Izrada edukacijskog programa za učenje programiranja CNC strojeva na Microsoft Windows Presentation Foundation platformi

Šestan, Paolo

SVEUČILIŠTE U RIJECI
Studij politehnike

Diplomski sveučilišni studij politehnike i informatike

Paolo Šestan

Izrada edukacijskog programa za učenje programiranja CNC strojeva na Microsoft Windows Presentation Foundation platformi

Diplomski rad

Mentor: Izv. prof. dr. sc. Marko Maliković

Rijeka, 2022.

SVEUČILIŠTE U RIJECI

Studij politehnike

Rijeka, 12. 03. 2021.

Zadatak za diplomski rad

Pristupnik: Paolo Šestan

Naziv diplomskog rada: **Izrada edukacijskog programa za učenje programiranja CNC strojeva na Microsoft Windows Presentation Foundation platformi**

Naziv diplomskog rada na eng. jeziku: **Development of educational software for teaching CNC machine programming using the Microsoft Windows Presentation Foundation platform**

Sadržaj zadatka:

Izraditi edukacijski program (okruženje) za učenje programiranja CNC strojeva koristeći Microsoft Windows Presentation Foundation platformu. Korisničko sučelje treba sadržavati prostor za pisanje strojnog G-koda i prostor za 3D simulaciju u koji će biti ugrađen već postojeći simulator CAMotics dostupan kao projekt otvorenog kôda. U edukacijski program odnosno okruženje ugraditi elemente specifične za takvo okruženje (uloge nastavnika/učenika, mogućnost dodjeljivanja prava, mogućnosti podjele i upravljanja grupama, mogućnosti zadavanja i predaje zadataka, suradničkog rada i tako dalje). U diplomskom radu opisati problem programiranja CNC strojeva, opisati osobine i namjenu Microsoft Windows Presentation Foundation platforme, opisati elemente WPF platforme koji se koriste u izrađenom edukacijskom programu, opisati edukacijske mogućnosti okruženja (uloge, prava, podjela i upravljanje grupama, način zadavanja i predaje zadataka, suradnički rad itd.), opisati na koji način radi program koji je kreiran (ulaz, izlaz, algoritmi i drugo), priložiti slike s prikazanim načinom rada programa i na kraju rada (kao prilog) priložiti cjelokupan programski kôd.

Mentor: Izv. prof. dr. sc. Marko Maliković

Voditelj za diplomske radove

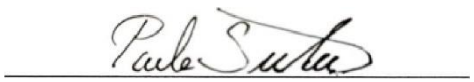
Zadatak preuzet: 15. 03. 2021.

(potpis pristupnika)

IZJAVA

Izjavljujem pod punom moralnom odgovornošću da sam diplomski rad izradio potpuno samostalno, isključivo korištenjem znanja stečenim na sveučilištu u Rijeci odsjeku za politehniku, korištenjem podataka iz navedene literature te uz stručno vodstvo mentora

Pristupnik: Paolo Šestan



(potpis pristupnika)

SADRŽAJ

1. UVOD

Računalni alati u ulogama nastavnih pomagala kojima je prvenstvena namjena približavanje koncepata učeniku kroz razne oblike vizualizacije ili grafičkog sažimanja nekog sadržaja postali su sve prisutniji u svakoj edukacijskoj okolini. Neovisno radi li se o „pasivnim“ alatima za vizualizaciju nekog koncepta ili principa (poput slikovnog prikaza, animiranja, grafičkog prikaza, itd.) ili se radi o sveobuhvatnim simulatorima kompleksnih fizičkih pojava ili aktivnosti. Neosporiv utjecaj takvih alata na svim razinama obrazovnog djelovanja lako je zamijetiti već i pri površnom promatranju nekog nastavnog procesa, posebice u nastavi radno-tehničkog područja. Računalni alati za učenje programiranja namijenjeni specifičnim uzrastima učenika postoje i u širokoj su primjeni već značajni niz godina, a alati poput primjerice Scratch programskog sustava predstavljaju jedan od prvih oblika takvog tipa nastavnog sredstva. Pojednostavljivanje koncepata prisutnih u programiranju, njihov prikaz na intuitivan i vizualno privlačan način te omogućavanje grafičke reprezentacije rezultata kreiranog programa neke su od najvažnijih značajki koje Scratch sustav čine toliko važnim i široko prihvaćenim u obrazovnim krugovima diljem svijeta. S druge strane, ako promotrimo neko drugo, odvojeno područje programiranja kao što je to programiranje CNC strojeva, odnosno pisanje G-kôda moguće je zamijetiti određene razlike. Većina alata namijenjena izradi CNC programa omogućuju zadovoljavanje tek osnovnih potreba neke obrazovne situacije, obrazovne značajke takvih alata uglavnom se svode tek na osiguravanje sintaktičke točnosti pisanog kôda te pridržavanje odabranim konvencijama pisanja samog programa, čak je i alatima koji imaju funkciju potpune 3D simulacije radnog stroja, teško zadovoljiti i one osnovne potrebe odgojno-obrazovnog okruženja. Iz navedenih je razloga upotreba takvih alata kao pomagala u nastavi često manje je nego idealna. Glavni nedostatak alata toga tipa se najčešće očituje u nedostatku velikog broja potrebnih funkcija kakve bi takav sustav edukacijske namjene trebao posjedovati. U ovom će radu biti predstavljene potrebe i zahtjevi koje alat za učenje CNC programiranja u profesionalnoj ili općoj obrazovnoj okolini treba zadovoljavati kako bi u što većoj mjeri osigurao kvalitetu provedene nastave. Na samom kraju rada biti će predstavljen razvoj konceptne izvedbe računalnog alata kreiranog pomoću se Windows Presentation Foundation programskim bibliotekama za izradu korisničkih sučelja. Izrađeno sučelje ukomponirati će postojeću aplikaciju (dostupnu kao alat otvorenog kôda) za 3D simulaciju izvođenja G-kôda koja će uz dodatak novog sučelja i funkcionalnosti doseći željenu razinu prilagodbe za korištenje u obrazovnoj okolini.

2. UČENJE PROGRAMIRANJA

Otkako je praksa izučavanja programiranja u obliku izdvojene znanstvene discipline proširena u srednjem, a kasnije i u osnovnoškolskom obrazovanju, postojala je potreba za olakšavanjem shvaćanja karakterističnih problema programiranja novoj, izrazito širokoj i raznovrsnoj publici. Predočavanje problematike i koncepata prisutnih i u onim najosnovnijim zadacima programiranja može učeniku koji se sa njima po prvi puta susreće predstavljati značajnu i naoko nepremostivu prepreku u savladavanju nekog dijela gradiva. Kako bi nastavnici bili u mogućnosti učenicima dočarati ne samo osnovne, već i one nešto naprednije koncepte različitih programskih jezika, pojavila se je potreba za sve značajnijom implementacijom dodatnih nastavnih sredstava u svakodnevnom poučavanju programiranja.

2.1. Problematika prijenosa informacije i nastavna pomagala

Kao značajnije prepreke u procesu prijenosa znanja sa nastavnika na učenike, vrlo često se pojavljuju upravo poteškoće shvaćanja novog, nepoznatog gradiva veće razine kompleksnosti. U situaciji kada nastavnik učenicima mora predstaviti koncepte sa kojima su u dotadašnjem obrazovanju imali vrlo mali ili nikakav doticaj, do izraza dolazi nastavnikovo rukovanje metodičkim principima i njihova primjena u praksi. Nastavnikove se vještine nadopunjuju njegovim poznavanje razreda kojeg podučava ali i u velikoj mjeri sposobnošću efektivnog raščlanjivanja gradiva na manje cjeline koje će razred moći shvatiti. Uspješnost nastavnika da ovlada takvim i sličnim situacijama karakterističnim procesu poučavanja ima direktan učinak na razinu učinkovitosti samog poučavanja [1], a samim time i na opći uspjeh učenika u savladavanju gradiva. Iako bi bilo idealno da svaki nastavnik posjeduje određenu razinu sposobnosti savladavanja takvih situacija, činjenica je da ta crta osobnosti nije norma kod svakog pojedinca, te ne može biti apsolutno mjerilo uspješnosti i kompetencija nastavnika u njihovom djelovanju. Jedna od svrha nastavnih pomagala je i upravo ta [2] da smanjuju negativni utjecaj urođenih sklonosti pojedinca na krajnji rezultat njegova djelovanja u ulozi nastavnika. Kada se učenici po prvi puta susreću sa programiranjem kao nastavnom temom (često u petom razredu osnovne škole, moguće i ranije) u školskom okruženju, nastavnik informatike vrlo rijetko ima priliku gradivo kao samostalni koncept programiranja povezati sa nekom drugom, ranije izučavanom temom. Početno uvođenje programiranja u nastavu često se sastoji tek od osnovnog povezivanja sa poznatim oblicima primjene ili „opipljivim“ (vidljivim) rezultatima njegovog postojanja poput: video igara, kalkulatora, operacijskih sustava, itd. Ti su primjeri iako sveprisutni i poznati većini učenika, i dalje nedostatni u svojoj svrsi opisivanja stvarnog potencijala programiranja kao i razvoju

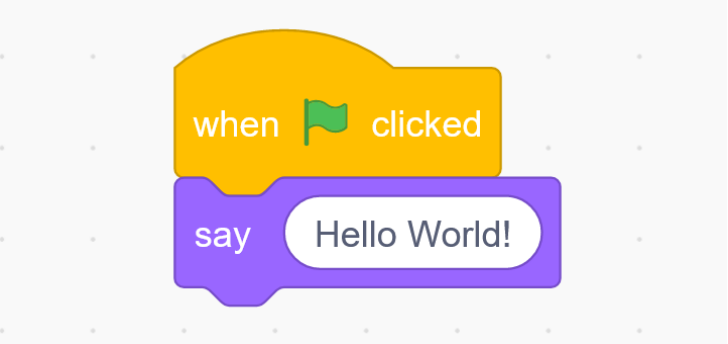
učenikova shvaćanja temeljnih koncepata poput programskih naredbi, instrukcija ili tipova podataka. Osnovica razvoja programskog razmišljanja u učenika osnovnih škola nerijetko se sastoji tek od rješavanja zadataka u nekom odabranom programskom jeziku sa povećavajućom razinom tehničke kompleksnosti do postizanja razine kompetencije zahtijevane nekim predmetnim kurikulumom ili postavljenim nastavnim ciljem. Takav pristup pri pokušaju pobude motivacije i razvoju programskog mišljenja često može imati i potpuno suprotan učinak od željenog. Frustriranost, nezadovoljstvo vlastitom sposobnošću savladavanja, kao i neshvaćanje potrebe i važnosti same teme, sve su to razlozi razvijanja odbojnosti prema programiranju u najranijoj fazi učenikovog kontakta sa tim gradivom, a koja može potrajati daleko u budućnost njegova školovanja pa čak i narednog profesionalnog rada [3]. Takvi i slični negativni utjecaji na učeničko mišljenje i doživljaj teme može biti izravno prouzročeno nerazumijevanjem temeljnih principa, zadaća i primjena programiranja u samom početku njegovog izučavanja [4]. Nastavna pomagala u gradivu programiranja imaju nezavidnu zadaću predočavanja njegovih osnovnih koncepata učenicima koji se po prvi puta u svome životu susreću sa ovim tipom apstraktnog koncepta. Također, učenici u tim dobnim skupinama su često tek u procesu fiziološkog razvoja sposobnosti razumijevanja logičkih principa potrebnih za shvaćanje koncepata programiranja [5]. Nastavna pomagala koja su najpogodnija za primjenu u okruženju računalne ili informatičke učionice za obradu teme programiranja su vrlo često upravo računalne aplikacije te razni programi ili animacije. U tu svrhu razvijeni su brojni računalni alati za potporu nastavniku od kojih je nemali broj preporučen od strane uglednih obrazovnih vladinih i ne-vladinih organizacija.

2.2. Vizualno učenje

Istraživanja su pokazala kako se značajan broj ljudi u obrazovnom okruženju ali i općenito u svijetu pribraja upravo takozvanom „vizualnom“ tipu učenika [7]. Vizualni tip učenja ili „Vizualno učenje“ pristup je učenju pri kojem se koncepte, principe i opću građu neke teme u što većoj mjeri nastoji prikazati kroz lako razumljive grafičke elemente poput grafova, objekata, animacija, i dr. Takvim pristupom razvoju razumijevanja primijećen je pozitivan odziv u konačnim rezultatima, pobudi motivacije, kao i u dugotrajnoj retenciji znanja [8]. Vizualno je učenje također izrazito lako primjenjivo na području podučavanja programiranja, te se već naširoko koristi u svrhu ublažavanja gore navedenih negativnih strana korištenja nekih od klasičnih nastavnih metoda u podučavanja programiranja.

2.2.1. Vizualno programiranje

Kao jedan od najraširenijih računalnih alata za učenje programiranja i u onoj najranijoj dobi (niži razredi osnovnih škola) često se spominje upravo sustav alata Scratch. Scratch predstavlja sustav koji se sastoji od programskog jezika zasnovanog na principima vizualnog programiranja, uređivačkog okruženja za „pisanje“ programa te raznovrsnog popratnog sadržaja čija je namjera zaokupiti i usmjeriti pažnju i onih manje zainteresiranih za sam proces programiranja. Sam sustav je osmišljen kao prvenstveno edukacijski za osobe od 8 do 16 godina i kao idealan prvi doticaj učenika sa programiranjem. Usprkos tome, Scratch predstavlja izrazito koristan alat i za osobu bilo koje dobi koja se po prvi put upušta u upoznavanje svijeta programiranja. Dok se u klasičnom programiranju kao što je na primjer jezik C++ naredbe ispisuju „ručno“, u obliku teksta, Scratch ima pripremljene takozvane „blokove“ za svaku od naredbi koje nudi. Povlačenjem, slaganjem i međusobnim povezivanjem blokova raznih namjena na radnoj površini implementira se program željeni program ili algoritam. Samom je primjenom unaprijed kreiranih blokova umjesto ručnog unosa teksta programski jezik Scratch uspio u otklanjanju nekoliko od najčešćih grešaka koje se mogu pojaviti u počecima ovladavanja programskim načinom razmišljanja. Kako se tu radi o greškama najčešće sintaktičke prirode ili o tek semantičkoj razlici nekih programskih funkcija naredbi ili operatora, njihovo je zanemarivanje u početničkom izučavanju ne samo dozvoljeno, već često i potpuno poželjno. Taj se pristup primjenjuje zbog činjenice da učenje na takvim greškama u ranijim fazama obrazovanja ni u kojem slučaju ne pridonosi bržem ili kvalitetnijem razvoju programskog načina razmišljanja. Kod usporedbe vizualno i klasično pisanog programa (slike 2.1. i 2.2.) odmah je primjetno atraktivan i šareni izgled kôda pisanog Scratch programskim jezikom (slika 2.1.) kao i jednostavnost kojom je sam program ispisan što je u direktnoj suprotnosti sa kôdom na slijedećoj slici (slika 2.2.) pisanim u C# programskom jeziku. Iako je ovdje demonstrirani „klasični“ programski jezik naširoko smatran kao jedan od modernih jezika napredne sintakse i izuzetne lepeze mogućnosti, upravo su te karakteristike neke od njegovih najvećih mana i ono što ga po mišljenju mnogih [8] čini nepogodnim kao instrumentom za učenje osnovnih koncepata programiranja. Scratch ne posjeduje niti približno količinu mogućnosti u usporedbi sa jezikom poput C# ili pak nekih drugih kao što su C++, Python, JavaScript, PHP, itd. ali zato posjeduje kvalitete koje ga čine jednim od najboljih alata za učenje programiranje uopće.



