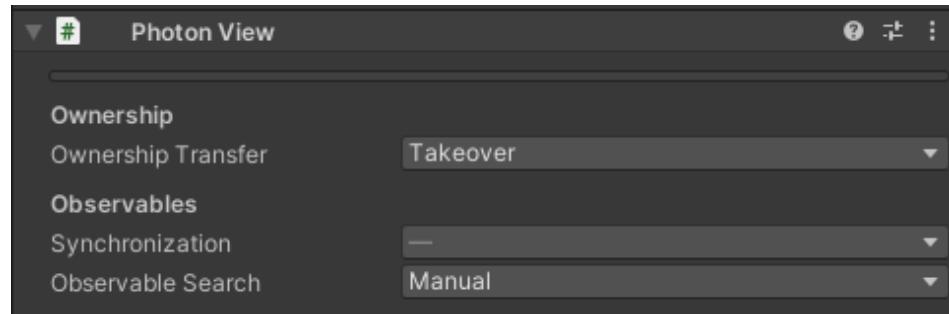
- Photon.Realtime je neovisan o Unityju te mu dodaje mnoge bitne značajke (na niskoj razini), čije je korištenje olakšano zbog Photon.Puna
- Photon.Realtime rješava probleme vezane uz povezivanje, komunikacije i skalabilnosti, te predstavlja sučelje koje definira način komunikacije između klijenata i poslužitelja.
- Osim pasivnog rješavanja komunikacije između čvorova, pomoću Photon.Realtimea moguće je obavljati naredbe poput **Connect**, **RaiseEvent**, ali i **dohvaćati informacije o sobama i igračima**.
- Primjeri
 - PhotonNetwork.PlayerList; je naredba koja vraća **kopiju liste trenutnih igrača u nekoj sobi**. Lista se automatski ažurira prilikom dolaska i odlaska igrača iz sobe.
 - Player je klasa koji sadržava sve bitne **informacije o korisniku**. Neke od bitnijih su identifikator, nadimak, te prilagođena svojstva (engl. custom properties) koja mogu biti dodana za vrijeme razvijanja igre.
 - RoomInfo je klasa koja sadrži **pojednostavljene informacije o sobama** i služi samo za vraćanje informacije, ali ne i postavljanje ili mijenjanje. To su informacije potrebne za prikazivanje sobe te pridruživanje sobi poput imena sobe, broja igrača u sobi i informacije o tome može li se neki vanjski igrač pridružiti sobi.

Serijalizacija u PUN2

- PUN podržava TCP i UDP transportne protokole
- Photon nudi i opciju podešavanja vrijednosti koja označava koliko svaki objekt puta u sekundi šalje poruke osvježavanja svoga stanja (engl. Send Rate) s inicijalnom vrijednošću od 10
- UDP je podrazumijevani protokol koji se koristi u kombinaciji s algoritmom osiguravanja pouzdanosti
- Photon koristi optimizirani binarni protokol za komuniciranje koji je kompaktan i jednostavan za parsiranje
- Photon automatski serijalizira primitivne tipove podatak poput: byte, bool, short, int, long, float i double kao i String i par osnovnih struktura podataka.
- Za serijalizaciju vlastitih razreda potrebno i napisati metode za njihovu serijalizaciju i deserijalizaciju tako što ćemo implementirati sučelje IPunObservable i nadjačati metodu OnSerializePhotonView.

