Interaktivna simulacija dinamike jedrenja primjenom alata Unity

Seminarski rad iz kolegija “Interaktivni simulacijski sustavi”

Valentino Vukelić, Luka Braut, Paula Klobučarević  
10. siječnja 2022.

Djelovođe: izv. prof. dr. sc. Siniša Popović

***Sažetak* – U ovom seminarskom radu prikazana je izrada interaktivne simulacije dinamike jedrenja u programu Unity. U simulaciji je implementiran pojednostavljeni fizikalni model broda kojim se upravlja pomoću kormila te zatezanjem jedra. Na kretanje plovila najviše utječe smjer i brzina vjetra. Također su dodane vizualne komponente koje vjerno prikazuju stvarno okruženje broda. Krajnji cilj projekta je bio omogućiti korisniku što stvarniji prikaz kretanja i upravljanja brodom.**

# Uvod

Jedrenjem smatramo upravljanje plovilom na vodi koje se isključivo pokreće snagom vjetra. Ova disciplina u povijesti je bila vrlo važna u području ribarstva, a u današnje doba se pretvorila i u sport, te rekreativno uživanje. Upravljanje plovilom može biti vrlo komplicirano jer svaki pokret plovila najviše ovisi o vjetru, odnosno o njegovoj brzini i smjeru koji nekada znaju biti nepredvidljivi i vrlo promjenjivi. Zato je vrlo važno steći iskustvo kako u stvarnom okruženju tako i u virtualnom, odnosno simulacijskom. U tom aspektu nam mogu pomoći interaktivni simulacijski sustavi kako bi početnike mogli što bolje pripremiti za upravljanje u stvarnom sustavu. U ovom radu prikazana je jedna takva interaktivna simulacija koja nam prikazuje pojednostavljeni model jedrenja. Kod izrade simulacijskog sučelja vrlo je bitno implementirati što vjerniji fizikalni model sustava kako se simulacija ne bi previše razlikovala od stvarnog okruženja. U ovo ubrajamo gibanje vjetra i mora te najvažnije model jedrilice. Fizikalni model broda je vrlo složen jer ovisi o puno faktora, tako da se u nekim jednostavnijim simulacijama koje se neće koristiti u svrhu treniranja upravljanja plovilom mogu koristiti i aproksimacije gibanja broda pomoću krivulja. Takav pristup implementiran je i u ovom radu. Vrlo važan dio u simulacijskom okruženju je i mogućnost promjene uvjeta u kojima se izvodi simulacija. U našem primjeru to bi bila brzina vjetra, a kod neke složenije simulacije moglo bi se uvesti i promjena ostalih vremenskih uvjeta, kao što su veličina valova ili padaline. Upravljanje brodom se postiže na dva načina: zakretanjem kormila i zatezanjem odnosno otpuštanjem jedra. Sve funkcionalnosti napravljene su u programu Unity.

# Uloge pojedinih članova tima

U sveukupnim poslovima na izradi seminarskog rada te pisanju ovog izvješća, članovi tima sudjelovali su na sljedeći način:

* Valentino Vukelić – fizika okretanja jedra, fizika naginjanja broda, fizika plovidbe, kompas i prikaz orijentacije jedra, (4. Implementirani model fizike, 5.2 Model jedrilice, 6. Zaključak)
* Luka Braut – fizika skretanja broda, dodavanje mora, fizika plovidbe, namještanje krivulja, dorađivanje svih detalja fizike, (5.1 More, 5.3 Korisničko sučelje)
* Paula Klobučarević – komponente za ulaz s tipkovnice, prikaz stanja ulaznih komponenti na korisničkom sučelju, (Sažetak, 1. Uvod, 3. Unity game engine)

# Unity game engine

Interaktivnu simulaciju jedrenja implementirali smo u programu Unity[1] koji nam omogućuje jednostavnu izradu 3D simulacija i grafičkog sučelja. Vrlo je popularan u izradi mobilnih i kompjuterskih igrica zato što podržava većinu operacijskih sustava te je vrlo jednostavan za korištenje, pogotovo za početnike. Također se koristi i u raznim drugim industrijama poput filmske ili arhitektonske. Unity ima mogućnost izrade 2D i 3D simulacija. Programiranje simulacije i svih komponenti koje se nalaze u njemu se bazira na C# programskom jeziku.