Slika 2.1. Program pisan u Scratch programskom jeziku

```
using System;

public class Program
{
    public static void Main()
    {
        Console.WriteLine("Hello World");
    }
}
```

Slika 2.2. Program pisan u C# programskom jeziku

Scratch će kroz svoje jednostavno sučelje i još jednostavniju primjenu osnovnih funkcija programiranja učeniku omogućiti da svoju ideju u nekoliko „klikova“ pretvori u funkcionalan program te da u stvarnom vremenu na „pozornici“ promatra direktan rezultat svog upravljanja računalom (Slika 2.3.). Upravo je takav pristup kratkog vremenskog razmaka između učenikovog pisanja programskog kôda i prikaza rezultata njegovog programiranja ključan u razvoju rane motivacije prema toj i sličnim temama u računalstvu, informatici ali i šire. Takav oblik brze validacije učenikova rada često ne postoji kod klasičnog programiranja, ili je pak rezultat prikazan tek kroz manje atraktivan konzolni ispis, kao u primjeru na slici 2.4),

Slika 2.3. Izlaz Scratch programa sa slike 2.1.

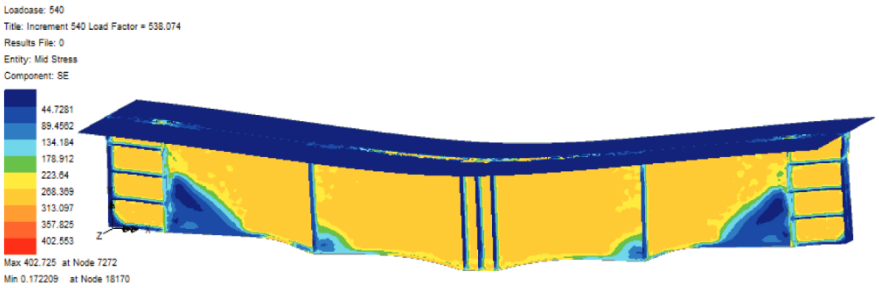
```
Hello World!

C:\Users\Korisnik\source\ConsoleApp1\bin\Debug\net5.0\ConsoleApp1.exe (process 20600) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window...
```

Slika 2.4. Izlaz C# programa sa slike 2.2.

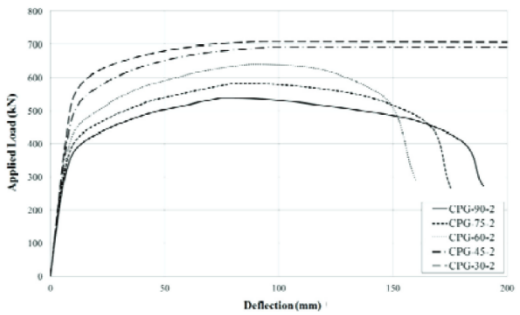
2.2.2. Simulatori i alati za grafičku analizu

Iako prethodno predstavljeni oblik vizualnog programiranja može biti itekako koristan u okvirima alata za koje su osmišljeni, rijetki su alati poput Scratch sustava koji su korišteni izvan nekog oblika obrazovnog okruženja. Prvenstveno zbog ograničenja koja se uglavnom tiču smanjenih mogućnosti, otežane implementacije programerskih paradigmi (objektno orijentirano, funkcionalno, proceduralno programiranje itd.) i sličnih, takvi se sustavi uvelike smatraju tek početničkim pomagalima. Drugi poznati oblik vizualizacije procesa na računalu pri učenju i poučavanju, tiče se prikaza rezultata naprednih matematičkih jednadžbi, strukturnih analiza ili fizičkih pojava kroz grafičko označavanje na realnim računalnim modelima. Svima su poznati dvodimenzionalni grafovi koji na svojim osima omogućavaju prikaz odnosa i usporedbu dvaju ili više veličina koje se mjere. Vizualizacija skupova podataka kod kojih vrijednosti pojedinih veličina ovise o međusobnom odnosu većeg broja drugih veličina te njihovo izdvojeno promatranje nije smisleno u vidu šire slike stanja promatranog sustava, problem je koji se nastoji

riješiti kroz detaljne grafičke analize i vizualizaciju rezultata sa što većom razinom istinitosti u odnosu na stvarno stanje promatranog sustava. Takve analize nazivaju se simulacijama, a računalni alati koji se koriste u tu svrhu, simulatorima. Izlazni podaci takvih alata, ako se primjerice radi o simuliranju utjecaja neke fizičke veličine nad stvarnim predmetom su najčešće ucrtani direktno na 3D model promatranog predmeta. Veličine se mogu prikazati u obliku boja čije nijanse označavaju iznos simulirane veličine, smjer djelovanja, ili pak bilo koji drugi parametar dobiven provedbom simulacije. Usporedbom 3D ucrtanih rezultata sa dvodimenzionalnim prikazom grafa istih veličina, već i na prvi je pogled moguće vidjeti razliku u razumljivosti i intuitivnoj jednostavnosti samog prikaza (slike 2.5. i 2.6.)



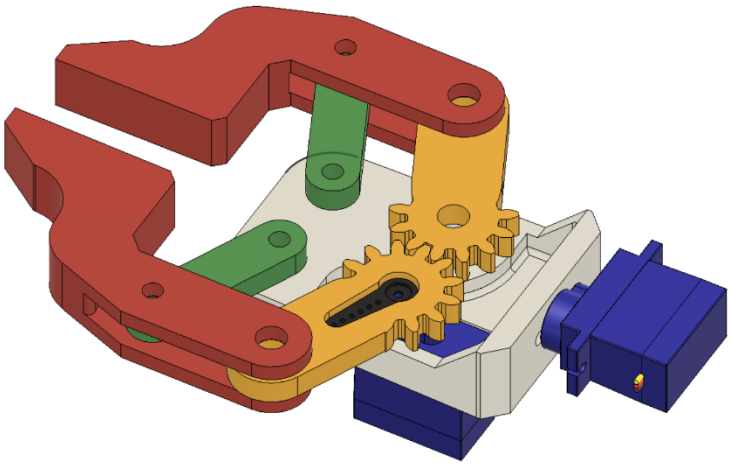
Slika 2.5. 3D prikaz rezultata analize naprezanja grede. [9]



Slika 2.6. 2D prikaz rezultata analize naprezanja grede. [9]

Na prikazanim se principima zasnivaju sve simulacije mehaničkih kretnji, mehanizama, deformacija itd. čija je prvenstvena namjena prikaz rezultata kompleksnih izračuna nekih fizičkih pojava na čovjeku intuitivno razumljiv način. Za prikaz simuliranih kretnji nekog kinetičkog mehanizma ili međusoban odnos nekih njegovih dijelova, koriste se simulacijski programi često integrirani kroz razne oblike CAD paketa programa koji nerijetko samu simulaciju omogućuju ne kao središnji dio mogućnosti programa, već kao jednu od njegovih mnogobrojnih dodatnih značajki. Simulacija kretanja i međusobne interakcije dijelova mehanizama može u svome najjednostavnijem obliku predstavljati tek nešto poput ograničenja oblika rotacijske kretnje i onemogućene translacije kotača postavljenog na osovinu, ali na isti način može predstavljati i

simulaciju većeg broja interakcija među komponentama čitavog sklopa različitih oblika osovina, poluga ili zupčanika u povezanoj cjelini kao na slici 2.7.



Slika 2.7. Simulacija međusobne interakcije većeg broja komponenti u programu Autodesk Fusion 360

Ono što je zajedničko alatima za vizualizaciju fizičkih veličina ili pojava poput naprezanja, topline, trenja, mehaničkog otpora, itd. sa računalnim alatima za vizualno učenje i programiranje jest to da obije grupe alata međusobno dijele svoju temeljnu svrhu. Svi spomenuti alati nastoje uz pomoć vlastitog načina grafičke komunikacije sa korisnikom predstaviti svoje izlazne vrijednosti (podatke) na vizualno atraktivan i jednostavnije razumljiv način u usporedbi sa prikazom sirovih podataka u izvornom (tekstualnom) obliku.

2.3. Predstavljanje apstraktnih koncepata

Iako je prikazom podataka u nekom od oblika 3D vizualizacije često moguće vrlo kompleksan skup podataka prikazati na način razumljiv i slabo upućenoj osobi (na primjeru sa slike 2.5. jednostavno je zaključiti koji dio grede je pod većim opterećenjem i na koji način dolazi do deformacije), takav prikaz ipak nije idealan za sve oblike podataka koje je potrebno vizualizirati. U slučajevima vizualizacije nekog procesa, sam je princip vizualizacije u većoj mjeri usmjeren na pojednostavljivanje pojedinih koraka procesa i njihova provođenja nego na grafički prikaz krajnjeg izlaza toga procesa. Gore opisano vizualno programiranje na primjeru sustava Scratch jedan je od oblika takve vizualizacije s ciljem približavanja same teme većem broju ljudi i upoznavanja njenih baznih koncepata. U nastavku ovoga poglavlja biti će prikazana još nekolicina

računalnih alata koji se koriste sličnim principima u svrhu razvoja vlastitog pristupa pojednostavljivanja nekog njima jedinstvenog procesa.

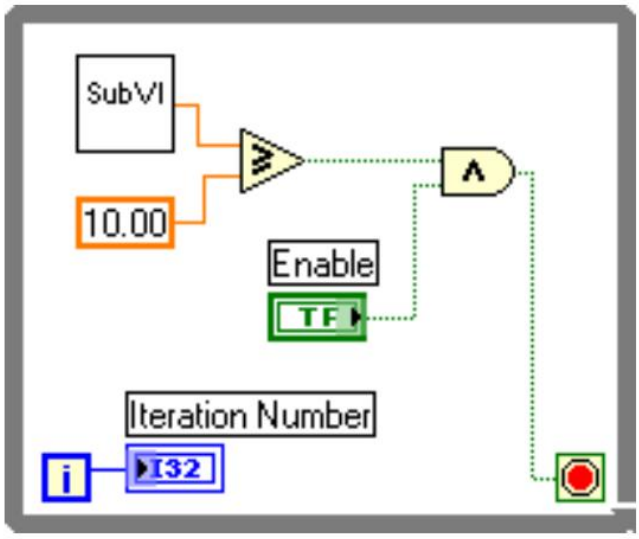
2.3.1. Petlje i iteracija

Koncept petlji i iteracije (ponavljanja) u programiranju često može biti jedan od teže shvatljivih dijelova za osobu koja se tek susreće sa disciplinom programiranja [10] neovisno o dobi same osobe. Kako je često običaj za tečaja programiranja, prvi pisani programi najčešće započinju u obliku linearno izvođenih ispisivanja slova i brojeva na ekran kroz neki oblik konzolnog sučelja (slika 2.4.). Tim je pristupom, do trenutka upoznavanja petlji i drugih oblika nelinearnog (skokovitog) izvođenja u nastavku tečaja moguća pojava zbunjenosti ili osjećaj prekomjerne zamršenosti gradiva kod polaznika. Koristeći se principima utemeljenim na jednostavnim naredbama, programski jezik Scratch na isti način implementira i iteraciju kroz nekoliko različitih oblika petlji („while“, „do-while“ i „for“ petlje) koristeći se nazivima poput „Repeat“ (eng. Ponavljaj), „Repeat until“ (eng. Ponavljaj dok) (slika 2.8.) i sličnim kako bi programsku funkciju naredbe približili standardnom govoru i samim time olakšali njeno razumijevanje i semantiku sveli na prirodno poznavanje govornog jezika. Osim sustava Scratch namijenjenog prvenstveno novim korisnicima,

Slika 2.8. Funkcija ponavljanja u Scratch programskom jeziku.

takvi se principi još koriste i u profesionalnim sustavima namijenjenim za upotrebu u industrijskim pogonima primijenjene automatizacije i praćenje rada strojeva ili robota. U takvim situacijama vizualno programiranje omogućava brzu provedbu ideje u gotov i zaokružen proces, iskorištavanje

i obradu dostupnih podataka uz direktnu povezanost i utjecaj na svoju stvarnu okolinu. Jedan od takvih sustava je i LabVIEW tvrtke National Instruments. LabVIEW uz osnovne metode zamjene programskih naredbi, funkcija i operatora sa grafičkim simbolima ili objektima poput Scratch-a također sadrži i veliki broj naprednih funkcija specifičnih za oblik primjene kakvoj je namijenjen. No dodatno podržava i suvremene standarde grananja, paralelizacije izvođenja programa te veliki broj programskih biblioteka specijaliziranih za različite svrhe ili namjene u specifičnom strojnom sklopovlju za čije je upravljanje pojedini sustav osmišljen. Programske petlje i upravljanje iteracijama u LabVIEW sustavu podržava čitav niz detaljnih modifikacija toka programa. Na primjeru jednostavne „while“ petlje (slika 2.9.) moguće je zamijetiti nekoliko načina upravljanja iteracijom, kao na primjer: promjena ulaznih veličina (trenutna vrijednost varijable na slici je 10,00) ili komponenata usporedbe (promjenjiva binarna TF vrijednost na slici je trenutno postavljena u „omogućeno“). Petlje u nastavi programskih jezika samo su jedan od problematičnih



Slika 2.9. „while“ petlja izvedena kroz LabVIEW sustav vizualnog programiranja. [11]

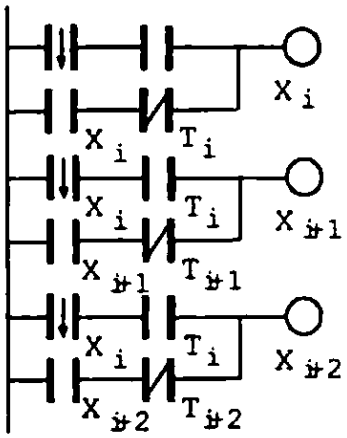
koncepata koji se često iskazuju kroz poteškoće u shvaćanju građe ili primjeni naučene teorije. Iako su petlje i dalje primjer tek osnovnog koncepta u programiranju, učenicima itekako mogu predstavljati prepreku za savladavanje i neke šire teme čije su one sastavni dio [12].

2.3.2. Ljestveni dijagrami

U industrijskim aplikacijama za postavljanje i najjednostavnijih automatiziranih sustava potreban je neki oblik programiranja upravljačkih sklopova. Strukture takvih sustava su u velikoj mjeri sastavljene od standardiziranih komponenti čime se postiže minimalna potreba za naprednim i jedinstvenim programskim rješenjima. Usklađivanje software-a sa većinom zahtijeva različitih

sustava može se u potpunosti osigurati tek modifikacijom bazne logike sustava, promjenom ulaznih (mjerenih) veličina, ili dodavanjem potrebnih parametara u kontrolni program. Takvo kreiranje upravljačkog programa za neki sustav tek u vrlo rijetkim slučajevima iziskuje specifična programska rješenja značajnije potrebe razvoja. U tu se svrhu koristi programiranje takozvanim „ljestvenim dijagramima“.

Ljestveni dijagrami su oblik vizualnog programiranja gdje se programske operacije prikazuju grafičkim simbolima (slika 2.10.) te glavnina toka programa ovisi o ulaznim vrijednostima (koji su najčešće fizičke veličine poput napona na izlazu senzora, itd.) te o logičkim operacijama koje se kroz sam tok programa nad njima izvršavaju.



Slika 2.10. Simboli i ulazne veličine korištene u ljestvenoj logici. [13]

U programiranju ljestvenim dijagramima moguće je izraditi iznenađujuće kompleksnu slijednu logiku kroz izuzetno ograničenu ponudu funkciji, operatora i drugih programskih paradigmi prisutnih u nekim naprednijim jezicima. Uz sva njena ograničenja, ljestvena logika prisutna je u gotovo svakom industrijskom okruženju kojem postoji potreba računalnog upravljanja nekim fizičkim strojem ili sustavom. Takav oblik upravljanja u svojoj srži simulira relejnu logiku prisutnu u nekadašnjim elektro-mehaničkim upravljačkim sustavima najčešće nadograđivanu suvremenim principima proceduralne paradigme klasičnog programiranja. Upravo su jednostavnost i skraćeno vrijeme razvoja odlike ljestvenih dijagrama koje je bitno naglasiti pri promatranju funkcionalnosti korisnih za edukacijsko okruženje.