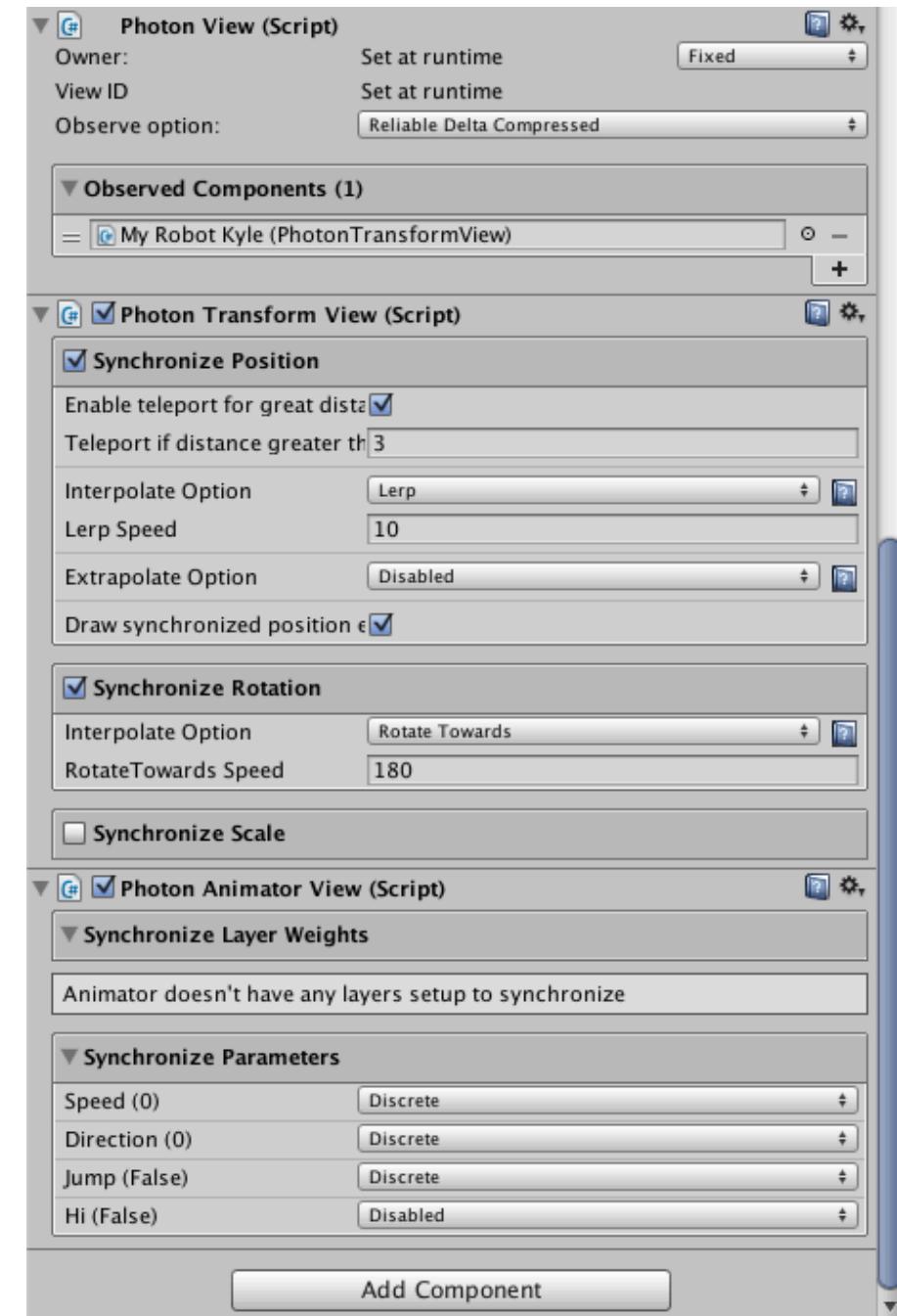
Photon.View

- Ključna komponenta s kojom se jednostavno vrši sinkronizacija stanja pojedinog objekta
- Obični objekti unutar igre postaju mrežni objekti ako se na njih zakači PhotonView komponenta
- PhotonView je zadužen za identificiranje objekta unutar mreže i sinkroniziranje njegovog stanja odnosno za informacije koje se često mijenjaju (poput pozicije i rotacije avatara)
- Na samoj PhotonView komponenti se mogu definirati 4 opcije za sinkronizaciju podataka:
 - Off – Sinkronizacija se ne provodi, ništa se ne šalje niti prima,
 - Reliable Delta Compressed – Pouzdana dostava poruka gdje se koristi interni optimizacijski mehanizam koji šalje null vrijednost ako se stanje objekta nije promijenilo,
 - Unreliable – Podaci pristižu ispravnim redoslijedom ali gubitak paketa je moguć i
 - Unreliable OnChange – Podaci pristižu ispravnim redoslijedom ali gubitak paketa je moguć. Ako je sljedeća poruka ista kao i prethodna, Photon pauzira slanje podataka dok se stanje objekta opet ne promjeni
- U Observed polje se postavljaju komponente koje se promatraju odnosno sinkroniziraju, dakle kroz to polje definiramo koji aspekti pojedinog objekta se prenose preko mreže
- OnPhotonSerializeView – poziva se nekoliko puta po sekundi kako bi skripte mogle čitati i pisati podatke koji se sinkroniziraju
 - Koristi se kako bi se definirali specifični podaci koji se sinkroniziraju kroz Photon.View komponentu
 - Skripta koja sadrži ovu funkciju mora se dodati u popis promatranih objekata
 - Samo se poziva kada je dodana u poziv promatranih objekata