# Implementirani model fizike

Model smo razdvojili u više dijelova zbog jednostavnosti. Prvo je okretanje jedra ovisno o trenutnoj rotaciji, smjeru i jačini puhanja vjetra. Zatim, naginjanje broda ovisi o trenutnoj rotaciji jedra, smjeru i jačini vjetra, te na kraju akceleracija odnosno brzina broda ovisi o smjeru i jačini vjetra i rotaciji jedra. Neovisno o vjetru, napravljeno je i skretanje broda koje ovisi samo o ulazu s tipkovnice od korisnika i trenutnoj brzini broda.

## Okretanje jedra

Okretanje jedra implementirano je u jednoj komponenti u datoteci *SailRotation.cs*. U *Update* funkciji komponente (Slika 1.), koja se poziva pri svakom iscrtavanju, računa se prvo akceleracija okretanja, te se ta akceleracija množi s vremenskim korakom i zbraja u brzinu okretanja. Zatim se ta brzina opet množi s vremenskim korakom i zbraja s trenutnim okretom jedra. Nakon toga se brzina smanjuje zbog trenja.

A screenshot of a computer

Description automatically generated with medium confidence

Slika 1. *Update* funkcija *SailRotation* komponente

Za izračun akceleracije, prvo je potrebno izračunati smjer i jačinu vjetra. Za vektor smjera vjetra zbraja se pravi smjer vjetra s prividnim koji je uzrokovan kretanjem broda kroz zrak. Zbog boljeg efekta prividan smjer vjetra se zbraja jedino ako vjetar ulazi sa zadnje strane vjetra, odnosno kut između vjetra i jedra je izmedu 90 i 270 (linija 42). Zatim se računa postotak otpora jedra s vektorskim umnoškom vektora vjetra i vektora jedra (linija 45). Zatim se to pomnoži s konstantom *sailAngularAcceleration*, koja je izabrana proizvoljno za najrealističniji efekt, te kvadratom brzine vjetra (linija 47). Na kraju *Update* funkcije poziva se *ApplyTrimming* funkcija koja ograniči maksimalnu rotaciju jedra ovisno o ulazu od korisnika.

## Naginjanje broda

Naginjanje broda implementirano je u jednoj komponenti u datoteci *BoatRoll.cs*. U *Update* funkciji komponente (Slika 2.) računa se prvo sila naginjanja od vjetra, te se od nje oduzme sila ispravljanja broda koja je uzrokovana težinom kobilice (utega na dnu broda). Ta se sila onda pomnoži s vremenskim korakom da se dobije akceleracija i onda se zbraja u brzinu naginjanja. Zatim se ta brzina opet množi s vremenskim korakom i zbraja s trenutnim naginjanjem broda. Nakon toga se brzina smanjuje zbog trenja.

Text

Description automatically generated

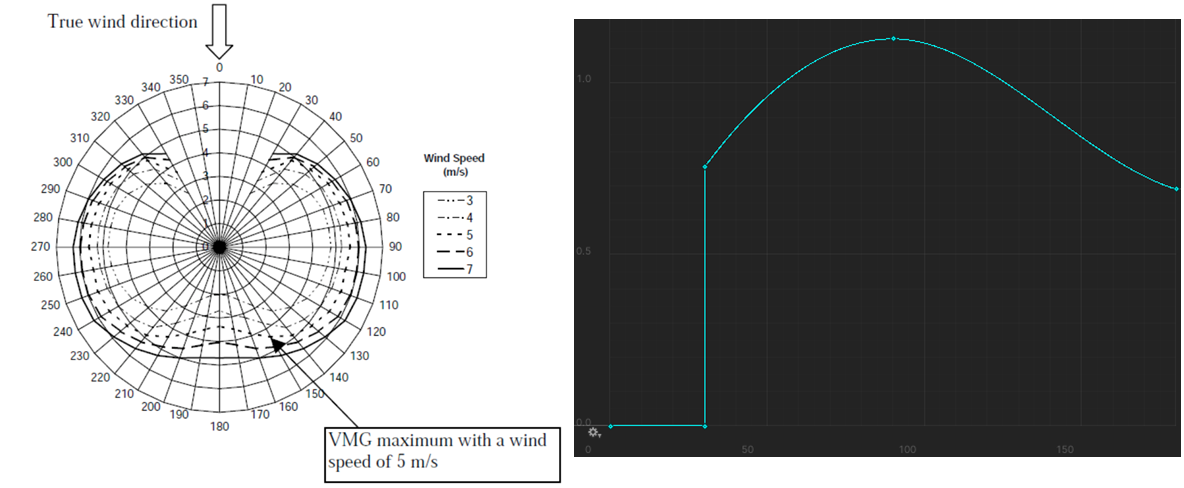
Slika 2. *Update* funkcija *BoatRoll* komponente