2.4. Postojeće funkcionalnosti i karakteristike edukativnih alata

Zastupljenost nastavnih pomagala u obliku različitih vrsta računalnih alata, prezentacija i drugih audio-vizualnih elemenata u odgojno-obrazovnim okruženjima se je u posljednjih desetak godina

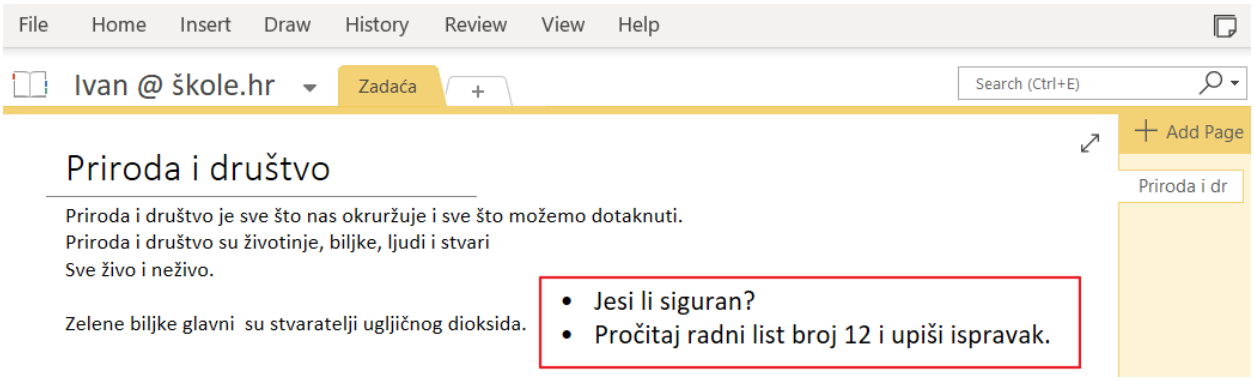
višestruko povećala [13]. Kroz upotrebu računala u suvremenoj učionici današnjice, upoznavanje, ponavljanje ili vježbanje gotovo svake teme ili gradiva može biti značajno pojednostavljeno ili ubrzano. Također, i samo savladavanje gradiva provodi se s povećanom efikasnošću kada se uz redovnu nastavu uključi i neki oblik računalnog alata ili sredstva. Nikad veća prisutnost računala u svakodnevnom životu učenika znači i njihovu sve napredniju informatičku pismenost koja dolazi do izražaja i prije provedbe nekog oblika ispitivanja kroz standardni nastavni proces. Takav razvoj čitavog društva pogoduje sve većoj primjeni računalnih alata u svim sferama obrazovanja. Nakon značajnog povećanja u primjeni, uslijedilo je i povećanje potražnje za takvim alatima u razvijenim i najzastupljenijim područjima školstva i obrazovanja.

Računalni alati čija je glavna namjena korištenje u obrazovnim okolinama, moraju osim samih osnova vjernog prikaza teme koju zastupaju, također težiti i određenim vrijednostima u organizacijskom smislu odgojno-obrazovne okoline. U opticaju postoji veliki broj računalnih alata koji mogu u potpunosti zadovoljavati edukacijske potrebe individualizirane okoline jednog pojedinca ili čak manju grupu učenika. No u slučajevima kada se radi o poučavanju čitavog razreda ili skupine studenata veće od 10 ili 15 osoba, izuzetno je poželjno da korišteni alati omogućuju i podupiru takav oblik istovremenog rada čitave grupe. Kako bi nastavnik bio u mogućnosti efektivno upravljati čitavom skupinom polaznika za vrijeme zadavanja zadataka, odrađivanja vježbi te predstavljanje teorijskih ili praktičnih principa neke teme, potrebne je da alat posjeduje takve mogućnosti. Aplikacija ili simulator koji prepoznaju ulogu nastavnika te mu omogućuju nadgledanje i moderiranje skupine u stvarnom vremenu za trajanja same nastave upravo su alati kakvi su potrebni u suvremenoj učionici. Najbitnije funkcionalnosti temeljene su na principima suradničkog učenja i međusobne povezanosti u realnom vremenu iz jednostavnog razloga da takve funkcionalnosti omogućavaju i značajno olakšavaju rad nastavnika i upravljanje učenicima u nastavnom procesu.

2.4.1. Mogućnosti edukacijskih programskih paketa

Potreba za mogućnošću međusobne komunikacije grupe učenika ili studenata sa nastavnikom i u situacijama kada njihova neposredna interakcija nije moguća u prethodnih se je par godina pokazala potrebitijom nego ikada. Kroz trajanje COVID-19 pandemije udaljena nastava se je odvijala mahom kroz različite oblike osiguravanja udaljene prisutnosti ili u nekim slučajevima kroz djelomično ili potpuno asinkronu nastavu kroz zadavanjem zadataka i vremenskog roka za njihovo rješavanje. Odvijanje nastave u budućnosti sigurno neće ostati nepromijenjeno nakon ovakvoga iskustva sa širokom primjenom komunikacijskih računalnih tehnologija, potreba za

udaljenom komunikacijom bit će veća nego ikada [14], a asinkrona komunikacija elektroničke pošte teško će moći zadovoljiti sve slučajeve upotrebe. Brojni alati u obrazovnoj sferi već pružaju komunikacijske funkcionalnosti jednostavne za primjenu i u obrazovnoj okolini. OneNote aplikacija tvrtke Microsoft omogućuje stvaranje zajedničke ili osobne bilježnice učenika nad kojom nastavnik može imati mogućnost pregledavanja, ispravljanja ili dodavanja komentara za neki dio pisanog rada, na slici 2.11 prikazan je pisani rad u osobnoj bilježnici učenika, a sa njegove desne strane je uputa i komentar nastavnika.



Slika 2.11. Primjer komentiranog učeničkog rada u OneNote bilježnici.

Ovakav oblik izravnog pristupa radu učenika ili studenta od strane nastavnika omogućuje ne samo pregledavanje gotovog rada na kraju vježbe ili zadatka, već i uvid u kontinuitet i napredovanje kroz čitavo vrijeme rada na zadatku, te uvelike može ubrzati reakciju na netočno ili nepoželjno rješavanje vježbe ili zadatka što će zauzvrat učeniku pravovremeno dati povratnu informaciju i priliku za ispravak te će umanjiti učestalost krivog učenja neke vještine. Kroz alate koji omogućuju međusobnu komunikaciju nastavnika i polaznika u bilo kojem obliku shodnom namjeni samog alata, moguće je značajno unaprijediti [14] veliki broj aspekata kvalitete održane nastave. Takva će postepena unaprjeđenja u konačnici izravno utjecati na razinu usvojenosti gradiva ili izučavane teme kod samih polaznika.

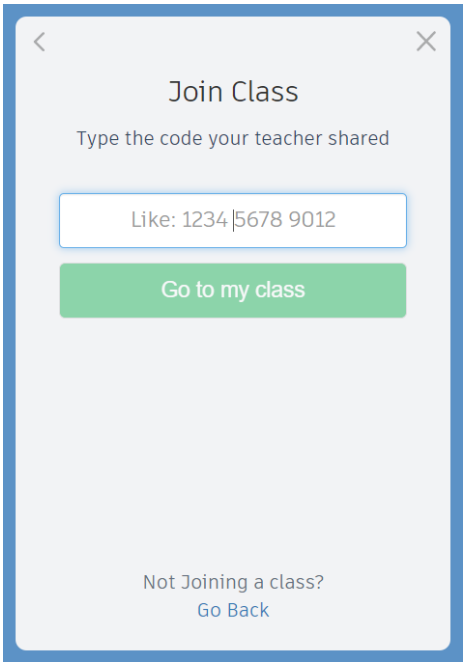
2.4.2. Suradničko učenje

Istraživanja [16] ali i dugotrajno praćenje nastavne prakse su pokazali kako bilo da se radi o učenicima u osnovnoj školi ili studentima na fakultetu, polaznici će gotovo uvijek, mjereno kroz njihove odgojno-obrazovne uspjehe, pozitivnije reagirati na rad u paru ili grupi nego na samostalno rješavanje sličnih vježbi. Zbog tih izuzetnih rezultata dokazanih znanstvenim metodama [16], takve bi oblike učenja bilo dobro obuhvaćati u što je većem broju nastavnih okolina. Dodatnu važnost implementaciji suradničkog učenja u nastavnim okolinama daje činjenica da kako je i u programerskoj profesiji suradnički rad vrlo česta svakodnevnica, te je izražena rijetkost samostalno rada jedne osobe na zadatku bez nadovezivanja na rad nekog drugog pojedinca ili grupe. Spomenuti primjer je samo jedan od mnogih koji iziskuju određene vještine suradničkog rada u profesionalnom kapacitetu, samim time od izuzetne je važnosti uvježbavanje takvih vještina kod osobe koja se osposobljava za rad u tom području.

2.4.3. Prednosti alata edukativne namjene

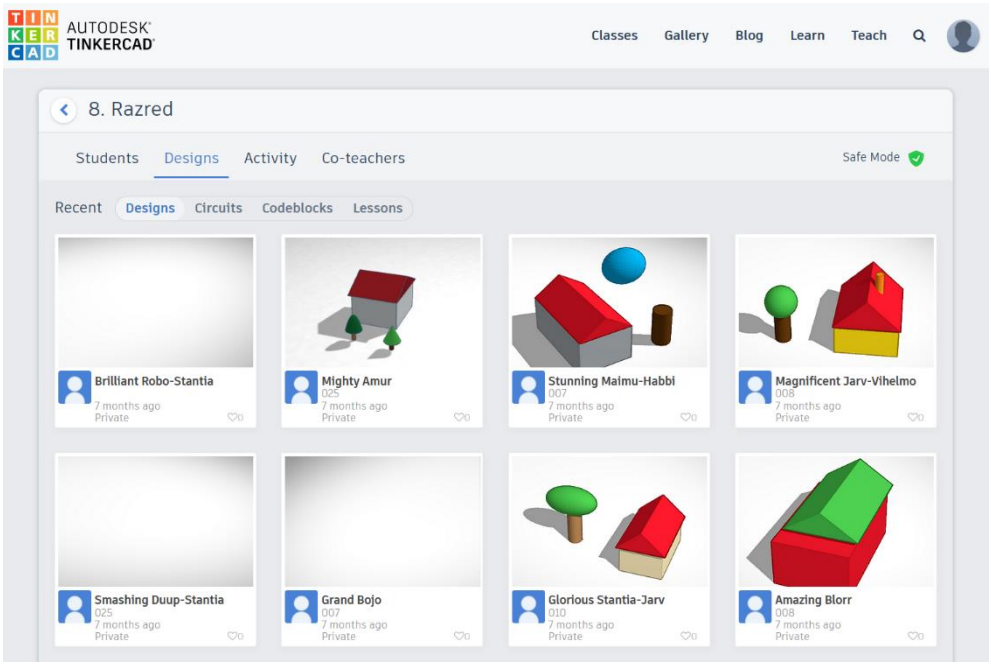
Tinkercad alat za 3D modeliranje tvrtke Autodesk predstavlja oblik edukativne aplikacije sa mnogobrojnim funkcionalnostima namijenjenim pojednostavljivanju rada sa većim brojem učenika u jednoj nastavnoj grupi, kao na primjer mogućnostima upravljanja razredom, zadavanju zadataka, pregledavanju radova, itd.

U svrhu analize ovakvog tipa mehanizma u edukativnim aplikacijama, promotriti ćemo funkcionalnost upravljanja razredom u Tinkercad aplikaciji. Kao jedna od glavnih značajki aplikacije ističe se mogućnost da korisnik-učitelj unaprijed kreira svakog pojedinog učenika za razred kojim će upravljati. Kreirani se učenici potom mogu pridružiti razredu samo upisivanjem šifre razreda u aplikaciju (slika 2.12.). Takvim je pristupom moguće zaobići čitav, obično obavezan proces kreiranja računa svakog pojedinog učenika. Dodatno se ovakvim mehanizmom uklanja značajan teret učitelju, posebno u cilju postizanju željenih rezultata u ograničenim vremenskim okvirima nastavne jedinice.



Slika 2.12. Pridruživanje učenika u kreiranu razrednu grupu.

Dodatna značajka Tinkercad aplikacije je integrirana funkcionalnost administracije kreiranih razreda. Svaki korisnik-učitelj ima mogućnost kreirati veći broj razrednih grupa sa kojima će u vidu nastavnog pomagala koristiti Tinkercad aplikaciju. Svaka od kreiranih grupa može posebno postavljati radne zadatke, grupne obavijesti, itd. Na slici 2.13. možemo vidjeti primjer otvorenog portfelja jedne grupe naziva „8. Razred“ u kojem se nalaze svi radovi pripadajućih učenika. Osim ovakvog pregleda radova čitave grupe, moguće je i pregledavati radove filtrirane po na primjer imenu ili nekoj drugoj značajki pojedinog učenika. Kroz testiranje ovih i sličnih mehanizama prisutnim u različitim modelima edukativnih aplikacija moguće je dobiti širu sliku potrebnih dodataka za budući razvoj aplikacije koja je predmet ovoga rada.



Slika 2.13. Lista radova jednog razreda u Tinkercad aplikaciji.

3. POSEBNOSTI IZUČAVANJA CNC PROGRAMIRANJA

Programiranje CNC strojeva, to jest za ovaj rad specifično, putanje radnog alata tro-osnog stroja u obliku vertikalne glodalice postiže se prosljeđivanjem koordinata i naredbi zapisanih G-kôd sintaksom. G-kôd predstavlja jedan oblik usko specijaliziranog programskog jezika, interpretiranog u koordinate kartezijeva sustava koje procesoru i elektronskom pogonu stroja daju naredbe o smjeru, udaljenosti i brzini pomaka radnog alata. Kako se svaki programski jezik razlikuje od drugih po nekim svojim specifičnostima, na isti je način i za pisanje G-kôd programa potrebno poznavati neke njemu specifične koncepte sastavljanja naredbi. Iako se sintaksa G-kôda od velike većine višenamjenskih klasičnih programskih jezika poprilično razlikuje, sama logika pisanja, analize i rješavanja problema u velikoj je mjeri ista ili vrlo slična. Spomenute karakteristike omogućavaju da se pristup samom procesu planiranja aplikacije približi arhitekturi edukacijskih aplikacija za učenje klasičnog programiranja za koje je dostupna puno pristupačnija literatura kao i istraživanja na tu temu, no i dalje se ne smije izostaviti činjenica kako se programiranje CNC strojeva u mnogočemu razlikuje od bilo kojeg oblika klasičnog programiranja te kao takvo iziskuje i određene specifičnosti pristupa njegovom izučavanju. U slijedećem ću tekstu nastojati prikazati neke od spomenutih sličnosti ali i specifičnosti koje svakako moraju biti naglašene i prepoznate u bilo kojem obliku edukativnog djelovanja.

3.1. Specifičnosti G-kôda

Svaki je oblik programa pisanog u G-kôdu izravno namijenjen izvođenju na nekom obliku mehaničkog stroja u stvarnome svijetu, zapravo i sam nastanak programskog jezika u obliku u kakvom G-kôd danas postoji, usko je povezan sa samom potrebom prosljeđivanja instrukcija računalno upravljanim mehatroničkom stroju. Prema tome, može se zaključiti kako se program pisan u svrhu izvođenja na nekom obliku upravljanog stroja i na, često usporedivo primitivnoj računalnoj podlozi (mikrokontroleri ili procesori ograničenih performansi) mora po mnogočemu razlikovati od klasičnih, prevođenih (u strojni kôd) ili interpretiranih programskih jezika namijenjenih izvođenju na klasičnijim računalnim okolinama. One karakteristike G-kôda koje su bitne za kasniju implementaciju u aplikaciji za njegovo učenje tiču se primarno načina prikaza pojedinih naredbi, linija i eventualnih funkcija u pisanom kôdu. G-kôd program svaku naredbu ili implementiranu funkciju zapisuje u obliku zasebne linije (slika 2.14) koje se izvode uglavnom sekvencijalno prema redoslijedu njihova zapisivanja, u takvom obliku program mora biti zapisan i u simulatoru koji će ga izvoditi.