Promatrane komponente

- PhotonTransformView omogućava sinkronizaciju pozicije, skale i rotacije objekta koji se kreće po sceni
 - Moguće odabrati koji će se od parametara sinkronizirati
 - Opcije sinkronizacije pozicije
 - Teleport if distance greater than – jednostavno promijeni položaj objekta instantno
 - Interpolation Option
 - Postavljanje pozicije svaki okvir
 - Lerp
 - Fiksna brzina
 - Extrapolation option – predviđanje pozicije ako ne dođu osvježenja
- PhotonAnimatorView koja omogućava sinkronizaciju Mecanim animacija putem mreže u Photonu
 - Moguće je sinkronizirati parametre animacije temeljem vanjskih parametara
 - Postoje dvije opcije sinkronizacije:
 - Discrete – sinkronizacija 10 puta po sekundi
 - Continuous – svaki video okvir se izvršava sinkronizacija (bolje animacije, ali veće opterećenje mreže)

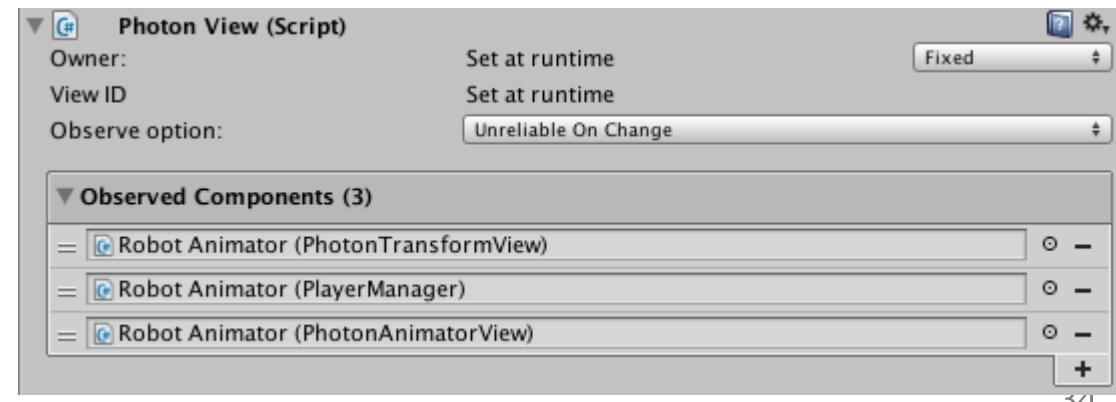


Ručna sinkronizacija varijabli

- Primjer igrača koji puca iz puške
 - Efekt pucanja pokazuje se na strani lokalnog igrača
 - Potrebno ga prenijeti putem mreže
- Ručno se varijable sinkroniziraju kroz modifikaciju metode OnPhotonSerializeView iz sučelja IPunObservable
- Podaci se pišu u metodi samo kada je photoView.IsMine = True, inače se čitaju
- Potom se dodaje u popis promatranih komponenti klase PlayerManager
- Kako bismo dodali varijablu zdravlja da se sinkronizira u postojeću skriptu?

```
public class PlayerManager : MonoBehaviourPunCallbacks, IPunObservable
{
    #region IPunObservable implementation

    public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
    {
        //
        if (stream.IsWriting)
        {
            // Vlasnik igrača šalje informaciju o pucanju
            stream.SendNext(IsFiring);
        }
        else
        {
            // Neki drugi igrač iz mreže puca
            this.IsFiring = (bool)stream.ReceiveNext();
        }
    } #endregion
}
```



Poziv udaljene procedure

- Događaji se mogu prenijeti mrežom pomoću poziva udaljenih procedura (engl. Remote Procedure Call - RPC)
- RPC predstavlja pozivanje metode u lokalnoj kopiji od nekog drugog igrača koji se nalazi u istoj sobi kao i pozivatelj
- RPC metoda se mora označiti atributom [PunRPC] nakon čega ju drugi igrači mogu pozvati pomoću PhotonView komponente

```
void MethodA()
{
    photonView.RPC("ChatMessage", RpcTarget.All, "tekst");
}
[PunRPC]
void ChatMessage(string text, PhotonMessageInfo info)
{
    Debug.LogFormat("Info: {0} {1} {2}", info.Sender, info.photonView,
    info.SentServerTime);
}
```

Poziv udaljene procedure

- Prilikom pozivanja RPC metode pod opcijama možemo definirati tko će sve pozvati metodu kroz enumeraciju RpcTarget koja ima sljedeće opcije:
 - All – Svi klijenti izvršavaju metodu
 - Ako koristimo ovu opciju onda klijent koji je pozvao RPC prvo lokalno izvršava tu metodu pa tek ju onda poziva na svim ostalim klijentima
 - Others – Svi klijenti osim pozivatelja izvršavaju metodu,
 - ViaServer – Poruke koje se šalju svima mogu se slati s ovom opcijom kako bi postigli da svi u istom trenutku pozovu neku metodu.
 - Korištenjem AllViaServer opcije klijent prvo signalizira poslužitelju da javi svima da izvrše tu metodu pa tako i on čeka na poslužitelj da mu javi kada smije izvršiti metodu,
 - Buffered – Osigurava da klijenti koji naknadno uđu igru također pozovu metodu
 - MasterClient – Metodu izvršava samo igrač koji je označen kao MasterClient u pojedinoj sobi.