Za izračun sile naginjanja potrebno je uzeti u obzir smjer vjetra, smjer broda i smjer jedra. Zatim se sila (*windForce*, linija 35) računa umnoškom tih faktora, brzine vjetra i konstantom *sailDragForce* izabranom proizvoljno. *WindDirectionSign* je smjer naginjanja (lijevo ili desno), *windDirectionDelta* je kut između broda i vjetra, a *dragPercentage* je postotak otpora zraka jedra koji ovisi o smjeru jedra i smjeru vjetra. Zatim se računa sila povratka (*backForce*, linija 37) koja ovisi o trenutnom naginjanju i težini kobilice (proizvoljna konstanta). Te dvije sile se oduzimaju i zbrajaju u brzinu okretanja množeći s vremenskim korakom. Na kraju se kao i prije smanjuje brzina naginjanja zbog trenja (linija 46).

## Pokretanje broda

Za razliku od ostalog, pokretanje broda nije napravljeno koristeći formule sila, jer su formule previše komplicirane i ovise o previše faktora, tako da premašuju opseg ovog projekta. Umjesto toga, pokretanje broda modelirali smo s tri krivulje.

Prvu krivulju (Slika 3.) smo pronašli u radu The Physics of Sailing[5]. Ta krivulja je funkcija smjera vjetra relativna smjeru broda, a vrijednost funkcije je maksimalna brzina koju jedrilica može postići ako ide u tom smjeru u odnosu na vjetar. To je brzina broda ako je jedro savršeno zategnuto (okrenuto). Krivulja nema vrijednosti (odnosno nula je) za kutove manje od 30 stupnjeva jer jedrilica ne može jedriti direktno u vjetar.



Slika 3. Krivulja maksimalne brzine broda ovisne o smjeru vjetra relativne smjeru broda i brzine vjetra [5]

Druga krivulja (Slika 4.) je kut pod kojem bi jedro trebalo biti da bi bila postignuta najveća moguća brzina. Tu krivulju nismo pronašli na internetu nego ju je Luka namjestio po iskustvu jer se on bavi jedrenjem. Apscisa os krivulje je također kut između broda i vjetra (kao i u prvoj krivulji).

A screenshot of a computer

Description automatically generated with medium confidence

Slika 4. Krivulja najboljeg kuta jedra ovisnog o smjeru vjetra relativne smjeru broda

Treća krivulja (Slika 5.) određuje koliko će brod ići sporije od maksimalne brzine (u postotku) ovisno o tome koliko je jedro krivo namješteno. Apscisa os funkcije je greška (u stupnjevima) u kutu jedra u odnosu na najbolji kut, a vrijednost funkcije je postotak maksimalne brzine koju brod može postići u toj konfiguraciji. Tu krivulju je također Luka odredio iz iskustva.