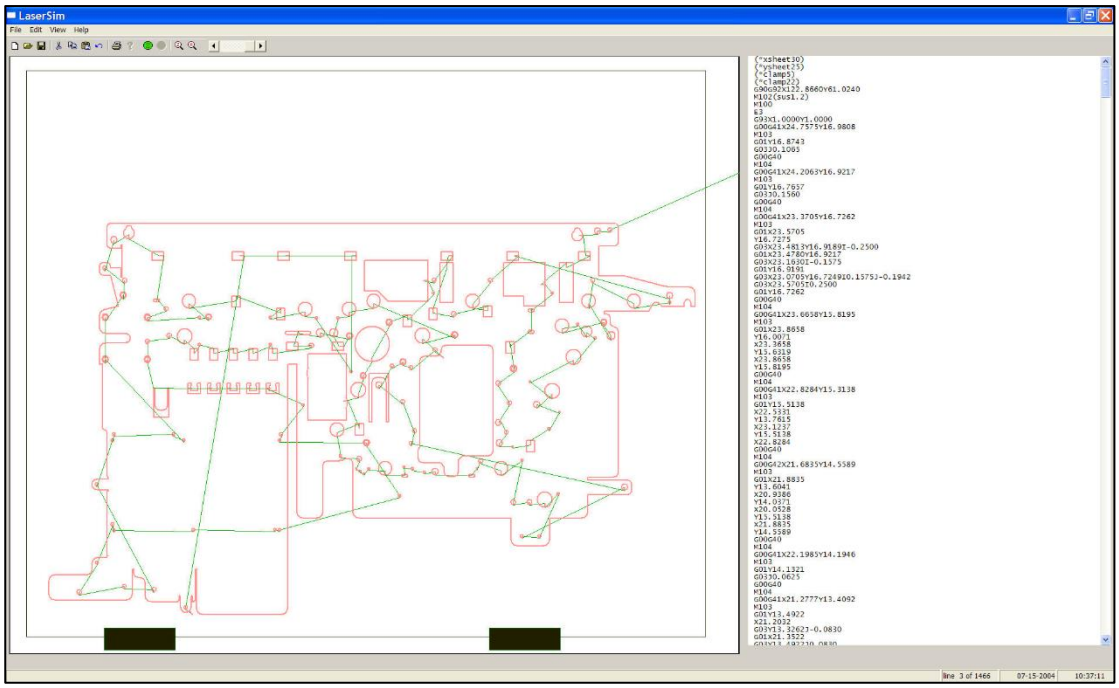
```
1   G21
2   M6 T1
3   F200
4   M3 S10000
5   G4 P10
6   G0 Z3
7   G0 X1.31 Y3.47
8   G1 Z-1.5
9   G1 X1.92 Y2.62
10  G1 X2.59 Y1.9
11  G1 X3.31 Y1.32
12  G1 X4.09 Y0.88
13  G1 X4.77 Y0.61
14  G1 X5.52 Y0.41
15  G1 X6.34 Y0.26
16  G1 X7.23 Y0.17
17  G1 X8.19 Y0.14
18  G1 X9.17 Y0.18
```

Slika 2.14. Linije G-kôd programa.

Na slici su sa lijeve strane vidljivi redni brojevi linija programa, dok se u prostoru sa desne strane nalazi sam CNC program sa svojim naredbama i iznosima pomaka po svakoj od osi stroja.

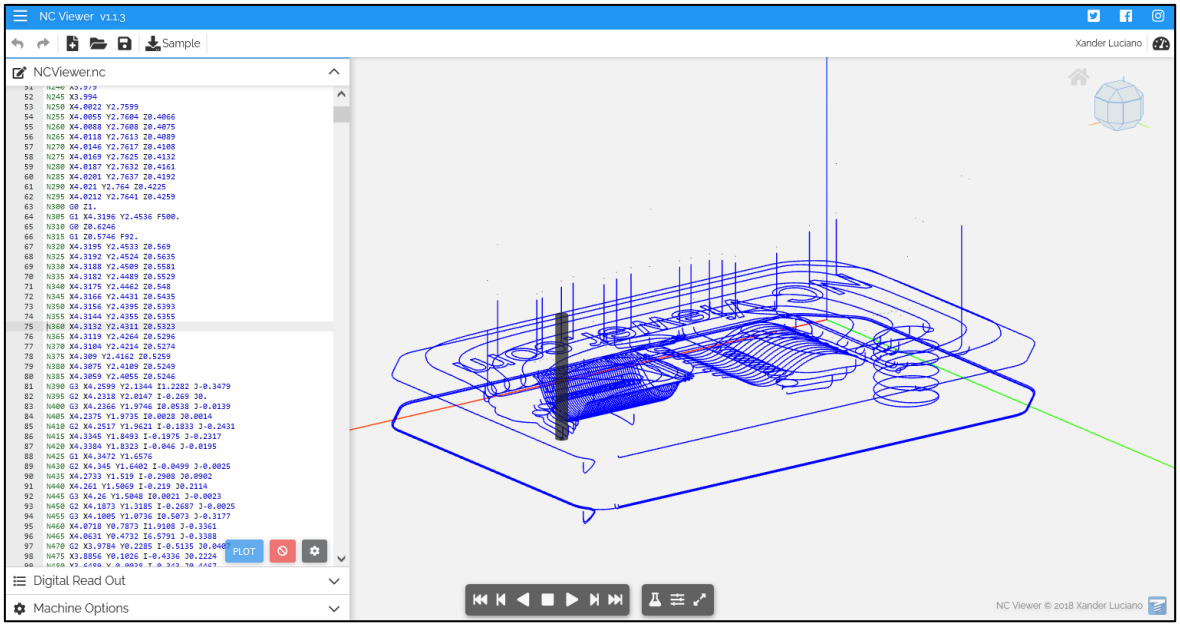
3.2. Simuliranje strojne obrade

U svrhu intuitivnog prikaza svake od naredbi zadanih kroz pisani G-kôd program, potrebno je njegovo stvarno djelovanje vizualno simulirati kroz neki oblik grafičkog prikaza. Za aplikacije koje nastoje korisniku prikazati takav oblik simulacije od velikog je značaja način na koji simulirani stroj obavlja svoju funkciju. Ako se aplikacija koja provodi simulaciju bavi prikazom radnog zadatka nekog stroja poput jednostavnijih plotera, CNC lasera ili sličnih sustava dvodimenzionalne radne domene, dovoljno je da sučelje kroz koje će aplikacija prezentirati simulaciju, također bude dvodimenzionalno. Na primjeru sa slike 2.15 vidimo prikaz iz LaserSim aplikacije specijalizirane za simulaciju rada CNC lasera koji predstavlja spomenuti oblik stroja sa radnom domenom u isključivo 2d prostoru.



Slika 2.15. Simulator rada laserskog CNC rezača,
LaserSim tvrtke CNCsoft.

Izvođenje simulacije u 2D prostoru višestruko je programski jednostavnije od sličnog zadatka u trodimenzionalnom prostoru. Već i samo pokretanje 3D simulacije iziskuje značajno veću količinu sustavnih resursa od 2D ekvivalenta. Uz prevođenje i simulacija CNC naredbi u stvarnom vremenu sa dopunskom trećom prostornom koordinatom dodatno se komplicira taj zadatak. Ako sliku 2.15. usporedimo sa sljedećom (slika 2.16), na kojoj je prikazan zadatak naoko slične kompleksnosti ali ovoga puta izvođen u trodimenzionalnom prostoru (stroj poput CNC glodalice ili sličnih) jasno je vidljivo kako prikaz takvog radnog zadatka kroz dvodimenzionalno sučelje neke aplikacije ne bi bilo moguće bez značajnog žrtvovanja intuitivnosti samog prikaza, a time i subjektivno doživljenog iskustva krajnjih korisnika. Može se prema tome zaključiti kako je trodimenzionalna simulacija neizbježna u slučaju kada se ona programski implementira za potrebe edukativne aplikacije (poglavlje 3.1., Zahtjevi alata u odgoju i obrazovanju) učenja CNC programiranja strojeva trodimenzionalne radne domene (ruteri, glodalice).



Slika 2.16. Simulator rada CNC glodalice NC Viewer tvrtke GCodeTutor.

Na primjeru simulatora NC Viewer prikazanog na slici 2.16. vidljivo je kako je prikazana samo putanja alata kroz radni prostor simuliranog stroja, zanemarujući pritom djelovanje oštrice nad radnim materijalom, njegov početni oblik, postupnu promjenu oblika ili bilo kakvu potencijalnu deformaciju radnog komada koja se može pojaviti u tijeku njegove obrade. Kako bi bilo moguće ispravno, realno i efektivno demonstrirati djelovanje čitavog programiranog procesa nad radnim komadom, u najmanju je ruku potrebno kroz simulaciju prikazati barem njegovo početno stanje (oblik, volumen), te njegovu promjenu uzrokovanu djelovanjem radnog procesa. Sam početni radni komad rijetko je problem prikazati jer se tu radi o statičkom trodimenzionalnom objektu čiji je prikaz u programski već prisutnoj 3D okolini trivijalno implementirati, no napredne funkcionalnosti rada aplikacije pojavljuju se problematikom simuliranja utjecaja alata na komad koji se obrađuje (na primjer: odstranjivanje čestica materijala glodalom). Za takav oblik dinamičke modifikacije trodimenzionalnog modela potreban je simulacijski softver kompleksnijih mogućnosti i s uglađenim sposobnostima upravljanja radnom memorijom sustava na kojem se pokreće zbog izraženijih potencijalnih negativnih posljedica na performanse pri izvršavanju računalnih zadataka tog oblika. Razvoj takvih i sličnih funkcionalnosti naprednije razine računalnog alata iziskuje proces programiranja aplikacije koji će vremenski značajno dulje trajati te će zahtijevati primjetno veće troškove, što je dodatni razlog da je za potrebe razvoja aplikacije koja je predmet ovoga rada korištena već postojeća komponenta (poglavlje 4.3.1., CAMotics alat za 3D simulaciju strojnog kôda).

3.3. Nedostaci postojećih računalnih alata

Od mnogobrojnih dostupnih računalnih alata namijenjenih pisanju programa i vizualno interpretiranoj simulaciji G-kôda, ne nedostaje onih koji podržavaju široku lepezu naprednih mogućnosti. Vrlo često su to alati koji sadržavaju i funkcionalnosti okarakterizirane u prethodnom poglavlju kao kompleksne za razvoj i implementaciju. Razlog je tome što na profesionalnom tržištu postoji dovoljna potreba za alatima koji omogućuju testiranje CNC programa prije njihova izvođenja na stvarnim strojevima gdje svaka greška u G-kôdu može biti vremenski ali i materijalno izrazito skupa. Usprkos postojanju značajne ponude na tržištu i njenoj istinskoj raznolikosti, glavni je dio ponuđenog softvera osmišljen kao profesionalni alat koji industrijskim korisnicima omogućuje smanjivanje potencijalno skupih grešaka spomenutih u tekstu iznad. Gotovo je nemoguće pronaći gotov alat koji će svojim funkcionalnostima uistinu udovoljiti zahtjevima odgojno-obrazovne okoline. Upravo je takav alat opisan u ovom radu. Korištenje alata fokusiranih na industrijske slučajeve za obrazovnu uporabu, u najmanju ruku je ne-efikasno, a često i značajno skuplje od alternative. Specifični dijelovi koji nedostaju postojećim alatima tiču se mahom funkcionalnosti opisanih na početku ovog poglavlja.

4. IZRADA ALATA ZA SIMULACIJU CNC PROGRAMIRANJA

U ovom će poglavlju biti predstavljen projekt izrade računalnog alata koji omogućuje pisanje strojnog G-kôda (CNC programiranje) i simulaciju rezultirajuće putanje tro-osnog CNC stroja. Temeljna ideja alata je potpuna prilagođenost uporabi u obrazovnoj okolini od viših razreda osnovne škole do naprednijih korisnika. Odabir i razvoj funkcionalnosti temelji se na mogućnostima nekih postojećih alata koji su se pokazali uspješnima u sličnim okolnostima kao i na jedinstvenim potrebama ovakvog alata. U nastavku ovog poglavlja detaljno opisujemo korištene tehnologije, kao i razloge za njihov odabir.

4.1. Zahtjevi alata u odgoju i obrazovanju

Kako bi se zahtjevi potrebnih funkcionalnosti aplikacije mogli konkretno definirati (izuzetno važan korak prije započinjanja samog razvoja), poželjno je promotriti neke od postojećih računalnih alata iz slične sfere uporabe i sličnim potrebama prosječnog krajnjeg korisnika u svrhu usporedbe poželjnih i nepoželjnih karakteristika takvog alata. Kod promatranja nekih od dostupnih računalnih aplikacija sa funkcionalnostima od značajnijeg interesa za ovu primjenu dovoljno je promotriti neke od ranije spomenutih funkcionalnosti i prednosti koje pružaju prvenstveno u odgojno-obrazovnim okolinama u radu sa većim brojem učenika ili polaznika. Nekoliko najvažnijih funkcionalnosti i karakteristika toga tipa predstavljene su u tablici 4.1. Osim tih temeljnih funkcionalnosti implementiranih u projektu i u početnim verzijama aplikacije, za njen je daljnji razvoj potrebno sastaviti iscrpni popis karakteristika i funkcionalnosti prema primjeru tablice 4.1. kako bi bilo moguće u potpunosti zadovoljiti zahtjeve korisnika i svake situacije koja se može pojaviti u odgojno obrazovnom okruženju.

Tablica 4.1. Važne funkcionalnosti aplikacije obrazovne namjene.

	Aplikacija	Funkcionalnost	Značaj i primjena
1.	Tinkercad	Suradnički rad	Motivacija korisnika i grupa na zajedničko rješavanje dobivenih zadataka
2.	OneNote	Komentiranje radova	Mogućnost ukazivanja na nedostatke i na načine poboljšavanja budućih radova
3.	Scratch	Atraktivna vizualizacija	Motiviranje i zaokupljanje fokusa korisnika
4.	Tinkercad	Podjela uloga i razreda	Mogućnost upravljanja većom grupom polaznika
5.	Tinkercad	Organizacija nastave	Zadavanje zadataka, ispita i vježbi u obrazovnoj okolini

4.2. Razrada projekta

Kroz izradu i planiranje projekta same aplikacije praćeni su principi ocrtani u ovom radu, kao i temeljnih zahtjeva koje aplikacija mora ispunjavati predstavljenim u poglavlju 4.1. Radno okruženje i funkcionalnost aplikacije u prvoj verziji mora zadovoljavati osnovne potrebe ovog tipa aplikacije predstavljene kroz prethodna poglavlja kao što su: karakteristike primjene suradničkog učenja, uloga nastavnika, predaja i pregledavanje gotovih radova, te administracija zadataka.
Temeljne funkcionalnosti aplikacije moraju se odražavati u nekoliko glavnih točaka:

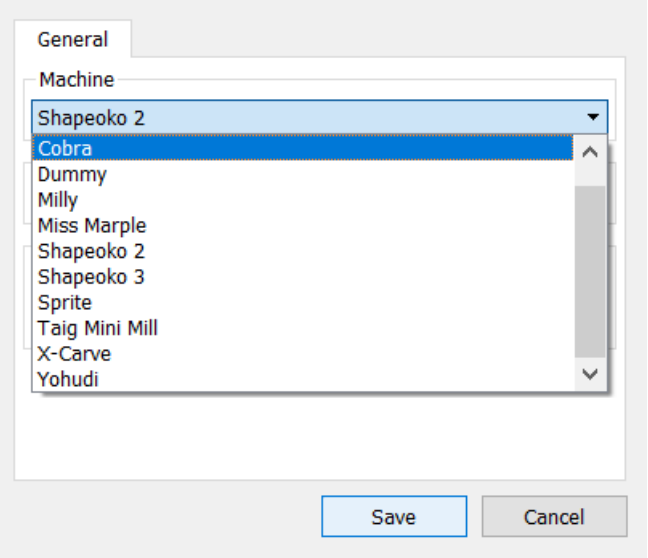
1. Središnji dio sučelja mora sadržavati prostor sa uređivačem teksta u koji će se upisivati naredbe G-kôda i dio u kojemu će se odvijati simulacija obrade komada materijala CNC strojem (glodalicom)
2. Simulaciju rada CNC stroja (obrade komada materijala) mora biti moguće pokrenuti, pauzirati i resetirati u toku njena izvođenja
3. Napisani G-kôd mora biti moguće spremiti u obliku tekstualne datoteke te tu istu kasnije otvoriti u aplikaciji
4. Nastavniku mora biti omogućeno pregledavanje napisanog G-kôda učenika kao i njegovo pokretanje u aplikaciji

Kako je iskustvo korisnika u ovakvom tipu aplikacije kao računalnog alata prvenstveno obrazovne namjene izuzetno važan dio čitavog projekta, ono i mora zadovoljavati nekoliko zacrtanih temeljnih principa:

1. Pregledno sučelje sa prikazanim minimalnim ali sasvim dostatnim brojem alata za obavljanje svih osnovnih zadaća aplikacije
2. Glavne funkcije aplikacije koje potrebuju neki oblik korisnikova unosa za njihovu aktivaciju (gumbi, padajući izbornici, itd.) moraju biti jednostavno dostupne i jasno naznačene (potrebno je, gdje god je to moguće koristiti prigodne ilustracije, slike ili zorno prikazane simbole)
3. Središnje sučelje mora zauzimati najveći dio dostupnog prostora zaslona te biti prilagodljivo potrebama i načinu korištenja svakog pojedinog korisnika.