Podizanje događaja

- Za aktivnosti u videoigri koje se mijenjaju često i skoro svaki vremenski okvir (primjerice pozicija i orijentacija igrača) te se moraju dostaviti do poslužitelja i drugih igrača te za njih koristimo PhotonView komponentu
- Za promjenu koje se ne događaju tako često i kako bi se mreža što manje opteretila periodički se prenose korištenjem događaja.
- Primjerice igrač je pokupio dodatnu municiju za svoje oružje.
- Događaji se stvore i emitiraju (engl. broadcast) putem mreže, a ostali objekti se mogu na njih pretplatiti (engl. subscribe) koristeći svoje metode.
- Tako, događaj stvoren na jednom mjestu u kôdu može biti primijećen u drugom mjestu u kôdu i na drugom sudioniku u igri
- Slično se može dobiti korištenjem RPC, ali one su vezane za objekte koji na sebi imaju Photon View komponentu, dok događaji nisu

Podizanje događaja

- Sučelje
MonoBehaviourPunCallbacks daje mnoštvo definiranih događaja te metoda koje hvataju dane događaje te se koriste razvoja višekorisničke umrežene igre
- Događaji i metode su većinom vezani za manipulaciju lokalnog korisnika i sobe
- Pomoću ovih metoda, moguće je reagirati na razne situacije i prilagoditi lokalnog korisnika tako da u igri postoji što manje mogućih pogrešaka

Metoda	Opis (kada se poziva)
<code>void OnCreateRoomFailed(shorta returnCode, string message)</code>	Kada dođe do pogreške u stvaranju sobe.
<code>void OnLeftRoom()</code>	Kada lokalni igrač napusti sobu.
<code>Void OnRoomListUpdate(List<RoomInfor m> roomInfo)</code>	Kada se ažurira lista igrača u trenutnoj sobi
<code>void OnPlayerEnteredRoom(Player player)</code>	Kada drugi igrač uđe u trenutnu sobu.
<code>Void OnMasterClientSwitched(Player player)</code>	Kada se promijeni vlasnik sobe.
<code>void OnJoinedRoom()</code>	Kada se lokalni igrač pridruži sobi.
<code>void OnJoinedLobby()</code>	Kada se lokalni igrač pridruži predsoblu.
<code>void OnConnectedToMaster()</code>	Kada se lokalni igrač spoji na Photon poslužitelja.

Podizanje događaja

- Moguće je definirati svoje događaje i metode koje ih hvataju
- Događaji se onda međusobno razlikuju po identifikatorima
- Za identifikator se koristi tip byte stoga bi u teoriji trebalo značiti da na raspolaganju imamo 256 takvih identifikatora
- Photon već koristiti neke identifikatore za vlastite događaje te su programeru na raspolaganju su ostavljeni identifikatori od 1 do 199
- Photon.Realtime dolazi sa sučeljem, IOnEventCallback koje pokriva rad s događajima, koji mogu biti stvoreni na poslužitelju ili klijentima
- Bilo koja klasa koja nasljeđuje ovo sučelje mora implementirati jedinu metodu koju sučelje void OnEvent(EventData photonevent) koje zapravo „hvata“ definirani događaj i na temelju toga aktivira dodatnu programsku logiku.

Metoda	Opis (kada se poziva)
bool RaiseEvent(byte eventCode, object eventContent, RaiseEventOptions raiseEventOptions, SendOptions sendOptions)	Kada je potrebno stvoriti novi događaj.