Graphical user interface

Description automatically generated

Slika 5. Krivulja postotka dostižne maksimalne brzine ovisne o kutu greške štimanja jedra

Pokretanje broda implementirano je u jednoj komponenti unutar *BoatMovement.cs* datoteke. U *Update* funkciji (Slika 6.) komponente prvo se određuje najmanji kut između jedra i vjetra (linije 50 - 53). Zatim se računa kut između broda i vjetra, te se iz njega preko druge krivulje isčitava najbolji kut jedra (linije 55 - 57). Onda se izračuna greška u najboljem i stvarnom kutu jedra (linija 59). Zatim se izračuna maksimalna brzina koju brod može postići u trenutnoj konfiguraciji množeći vrijednost prve krivulje (maksimalna brzina), vrijednost treće krivulje (postotak moguće brzine zbog greške) i brzina vjetra (linija 61). Da brod ne krene instantno tom brzinom, dodan je *accelerationFactor* koji ga postepeno ubrzava/usporava prema dobivenoj brzini (linije 63 - 67).

Text

Description automatically generated

Slika 6. *Update* funkcija *BoatMovement* komponente

## Skretanje broda

Okretanje broda implementirano je u jednoj komponenti u datoteci *BoatSteering.cs*. Komponenta samo okreće brod oko njegovog središta brzinom koja ovisi o brzini broda i jačini skretanja koja je učitana preko tipkovnice od korisnika.

# Vizualni elementi

## More

Vizualizacija mora koristi vanjsku biblioteku *Unity*-a pod nazivom *CREST*[2], pri čemu se koristi dio funkcionalnosti uključujući simulaciju oceana i valova, dok sama biblioteka sadrži brojne ostale funkcionalnosti poput vizualizacije jezera, rijeka, plićaka, podmorja i ostalih. More, odnosno ocean, popunjava većinu scene simulacije te sadrži valove koji dolaze iz smjera vjetra. Uz sve to CREST pruža vjernu vizualizaciju prolaska objekta (jedrilice) kroz more stvaranjem pjene i manjih valova na putanji prolaska objekta(slika 6).

Površina mora unutar *CREST*-a se renderira na standardni način za vodene shadere, odnosno lomi scenu. Lomljenje se provodi uzorkovanjem teksture boje kamere koja ima samo neprozirne površine. Tijekom renderiranja u međuspremnik se zapisuje dubina površine kako bi se osiguralo da su valovi koji se preklapaju ispravno sortirani u kameri.

Povezivanje na API *CREST*-a vrši se kroz *Awake* funkciju *Wind.cs* datoteku u kojoj se kroz parametear *ShapeGerstnerBatched CREST* skripte *\_weight* postavlja odgovarajuća jačina valova za trenutnu jači vjetra.

A body of water

Description automatically generated with medium confidence

Slika 7. *Površina mora s valovima*

Slika 8. *Reakcija površine mora na prolazak objekta kroz more*

## Model jedrilice

Model jedrilice preuzet je besplatno s web stranice Sketchfab[3]. Prije uvoza modela u Unity bilo je potrebno razdvojiti model jedra od ostatka jedrilice tako da se može neovisno o njemu okretati, to je napravljeno pomoću alata Blender[4] za 3D modeliranje.

## Korisničko sučelje

Korisničko sučelje

A sailboat in the ocean

Description automatically generated with low confidenceGraphical user interface

Description automatically generated

# Zaključak

Razvijena simulacija je dovoljno vjerna stvarnosti da bi se mogla koristiti za treniranje početnika odnosno stvaranja intuicije jedrenja kod osoba koje nisu upoznate s jedrenjem. Još uvijek nedostaju neke komponente jedrenja, tako da simulacija nije iskoristiva u nekakve profesionalnije svrhe, te vjerojatno nije nadogradiva zbog limitiranja mogućnosti pojednostavljenjem modela fizike krivuljama.

# Literatura

[1] Unity3D dokumentacija: <https://docs.unity3d.com/Manual/index.html>

[2] CREST dokumentacija: <https://github.com/wave-harmonic/crest>

[3] Model jedrilice: <https://sketchfab.com/3d-models/sailboat-76d0b1e24be14d2f9a524bfce3001aeb>

[4] Blender: <https://www.blender.org>

[5] The Physics of Sailing: <http://grizzly.colorado.edu/~rmw/files/papers/PhysicsofSailing.pdf>