4.3. Korištena tehnologija

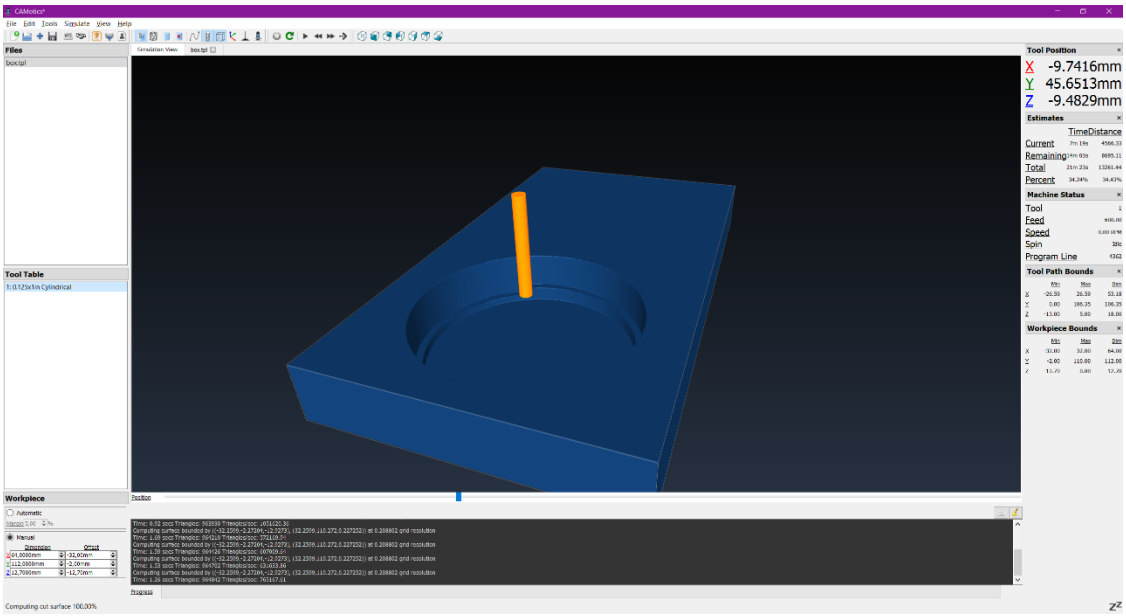
U svrhu izrade aplikacije u zadanim rokovima ovoga diplomskog rada, sama komponenta simulacije u aplikaciji koristiti će se već postojećim simulatorom CAMotics dostupnim u obliku projekta otvorenog kôda na GitHub stranicama. Korisničko sučelje će zahtijevati značajne promjene u odnosu na postojeće sučelje CAMotics aplikacije te će kao takvo biti iznova izrađeno implementiranjem prethodno opisanih funkcionalnosti pomoću Windows Presentation Foundation (WPF) platforme. U svrhu izrade alata koji će biti korišten u obrazovnoj okolini, bit će potrebno iz izvorne aplikacije ukloniti velik dio dostupnih naprednih mogućnosti koje u ovakvoj vrsti implementacije neće biti potrebne (primjerice, funkcija odabira vrste stroja kao na slici 4.1. kojim ćemo upravljati ne predstavlja toliko relevantni detalj za potrebe izučavanja samog načina pisanja osnovnog G-kôda). U ovom poglavlju ćemo opisati primijenjene principe programiranja, tok rada kroz razvoj projekta, te način implementacije spomenutih komponenti u zajedničku cjelinu.



Slika 4.1. Odabir vrste radnog stroja u CAMotics aplikacijii.

4.3.1. CAMotics alat za 3D simulaciju strojnog kôda

CAMotics aplikacija tvrtke Cauldron Development LLC predstavlja besplatan alat otvorenog kôda za simulaciju tro-osnog CNC programa kao i njegovu računalnu vizualizaciju u trodimenzionalnom prostoru. Aplikaciju je moguće pokrenuti na Linux, OS-X ili Windows operacijskim sustavim (www.camotics.org). Program aplikacije je pisan pretežito u C i C++ programskim jezicima (oko 49% i 43% ukupnog programskog kôda) te koristi XML format zapisa za podatkovne datoteke koje sadrže opće postavke aplikacije, spremljene postavke radne okoline ili kreiranog projekta u tijeku izvođenja. Izvorna namjena CAMotics aplikacije je, kako i možemo pročitati na njenim internetskim stranicama (www.camotics.org) „Pružanje mogućnosti detaljne simulacije programiranih CNC putanja u svrhu izbjegavanja opasnih i skupih pogrešaka u stvarnosti“ te je prema njoj vidljivo kako se i ciljana publika u najvećoj mjeri razlikuje u odnosu na aplikaciju koja je predmet ovoga rada. Prema tome dade se zaključiti kako će najvjerojatnije biti potrebne određene preinake u programskom kôdu da bi aplikacija mogla ispuniti sve zahtjeve postavljene u poglavlju 4.1. Iako ćemo se, kako je i u prethodnim poglavljima spomenuto, u ovoj implementaciji fokusirati isključivo na one najbitnije funkcionalnosti koje su primjenjive u aplikaciji edukativnog tipa za učenje CNC programiranja, CAMotics u svojoj izvornoj inačici podržava izuzetnu količinu radnih parametara i varijabli CNC stroja kojeg simulira. Na korisničkom sučelju izvorne aplikacije (slika 4.2.) ponuđen je veliki broj naprednih funkcionalnosti koje omogućuju istančano upravljanje simulacijom, od odabira samog stroja, kreiranja vlastitih alata definiranjem njihovih fizičkih



Slika 4.2. Izvorno korisničko sučelje aplikacije CAMotics.

parametara, ograničavanja radnog obima stroja, do prilagođavanja karakteristika i dimenzija materijala koji se obrađuje. Onaj dio funkcionalnosti bitan za ovu namjenu aplikacije sa najvećim fokusom na učenje samog programskog razmišljanja kod pisanja G-kôda odnosi se na mogućnosti povezanih sa: raščlambom i prijevodom upisanog G-kôda u programske koordinate i simulirane pokrete, datotečnim operacijama cjelokupnog sustava (spremanje, uređivanje i učitavanje datoteka) te trodimenzionalnim sučeljem i dinamičkom modifikacijom 3D objekata koji se u njemu nalaze kao dio simulacije (radni komad, alat, stroj). Potrebni dijelovi CAMotics aplikacije bit će ugrađeni u korisničko sučelje (okolinu) kreiranu prema spomenutim principima koristeći WPF programske biblioteke. O kojim je točno komponentama CAMotics aplikacije o kojoj ćemo detaljnije u poglavlju 4.3.2. i 4.4. gdje će između ostalog biti predstavljen tehnički način implementacije aplikacije u novo korisničko sučelje.

4.3.2. Windows presentation foundation

Windows presentation foundation je skupina programskih biblioteka, strukturalnih šablona i razvojnih alata koji omogućuju kreiranje desktop aplikacija za Windows (u novijim verzijama podržava i Linux te macOS) operacijske sustave. WPF je razvijen u okvirima .NET razvojne okoline te kao primarni programski jezik koristi C#, a datoteke strukture i dizajna korisničkog sučelja aplikacije organizirane su u XAML formatu. Poslovna i programska logika aplikacije razvija se kroz standardni C# kôd u objektno orijentiranoj programskoj paradigmi, dok se raspored grafičkih elemenata korisničkog sučelja: njihov oblik i izgled (veličina, boja, dodatni sadržaji)

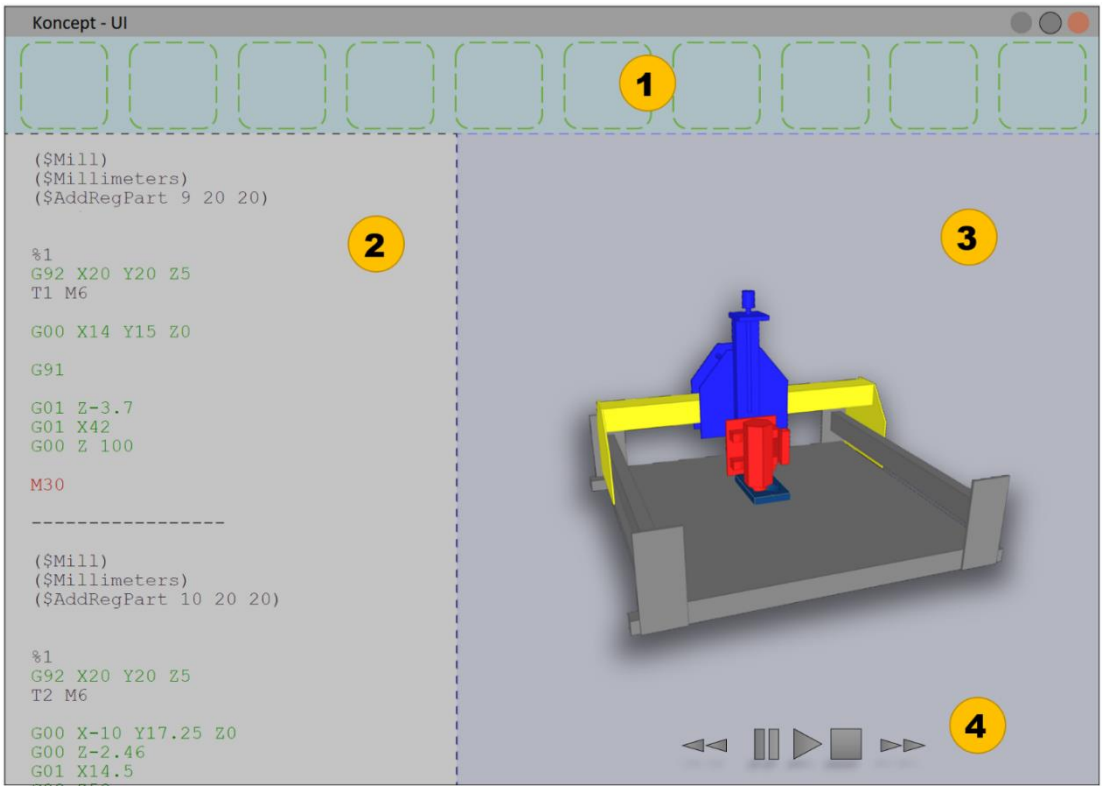
definira u XAML datotekama, čije sastavljanje često može predstavljati značajnu količinsku i vremensku većinu programiranja nove WPF aplikacije.

Pri izradi ove aplikacije glavni fokus biti će usmjeren na nekoliko odvojenih točki:

1. Razvoj korisničkog sučelja atraktivnog izgleda
2. Osiguravanje potrebnih funkcionalnosti za ispunjavanje zahtijeva obrazovne okoline
3. Povezivanje postojećih funkcija izvorne aplikacije sa novim korisničkim sučeljem.

Za vrijeme izrade samog sučelja, kreirana je shema za konceptualizaciju rasporeda i općeg izgleda ukupnog korisničkog sučelja (slika 4.3.). Kako je na shemi moguće i vidjeti, sučelje će se sastojati od:

1. Alatne trake koja će omogućavati pristup svim potrebnim alatima za manipulaciju tekstualnim (2) i grafičkim dijelom simulacije (3), ona će sadržavati alate za otvaranje i spremanje programskih datoteka, definiranje veličine radnog komada, definiranje promjera alata, opće postavke aplikacije, itd.
2. Prostora za pisanje G- kôda. Ovdje će se upisivati program kojeg će izvoditi simulacija (3).
3. Prostora za izvođenje simulacije sa 3D modelom stroja ili radnog komada.
4. Gumbi za pokretanje i zaustavljanje izvedbe simulacije.

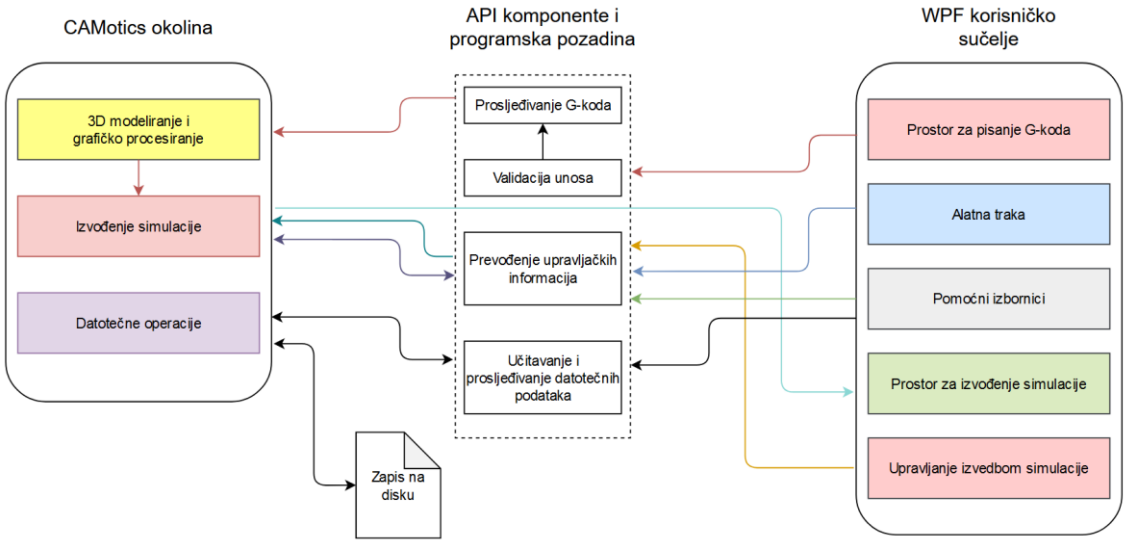


Slika 4.3. Konceptno sučelje aplikacije. 1- Alatna traka, 2- Prostor za pisanje G-kôda, 3- Prostor simulacije, 4- Gumbi za upravljanje izvedbom simulacije.

WPF će nam omogućiti da postojeće (šablonske) upravljačke elemente (izbornike, ikone, prozore, itd.) stvarnog korisničkog sučelja rasporedimo prema zamišljenom konceptu sa slike 4.3., zadržimo njihovu potpunu funkcionalnost te je upotpunimo dodatnim elementima potrebnim za naš specifičan oblik implementacije.