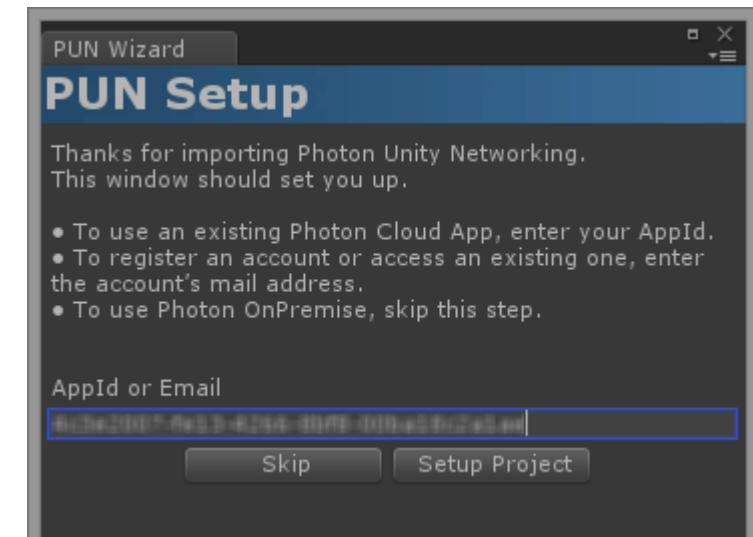
Slanje događaja i RPC metoda

- RaiseEvent i pozivi RPC metoda se ne izvršavaju odmah
- Photon pokušava grupirati sve poruke s porukama osvježavanja stanja kako bi se slalo manje paketa i time manje opteretila mrežna veza
- Navedeni scenarij se može izbjegići tako da se pozove metoda PhotonNetwork.SendAllOutgoingCommands() koja odmah šalje takve vrste događaja.
- Kako Photon ima poslužitelje koji se nalaze u javnom adresnom prostoru protokola IP sami poslužitelji omogućavaju prolazak NAT-ova
- Naime, svaki korisnik se prvo spaja na Photon master poslužitelj koji potom omogućava korisnicima da se spoje na game poslužitelj putem kojega razmjenjuju informacije



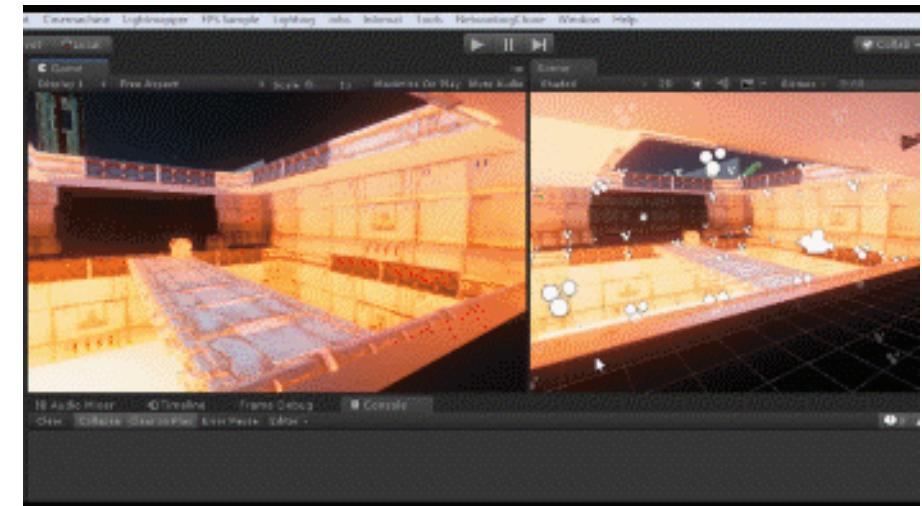
Postupak instalacije PUN2

- Iz trgovine sredstvima potrebno je preuzeti PUN 2 sredstvo te ga pohraniti i instalirati
- Potrebno se registrirati kako bi se dobio identifikator aplikacije AppID
- PhotonCloud omogućuje jednostavno postavljanje višekorisničke igre s ograničenim brojem korisnika (20 za besplatne račune)
- PUN 2 interno odradjuje sve što se tiče PhotonClouda odnosno spajanje na Name Server, spajanje na Master Server i kreiranje soba odnosno Game Servera
- Vodič za kreiranje igre u PUN 2:
<https://doc.photonengine.com/en-us/pun/current/demos-and-tutorials/pun-basics-tutorial/intro>



Dodatni alati

- Unity Project Cloner
<https://github.com/hwaet/UnityProjectCloner>
- Alat koji omogućuje jednostavno kloniranje projekta u više instanci kako biste mogli lakše testirati
- Sve instance se automatski sinkroniziraju
- Svaka od instanci se tretira kao zaseban korisnik
- Ne mora se svaki put izgraditi igra te pokretati ta verzija kako biste testirali
- Olakšava proces testiranja igre



Zadatak za iduće predavanje

- Pročitati sljedeći rad vezan za utjecaj kašnjenja na iskustvenu kvalitetu igara (ostalo isto od prethodnog predavanja zbog zamijene redoslijeda predavanja):
- <https://web.cs.wpi.edu/~claypool/papers/csgo-net-21/paper.pdf>