4.4. Principi izrade

Povezivanje odvojenih komponenti aplikacije (korisničkog sučelja i simulacijske okoline) postići će se omogućavanjem njihove komunikacije putem dodatne API (Application Programming Interface, eng. sučelje aplikacijskog programiranja) komponente koja će postojati između korisničkog sučelja i upravljačkih segmenata simulacijske okoline kao na slici 4.3.. API komponenta će služiti prevođenju ulaznih upravljačkih naredbi korisničkog sučelja (korisnika) u oblik kakav postojeća infrastruktura CAMotics aplikacije očekuje (modul prevođenja upravljačkih informacija), provjeravati će ispravnost unesenog G-kôda (modul validacije unosa) te taj kôd prosljeđivati simulacijskoj okolini. Dodatne zadaće API komponenti tiču se omogućavanja rada datotečnih operacija izvorne okoline te prenošenje radnih operacija alatne trake ka simulatoru.



Slika 4.4. Dijagram arhitekture i veza komponenti aplikacije.

Datotečne operacije u CAMotics okolini postojeća su komponenta te ju je nepotrebno iznova razvijati kroz programsku pozadinu novog sučelja, no kako i u novom sučelju mora biti moguće odabrati naredbu spremanja ili učitavanja datoteka sa pisanim G-kôd programom, a ta naredba mora i pokrenuti prikladne potprogramske strukture, potrebna je odvojena API komponenta (modul učitavanja i prosljeđivanja datotečnih podataka, slika 4.4.) koja će tekst upisan u prostoru

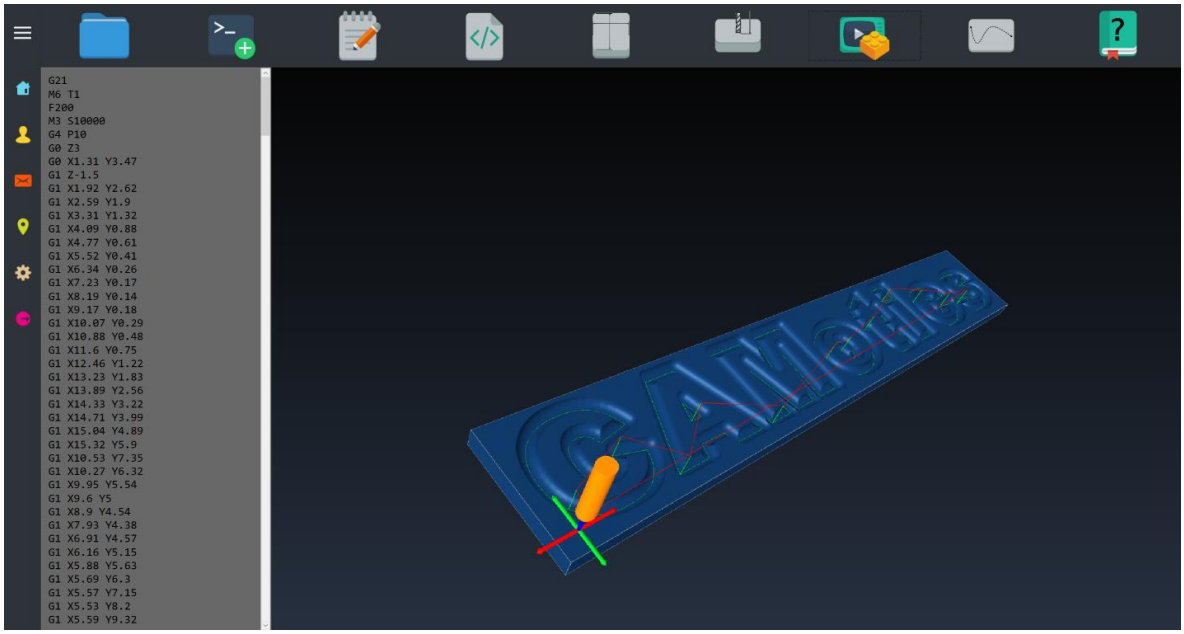
korisničkog sučelja proslijediti izvornoj okolini na daljnju obradu ili spremanje u obliku tekstualne datoteke. Komponenta „Pomoćni izbornici“ (slika 4.4.) odnosi se na dodatak alatnoj traci, a na konceptnom sučelju (slika 4.3.) predstavlja ga jedan od odjeljenja (kvadratni elementi iscrtani isprekidanom crtom) koje se nalazi na samom početku trake sa lijeve strane te je osmišljeno na način da se klikom na tu komponentu korisničkog sučelja prikaže padajući izbornik sa svim potrebnim dodatnim opcijama aplikacije koje nije potrebno prikazivati na alatnoj traci kako bi se u čim većoj mjeri smanjila pretrpanost zaslona korisnika i zadržala vizualna minimalnost i atraktivnost prikaza. Element pomoćnih izbornika omogućavati će naknadni razvoj funkcionalnosti te dodavanje povezanih opcija za podešavanje i konfiguraciju aplikacije. Ranije spomenuti prostor za izvođenje simulacije novog korisničkog sučelja prikazan i označen brojem 3 na slici 4.3. predstavlja tek prozor koji će sadržavati trodimenzionalni prikaz generiran kroz postojeće programske komponente za grafičko procesiranje izvorne CAMotics aplikacije, to znači da izrađena 3D simulacija radnog stroja, materijala i CNC kretnji prema uputama upisanog G-kôda neće biti prevođena kroz neku od API komponenti već će u novom sučelju izravno biti sadržan i dio izvornog korisničkog sučelja potrebnog da za prikaz simulacije. Takav pristup sastavljanja nove aplikacije ima kao negativnu posljedicu da usprkos većinskom korištenju novoga sučelja, i dalje postoji potreba zadržavanja i izvođenja dijela programa izvornog sučelja što može imati negativne posljedice za memorijsku učinkovitost čitave aplikacije. No ovakav oblik razvoja aplikacije odabran je kao kompromis u vidu smanjivanja vremenskih zahtjeva samoga razvoja ali i ukupne kompleksnosti zahvata.

4.5. Razvoj aplikacije

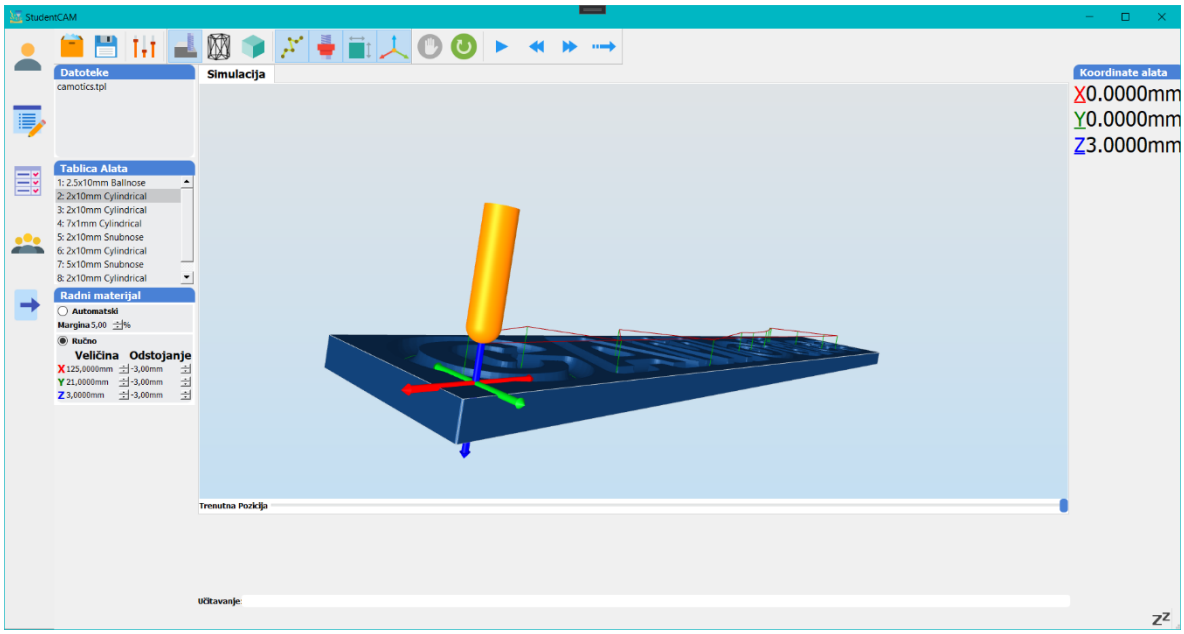
Aplikacija razvijena za potrebe ovoga rada prema projektu ocrtanom, između ostalih i u poglavlju 4.2. (Razrada projekta) okvirno predstavlja strukturu prve verzije računalnog alata koji će omogućiti zadovoljavanje zahtjeva predstavljenih u poglavlju 4.1. (Zahtjevi alata u odgoju i obrazovanju) kao i metodičke zahtjeve nastavnog pomagala o kojima je bilo riječ u poglavljima od 2.1. (Problematika prijenosa informacije i nastavna pomagala) do 3. (Posebnosti izučavanja CNC programiranja). Iako je sam proces razvoja aplikacije kroz programiranje pozadinskih funkcionalnosti, prilagodbe postojećih komponenti simulatora ili izrada grafičkih komponenti korisničkog sučelja (ikone, meniji, itd.) i dalje samo standardizirani način razvoja WPF aplikacije te sam po sebi ne iziskuje značajno detaljiziranje ili specificiranje pojedinih koraka, ipak je određene pojedinosti potrebno izdvojiti u svrhu približavanja novog korisničkog sučelja samom korisniku.

4.5.1. Korisničko sučelje

Na slici 4.5. prikazano je korisničko sučelje kakvo je izgledalo u prvoj verziji aplikacije. U ovom obliku odabrane su tamnije boje pogodne za dulje gledanje u zaslon računala kako bi se naprezanja očiju korisnika manju razinu. Prema projektu ta je mogućnost planirana kao promjenjiva prema preferencijama pojedinog korisnika. Konceptna verzija sučelja sa slike 4.5. dodatno je razvijana kroz naredne iteracije, te je svoj konačni izgled poprimio sa sučeljem na slici 4.6.



Slika 4.5. Konceptno korisničko sučelje nove aplikacije.



Slika 4.5. Konačni izgled korisničkog sučelja.

Na samom vrhu prozora aplikacije nalazi se glavni izbornik u obliku horizontalne trake s alatima od kojih je svaki alat ili grupa alata predstavljena prikladnom sličicom prema primjeru pomoćnog izbornika koja i u ovom slučaju služi kao dugme za pokretanje alata ili otvaranje drugih povezanih izbornika svakog od dostupnih alata (dodatno objašnjeni u tablici 4.3.).

Tablica 4.2. Funkcije pomoćnog izbornika.

Sličica	Značenje
	Profil korisnika
	Popis zadataka
	Popis ocjena
	Razred / grupa korisnika
	Prijava / odjava korisnika

Tablica 4.3. Neke od funkcija glavne alatne trake.

Sličica	Značenje
	Rad s datotekama
	Spremanje programa
	Postavke
	Prikaz alata
	Prikaz obrade materijala
	Prikaz prostornih vektora
	Pokretanje simulacije

5. DALJNJI RAZVOJ I UNAPRIJEĐENJA

Prvu verziju aplikacije potrebno je kroz testni period temeljito ispitati na ograničenom broju korisnika prije puštanja aplikacije u rad iz razloga što je neke (često nepredvidive) specifičnosti rada aplikacije jako teško ili čak nemoguće zamijetiti za vrijeme razvoja same aplikacije. Za početak, realnu statistiku stabilnosti rada kao i eventualne potrebne optimizacijske dodatke ili preinake bit će moguće odrediti tek kada aplikacija bude testirana na većem broju različitih računala, konfiguracija i kombinacija postavljenih korisničkih preferencija. Dodatno, učinkovitost kreiranog korisničkog sučelja u smislu zaokupljanja i fokusiranja pažnje te svođenja kognitivnog umora korisnika na najmanju moguću razinu za vrijeme duljeg korištenja aplikacije bit će vidljiva tek nakon provedenog ispitivanja stečenog mišljenja korisnika za vrijeme testnog perioda aplikacije. Verzija aplikacije u obliku u kakvom je predložena u ovom radu zadovoljavati će tek one osnovne upotrebne slučajeve u odgojno-obrazovnim okolinama. U svrhu što potpunijeg zadovoljavanja potreba ciljanih skupina korisnika potrebno je održavanje testne kampanje sa ciljem detaljne razrada potrebnih razvojnih unaprjeđenja, modifikacija postojećih funkcionalnosti te ispitivanje stabilnosti i prilagodljivosti aplikacije u radnom okruženju.

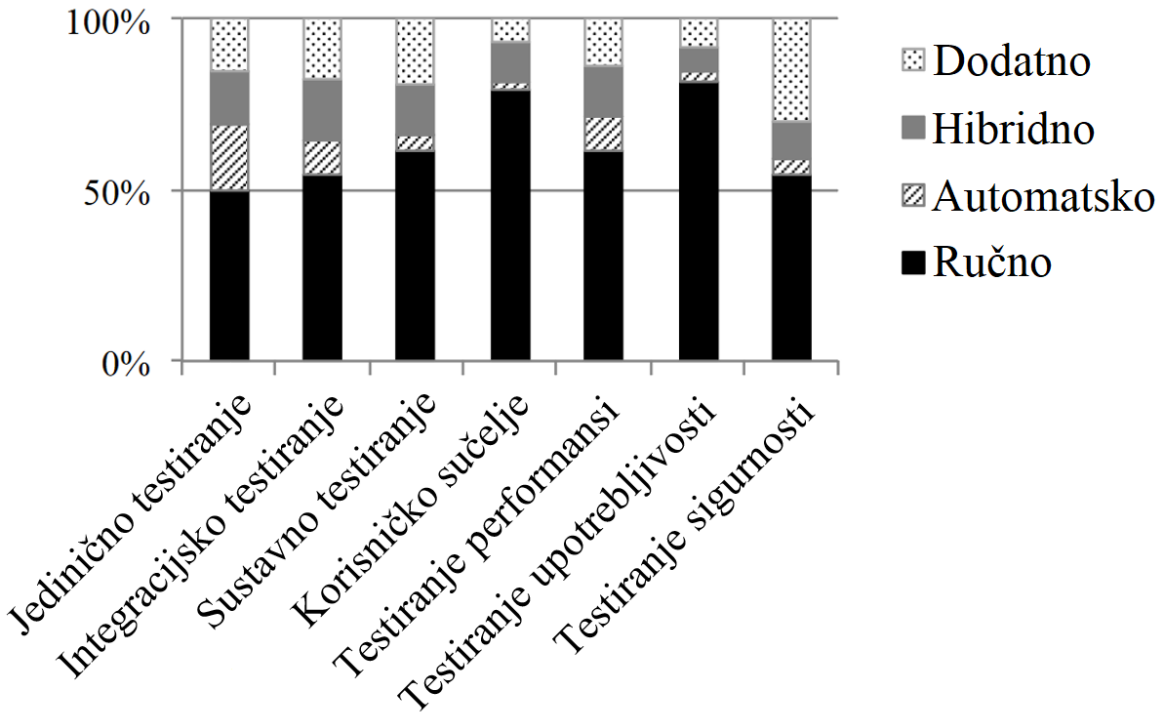
5.1. Testiranje funkcionalnosti aplikacije

Struktura testne kampanje čitave aplikacije mora biti unaprijed osmišljena i imati detaljno razrađeni plan svakog koraka ispitivanja kako bi bilo moguće uz što veću razinu učinkovitosti predvidjeti potencijalne problematične elemente te omogućiti njihovo pravovremeno rješavanje. U slijedećim ću poglavljima predstaviti primjereni način provedbe učinkovitog testiranja aplikacije dokumentiran kroz povezano znanstveno istraživanje (istraživanje je usmjereno na mobilne aplikacije, ali su korišteni principi primjenjivi za ispitivanje različitih računalnih alata), kao i načine modificiranja tog procesa te njegove primjene za specifičnosti ove aplikacije.

5.1.1. Proces testiranja

Struktura testne kampanje za ovu aplikaciju bit će temeljena na pronalascima kvalitativne studije istraživača sa sveučilišta British Columbia u Kanadi [17]. Na slici 5.1. moguće je vidjeti pojednostavljeni prikaz zaključaka donesenih u predstavljenom istraživanju u obliku grafa na kojem se uspoređuje postotak upotrebe određenog oblika ispitivanja komponenti ili sustava aplikacije sa načinom provedbe tog oblika ispitivanja, primjerice vrlo je primjetno kako se najveći

postotak čitave testne kampanje programa odvija upravo ručnim načinom ispitivanja što znači da iako postoje razne opcije korištenja automatizacije u tu svrhu, ipak se do



Slika 5.1. Uobičajene proporcije procesa testiranja aplikacije [17].

najveće pouzdanosti rezultata dolazi tek sistematskim ispitivanjem u realnim uvjetima rada sa stvarnim korisničkim subjektima. Spomenuto istraživanje potkrepljuje raniju tvrdnju u ovom radu kako uvijek postoji potreba za testiranjem aplikacije sa većim brojem realnih i aktivnih korisnika. Na grafu (slika 5.1.) su na horizontalnoj osi navedene operacije ispitivanja (samo one relevantne za slučaj ove aplikacije), dok je na vertikalnoj osi dana zastupljenost načina provedbe svake od operacija u postocima. Iz grafa se dadne iščitati sedam relevantnih operacija testiranja kao i četiri načina njihove provedbe. Od relevantnih operacije testiranja za specifičnu aplikaciju promatranu u ovom radu prisutnih na grafu, za potrebe dobivanja šire slike kroz ovaj oblik pred-produkcijskog testiranja dovoljno će biti osigurati testne rezultate iz nekoliko ispitnih operacija. Jedinično testiranje (unit testinig) individualnih komponenti programskog kôda u ovom stadiju razvoja neće biti potrebno iz razloga jer se većina novoga kôda sastoji od vizualnih elemenata u XAML formatiranim datotekama, a tek je nekolicina u obliku C# programskih naredbi, od toga još je manja zastupljenost definiranih programskih klasa ili metoda čije se funkcijske namjene mogu ispitati kroz individualiziranu testnu metodu. Iz toga je razloga u trenutnom stadiju razvoja jedinično testiranje moguće preskočiti bez bojazni od pojave većih problema, no također je bitno napomenuti da će jedinično testiranje programskih komponenti postati puno važnije u budućnosti

kada se budu implementirali neki od razvojnih ciljeva i kada poslovna logika aplikacije bude dodatno proširila. Integracijsko, sustavno i testiranje korisničko sučelja vjerojatno su najbitnije operacije ove testne kampanje, kako se glavne preinake izvorne aplikacije tiču upravo ovih programskih domena, pri provedbi testiranja posebnu je pažnju potrebno usmjeriti upravo njima. Na grafu posljednje prikazano testiranje sigurnosti u ovom je koraku razvoja također moguće zanemariti pošto aplikacija još nema razvijenu mrežnu komponentu (planirani razvojni korak za buduće verzije) te je njen utjecaj na sustav ograničen na lokalno računalo i samim time je minimalan ili nepostojeći. U konačnici, testiranje performansi i upotrebljivosti odnosi se na ranije spomenutu memorijsku i radnu učinkovitost čitave aplikacije te predstavljaju bitnu značajku opće razine uporabljivosti aplikacije.

5.1.2. Modifikacija i prilagodba procesa

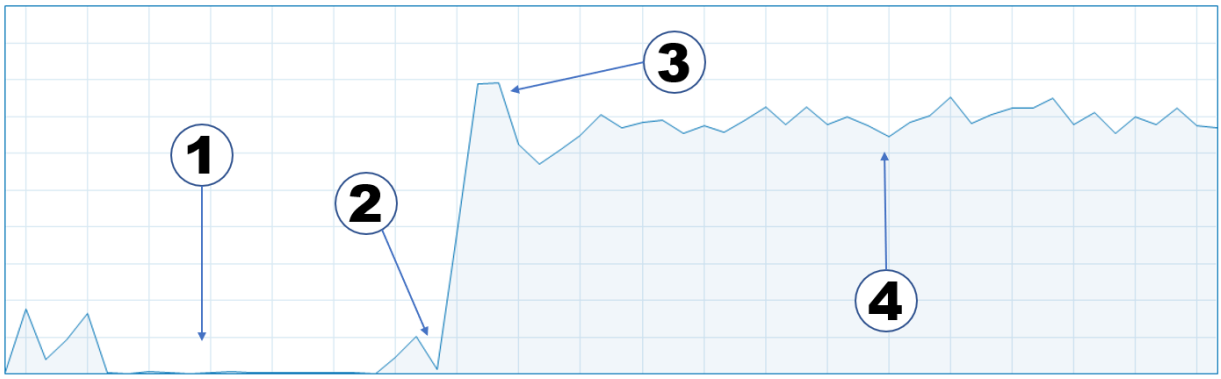
U svrhu provođenja testne kampanje u ograničenim vremenskim okvirima i bez nepotrebnog fokusiranja na manje kritične elemente ali kako bi se samo testiranje i dalje pridržavalo gore iscrtanih principa te i dalje u velikoj mjeri predstavljalo sveobuhvatno ispitivanje aplikacije, isto bi se trebalo sastojati u najmanju ruku od slijedećih izdvojenih koraka:

1. Ispitivanje stabilnosti aplikacije na najvišoj razini (funkcioniranje svih komponenti aplikacije u zajedničkom radu) (integracijsko testiranje).
2. Ispitivanje utjecaja na kontinuirani rad domaćina (utjecaj rada aplikacije na reaktivnost i prosječan odziv sustava na kojem je aplikacija pokrenuta).
3. Ispitivanje svih komponenti korisničkog sučelja i osiguravanje njihova zadovoljavajućeg funkcioniranja.
4. Dodatna ispitna operacija specifična za namjenu ove aplikacije odnosi se na propitivanje mišljenja i bilježenje savjeta korisnika relevantne stručnosti (nastavnici, CNC operateri, itd.).

Tek kada je testna kampanja u potpunosti provedena kroz sve njene najznačajnije korake, moguće je krenuti u daljnji razvoj dodatnih značajki i funkcionalnosti. Bitno je i nakon svake naredne značajnije preinake kôda aplikacije provesti temeljito testiranje nove programske komponente kao i postojećih komponenti na koje je moguć potencijalni utjecaj novog dijela aplikacije.

5.1.3. Provedba testiranja i analiza podataka

Kako je testiranje performansi jedini oblik predstavljenih operacija koje je moguće smisleno kvantificirati i grafički prikazati u ovome radu, u narednome je dijelu prikazan stvarni utjecaj upravo te komponente aplikacije. Testiranje se je provodilo na minimalno opterećenom Windows 10 operacijskom sustavu tek uz pokrenutih nekoliko dodatnih alata; Windows Task Manager koji će poslužiti za dobivanje podataka o trenutnim performansama računala, razvojna okolina Visual Studio 2019 kroz koju se pokreće sama aplikacija te Microsoft Office Excel u kojem će se bilježiti prikupljeni podaci. U početku je bitno napomenuti kako sustav domaćin ne posjeduje odvojenu grafičku karticu, to jest radi se o integriranom sustavu 3D akceleracije na Intelovom i7 procesoru starije generacije. Zbog spomenutog, a i zbog izraženog korištenja 3D grafičkih elemenata u simulacijskoj komponenti aplikacije, prvo ispitivanje performansi ticati će se općeg pregleda razine uporabe grafičkog procesora sustava domaćina. Na slici 5.2. prikazano je razina upotrebe GPU resursa za vrijeme besposlenog rada aplikacije (točka 1) i za vrijeme pokrenutog izvođenja simulacije (točka 4). Iz grafa je vidljivo kako postoji značajan skok u upotrebi GPU resursa nakon pokrenute simulacije (točka 2), ali je također moguće zamijetiti kako nakon



Slika 5.2. Razina upotrebe dostupnih GPU resursa za vrijeme rada aplikacije.

početnog skoka i naglog rasta (točka 3), potreba za resursima neznatno opade te se kroz naredno vrijeme izvođenja simulacije (područje oko točke 4) stabilizira bez kontinuiranog rasta ili iznenadnih skokova. U tablici 5.1. prikazani su rezultati dobiveni izračunom prosječnih vrijednosti upotrebe GPU resursa kroz 10 izvedenih testnih pokretanja simulacije. Prikazane vrijednosti dodatno dokazuju kako se vrijednosti utrošenosti GPU resursa zaista pokazuju stabilnim na nešto više od 65% prosječne upotrebe pri stvarnom radu. Uzimajući u obzir spomenute karakteristike domaćinskog sustava ovakvi rezultati svakako zadovoljavaju potrebe aplikacije.

Tablica 5.1. Prosječna razina upotrebe dostupnih GPU resursa.

Točka na grafu	Upotreba GPU resursa [%]
1	< 2
2	13,5
3	84
4	65

Nadalje, promatranjem prosječne opterećenosti sustava, to jest opterećenost procesora (CPU-a) na razini prosjeka svih njegovih logičkih jedinica i prosječne utrošenosti sistemske memorije (RAM) moći ćemo opaziti postojanje eventualne situacije „curenja memorije“ u programskom kôdu koja bi bila okarakterizirana prekomjernim korištenjem RAM memorije kao i povećavanjem njene uporabe kroz vrijeme, također indikator problematičnog izvođenja programa može biti i značajna opterećenost CPU-a sustava. Tablica 5.2. prikazuje usporedbu utjecaja pokretanja aplikacije u besposlenom načinu rada (pokrenuta aplikacija bez aktivnog izvođenja simulacije) i pozadinske opterećenosti domaćinskog sustava (opterećenost sustava bez pokretanja aplikacije). U iščitanim podacima je moguće vidjeti kako sam čin pokretanja aplikacije u stanju besposlenosti nema pokazatelja prekomjernog ispunjenja memorije ili opterećenosti sustava jer iako dolazi do primjetnog povećanja općeg opterećenja procesora, uporaba sistemske memorije poveća se tek za nešto manje od 1 GB što je očekivani i poželjan rezultat.

Tablica 5.2. Usporedba pozadinskog i radnog opterećenja sustava.

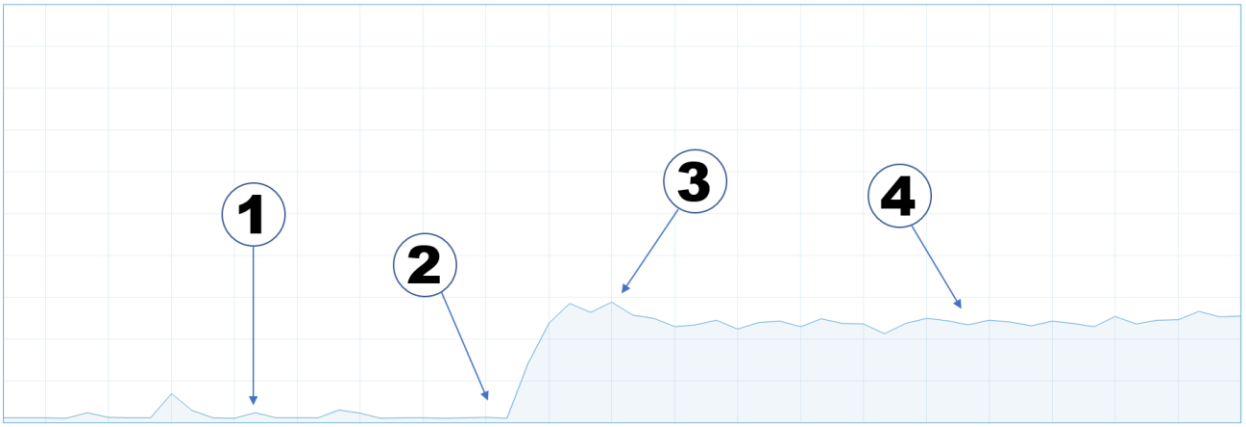
Opterećenje	Opće procesorsko [%]	Memorijsko [GB]
Pozadinsko	~ 1	6,8
Radno	24	7,7

Ako promotrimo razinu opterećenosti sustava s pogleda vremenski promjenjivih veličina nakon pokretanja same simulacije u aplikaciji dobivamo grafove prikazane na slici 5.3. i slici 5.4. Na grafovima su označene točke od značajnijeg interesa, te kao na slici 5.2., one označavaju različite korake u procesu pokretanja 3D simulacije u već pokrenutom radnom okruženju aplikacije. Svaka točka na grafu odgovara događajima prema tablici 5.3.

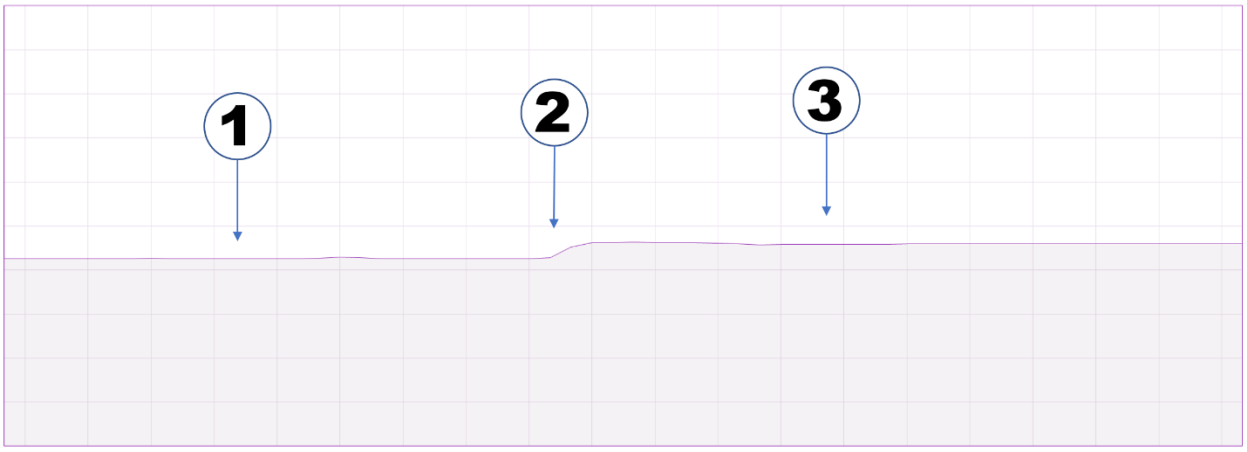
Tablica 5.3. Značaj točaka na grafu opterećenosti sustava

Točka na grafu	Značenje
1	Opterećenost u mirovanju
2	Točka pokretanja simulacije
3	Inicijalni odziv (skok)
4	Opterećenost u radu

Grafički prikazi opterećenosti nastali su mjerenjem prikazanih veličina kroz Windows Task Manager. Grafovi prikazuju ukupnu opterećenost čitavog sustava u prikazanom vremenskom razmaku od 60 sekundi. Svaki grafički prikaz (slike 5.3. i 5.4.) ima raspon vrijednosti koji obuhvaća iznose od 0 do 100% opterećenja, neovisno o kojoj se mjerenoj veličini radi (CPU ili RAM).



Slika 5.3. Razina opće opterećenosti procesora sustava.



Slika 5.4. Razina utrošenosti memorije sustava.

U pripadnoj tablici 5.4. prosječne su vrijednosti dobivene izračunom za razine opterećenosti kroz 10 izvedenih testnih pokretanja simulacije.

Tablica 5.4. Prosječna razina opterećenosti sustava

Točka na grafu	Opće CPU [%]	RAM [GB]
1	~ 1	6,8
2	~ 1.3	7
3	29,5	7,9
4	25	/

U podacima je vidljivo kao osim karakterističnog skoka između točaka 2 i 3 (prisutno i na grafu utrošenosti GPU resursa, slika 5.2.) koji odgovara trenutku pokretanja 3D simulacije te je očekivan, nema značajnije razlike u usporedbi sa opterećenošću sustava u mirovanju (tablica 5.2.). Analizom podataka se ne pojavljuje nikakva sumnja u problematičnost programa s obzirom na memorijsko upravljanje i procesorsko blokiranje. U konačnici, iz rezultata ove tablice prosječne analize kao i iz tablica 5.1. i 5.2. dade se sa značajnom sigurnošću zaključiti kako kreirana aplikacija u trenutnoj verziji ne predstavlja opasnost od pretjeranog opterećenja sustava te ne potrebuje nikakve trenutne optimizacije. Također, može se zaključiti i da je aplikacija pogodna kao osnova za daljnje razvijanje i nadograđivanje novim komponentama i funkcionalnostima.

5.2. Dodatni razvoj simulatora

Nakon provedenog testiranja rada aplikacije, slijedeći korak je planiranje daljnjeg razvoja sustava. Neke od najbitnijih funkcionalnosti koje bi još bilo potrebno razraditi te eventualno implementirati mogu biti:

1. Kreiranje i zadavanje identiteta korisnicima (korisničkih računa).
2. Podjela uloga korisnika na nastavnike i učenike/studente te kreiranje grupa ili razreda
3. Administracija razrednih grupa od strane korisnika-nastavnika.
4. Zadavanje vježbi/zadataka na razini razredne grupe ili individualnih korisnika-učenika.
5. Predaja zadatka korisnika-učenika na ocjenjivanje korisniku-nastavniku
6. Dodatne funkcionalnosti ispitnih zadataka i administrativnih detalja

Osim spomenutih značajki povezanih sa funkcionalnošću aplikacije u edukativnoj okolini i njenoj mrežnoj povezanosti, neke dodatne funkcije koje se tiču radnih principa i funkcionalnosti dostupnih dodatnih alata su:

1. Mogućnost odabira materijala kojeg se obrađuje.
2. Prikaz izračunate sile koja se javlja na oštrici za vrijeme rada stroja u obliku brojčane vrijednosti ili grafičkog prikaza (odvojeni dvodimenzionalni graf sa prikazanim iznosom sile na radnu oštricu kroz vrijeme)
3. Prikaz zona utjecaja topline uzrokovane radnim zagrijavanjem na materijal (Grafički prikaz izravno na modelu radnog komada, primjer na slici 2.5.)

5.3. Ispitivanje alata u realnim obrazovnim situacijama

Osim testnog perioda sa ograničenim brojem korisnika opisnog u prethodnim poglavljima, ciljanog na ispitivanje programskih i sustavnih karakteristika aplikacije (radna stabilnost, memorijska učinkovitost, itd.), za potpuno zadovoljavanje potreba realne odgojno-obrazovne okoline u sljedećem koraku testne kampanje bit će potrebno preciziranje dodatnih ciljeva razvojnog procesa kao i njihovo što potpunije usklađivanje sa odgojno-obrazovnim težnjama. Za potpuno ispitivanje kreirane aplikacije nužna je provedba probnog razdoblja rada aplikacije sa korisnicima koji odgovaraju populaciji ciljane skupine (profesori i učenici u obrazovnom okruženju), za vrijeme kojeg će korisnici moći zabilježiti sve nedostatke ili poželjna unaprjeđenja aplikacije. Takvim će se pristupom omogućiti odstranjivanje grešaka koje se otkriju, ali i osigurati potpuna prilagodba aplikacije realnom odgojno-obrazovnom okuženju.
Dodatna ispitna operacija specifična za namjenu ove aplikacije odnosi se na propitivanje mišljenja i bilježenje savjeta korisnika relevantne stručnosti.

6. METODIČKA RAZRADA TEME

6.1. Razlozi i motivacija za odabir teme

Kako je u tekstu i navedeno, svrha predstavljene analize i provedenog razvoja u sklopu ovoga rada bila je ukazati na nedostatak prikladnog alata za kvalitetno provođenje nastave ciljanog oblika, i uz to ponuditi alternativu trenutnom načinu njenog provođenja. Pri poučavanju programiranja CNC strojeva, predmetni nastavnici u srednjim ili osnovnim školama koji u sklopu svoje organizacije nemaju pristup kvalitetnim nastavnim pomagalima kojima bi polaznicima nastave mogli dočarati stvarno djelovanje CNC stroja, primorani su koristiti alate i računalne sustave ne-namijenjene toj svrsi. Simulatori ili slični alati koji omogućuju neki oblik vizualizacije rada stroja, uglavnom su programski paketi namijenjeni u najmanju ruku višim razinama edukacije profesionalnog djelatnika na CNC stroju. Korištenje takvih alata u edukativnom okruženju na razini viših razreda osnovne ili srednje škole nije idealno, te može imati negativni utjecaj na kvalitetu znanja kojeg će polaznik nastave steći. Računalni alat kreiran kroz programski razvoj u sklopu ovoga rada predstavlja uokvirenu aplikaciju namijenjenu uporabi prvenstveno u okruženju poput jednog školskog razreda. Nastavnik koji upravlja razredom ima mogućnost upravljanja virtualnom učionicom kroz koju je moguće polaznicima dodjeljivati zadatke, unositi komentare, ali i ocjenjivati radove. Polaznici nastave na isti način mogu imati uvid u dobivene zadatke, rokove za njihovo rješavanje, te povratne informacije na njihov rad. Uz te značajke, ali i pojednostavljeno grafičko sučelje simulatora, ova aplikacija predstavlja objedinjenu platformu čije korištenje omogućuje kvalitetno provođenje nastave na svim razinama izučavanja ove teme.

Širenjem upotrebe računalno upravljanih strojeva i sustava u sve većem broju različitih industrija i zanimanja, na isti će se način i izučavanje njihova načina korištenja morati prilagoditi sve većem broju učenika. Jedan od načina je upravo prilagodba alata i načina pristupa samom poučavanju. U tablici 6.1. definiran je konceptni izvedbeni nastavni program sa ciljanom upotrebom ovakve aplikacije.

6.2. Konceptni izvedbeni nastavni program

Koncept izvedbenog nastavnog programa za 3. razred srednjih strukovnih škola predstavljen je u tablici ispod.

Tablica 6.1. Koncept izvedbenog nastavnog programa

<table>
<tr><td>Nakon provedbe izvedbenog programa, učenik će moći:</td><td>1. Razlikovati karakteristike raznih računalno upravljanih strojeva
2. Opisati princip rada CNC stroja
3. Opisati svrhu osnovnih i složenih funkcija G- kôda
4. Samostalno izraditi program za upravljanje CNC glodalicom</td></tr>
<tr><td colspan="2">Razrada</td></tr>
<tr><td>Nastavne cjeline</td><td>Razrada nastavne teme</td></tr>
<tr><td>1. Uvod u računalno upravljane strojeve i sustave</td><td>• Primjena računalnog upravljanja u industriji
• Primjeri iz učeniku bliske okolice:
  • Strojevi za rezanje stakla
  • Zavarivanje automobila
  • Strojevi za sastavljanje tiskanih pločica
• Primjeri proizvoda koji su omogućeni zahvaljujući postojanju CNC strojeva
• Upoznavanje sa drugim računalno upravljanim sustavima
  • 3D printeri
  • Bespilotne letjelice
  • Itd.</td></tr>
<tr><td>2. Računalno upravljani sustavi</td><td>• Primjeri na tvornici automobila
  • Proizvodnje linije automatiziranog zavarivanja
  • Proizvodne linije za bojanje
  • Transportne linije</td></tr>
</table>

	• Principi rada računalno upravljanih sustava
	  • Programski ustavi upravljanja
	  • Sustavi za nadzor rada
	  • Sustavi za nadzor kvalitete
	• Načini upravljanja automatiziranim sustavima
	  •

3. Osnove računalno upravljanih strojeva	• Mehaničke komponente • Elektroničke komponente • Elektro-mehaničke komponente • Softverske komponente • Veza računalo – stroj • Korisnička sučelja

4. Upoznavanje osnova upravljanja CNC strojeva	• Interakcija stroj – računalo – radnik
	• Upravljana strojna oprema
	• Motori i aktuatori
	• Tipovi upravljane opreme
	  • Radni strojevi
	  • Tokarski strojevi
	  • Glodalice
	  • Laseri
	  • 3D printeri
	  • Robotske ruke
	  • Transportni sustavi
	  • Upravljanje pogonom bespilotnog sustava
	  • Zemni sustavi
	  • Zračni sustavi
	  • Pomorski sustavi

	• Više-osni sustavi
	  • Tro-osni sustavi
	  • Sustavi sa dodatnim osima
	• Upravljačko elektroničko sklopovlje (driveri)
	• Upravljanje koračnim i servo motorima

5. Prijenos instrukcije radnom stroju	• Interakcija sa upravljačkim sustavom
	  • Upravljanje korakom
	  • Upravljanje brzinom
	  • PWM signal
	• Slanje instrukcija
	  • Statička programska rješenja
	  • Mikro-kontroleri
	• Interaktivni računalni alati
	  • Izravno spajanje na sustav
	  • Simulatori
	• Simulacija CNC stroja
	• Upravljanje pomakom u sustavu

6. Osnove CNC programiranja	• Način programiranja CNC stroja • Karakteristike G-kôda • Specifičnosti i razlike pojedinih verzija programa • Osnove kretnji u koordinatnom sustavu • Pokretanje kreiranog programa • Osnovne funkcije • Upravljanje funkcijama CNC stroja • Programiranje rada CNC glodalice • Simuliranje kretnji stroja

7. Korištenje osnovnih naredbi G-kôda	• Upoznavanje alata za simulaciju rada CNC stroja
	• Osnove pomaka - G00 i G01
	• Linearna interpolacija pokreta
	• Apsolutno i relativno (inkrementalno) pozicioniranje
	  • G90
	  • G91
	• Postavljanje brzine pomaka i okretaja alata
	  • F naredba
	• Pomaci Z osi
	  • Pozitivni i negativni iznosi pomaka
	• Postavljanje referentnih točaka
	  • Postavke radnog volumena
	  • Postavke odstojanja alata
	• Postavke radnog materijala

8. Rad sa složenim funkcijama G-kôda	• Upoznavanje naprednih naredbi
	  • Radijalna interpolacija
	  • Praćenje odstojanja
	  • Interpolacija provrta
	  • Spiralna interpolacija navoja u provrtu
	• Kombinirano korištenje naprednih naredbi u simulatoru
	• Vježbanje na realnim zadacima

6.3. Metodička obrada sadržaja

U gore opisanim nastavnim cjelinama 7 i 8 obrađuje se sadržaj usko povezan sa temom ovoga rada. Obrada ovakvih tema zahtijeva primjenu metodičkih pristupa koji će omogućiti da učenici za vrijeme nastave čim više dođu u kontakt sa praktičnim produktima obrađivane teme, što je u ovom slučaju rad na simulatoru i pisanje CNC programa. Kao najprikladniji metodički pristupi za ovu temu izdvajaju se usmeno tumačenje, pokazivanje i prikazivanje. U nastavnoj jedinici „Korištenje osnovnih naredbi G-kôda“ nastavnik će frontalnim prezentiranjem teorije učenicima predstaviti temu i njene pojedinosti, nakon čega će metodom demonstracije pokazati način rada novog računalnog alata kojeg će koristiti. Izvođenje nastave odvija se u računalnoj učionici gdje će učenici imati mogućnost samostalno koristiti potrebne aplikacije.
U svrhu praćenja i vrednovanja rada svakog pojedinog polaznika nastave, nastavnik će periodički nadgledati rad učenika, ispitivati njihovu razinu usvojenosti sadržaja i način primjene naučene teorije. Pismenom provjerom u obliku rješavanja zadatka (pisanje G-kôda prema zadanom nacrtu) moguće je na kraju nastavne jedinice provjeriti ukupnu razinu stečenog znanja.

6.4. Priprema za nastavu

U prilogu 1. ovoga rada nalazi se ispunjeni obrazac metodičke pripreme jedne nastavne jedinice za obradu teme 8 – „Rad sa složenim funkcijama G-kôda“ iz konceptnog izvedbenog nastavnog plana kakav je opisan u tablici 6.1.

7. ZAKLJUČAK

Ovaj se je rad vodio početnom analizom potrebe razvoja specijaliziranog edukacijskog računalnog alata za simuliranje strojne CNC obrade. Predstavljena lepeza zahtjeva takvog alata sastoji se od mogućnosti prevođenja klasičnog G-kôda, grafičkog prikaza putanje alata, pa čak i 3D vizualizacije same strojne obrade materijala. Osim spomenutog, alat također mora zadovoljavati i brojne zahtjevne potrebe obrazovne okoline. Preliminarnom analizom tržišta vidljivo je kako oblik alata koji bi zadovoljio predstavljene tehničke zahtjeve, a u isto vrijeme posjedovao i sve značajke potrebne za njegovu kvalitetnu uporabu u odgojno-obrazovnom okruženju nije dostupan na tržištu. Tim zaključkom postavljena je osnova za razvoj nove aplikacije koja će upravo predstavljenu kombinaciju funkcionalnosti imati kao svoju ciljanu točku. Novi programski alat je osmišljen na način da svojim izgledom, funkcionalnostima i načinom njihove implementacije u što većoj mjeri podržava nastavnu okolinu u kojoj će se koristiti. Koncept aplikacije koja bi podržavala potrebne tehničke i edukativne karakteristike iz popisa zahtjeva, osmišljen je na način da se modifikacijom postojećih i razvojem novih mogućnosti neke aplikacije na tržištu, u kraće vrijeme postigne proizvod ciljane kvalitete. Aplikacija koja je odabrana u tu svrhu je alat CAMotics, tvrtke Cauldron Development. Sam odabir ove aplikacije bio je uvjetovan u najvećoj mjeri dostupnošću njenog programskog kôda, što je u slučaju CAMotics-a velika prednost zahvaljujući njenoj potpunoj otvorenosti kôda (Open Source). Programski razvoj u ovom radu opisuje modifikaciju postojećih značajki spomenute aplikacije, te kreaciju novog korisničkog sučelja sa implementiranim potrebnim dodatnim funkcionalnostima. Ovakvim je pristupom uz minimalnu modifikaciju jezgrenog kôda aplikacije omogućen razvoj proizvoda kompleksnih mogućnosti i kvalitete u ograničenom vremenskom razdoblju. Kroz rad je predstavljeno nekoliko mogućih primjena ovakvog načina razvoja, njegove prednosti i mane, te konkretni primjeri kompleksnosti i poteškoća koje se mogu pojaviti kroz razvoj ovakve programske implementacije. Rad se dotiče i procesa ispitivanja kreirane aplikacije, njenih performansi, a u konačnici i različitih mogućnosti unaprjeđivanja u budućnosti. Značajke same aplikacije krojene su prema mogućnostima postojećih aplikacija sličnih ciljanih karakteristika (edukativne aplikacije tehničko-tehnološke prirode) ali i prema ključnim idejama odgojno-obrazovne struke. Kako ovaj spoj mogućnosti ne posjeduje neka druga aplikacija slične namjene, a za njima svakako postoji potreba, može se zaključiti da postoji i prostor za ovakav tip aplikacije na tržištu, a samim time i potencijal za njeno plasiranje.

LITERATURA

[1] Andrew M., Cobb C.; Giampietro P.: Verbal ability and teacher effectiveness, 2005. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.971.7610&rep=rep1&type=pdf , s interneta 15. kolovoza 2021.

[2] Medley D, i dr.: Teacher Competence and Teacher Effectiveness, 1977. https://files.eric.ed.gov/fulltext/ED143629.pdf , s interneta 15. kolovoza 2021.

[3] Saroyan A.; Snell L.: Variations in lecturing styles. 1997. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.452.3224&rep=rep1&type=pdf , s interneta 16. kolovoza 2021.

[4] Horner R.: Encyclopedia of Behavior Modification and Cognitive Behavior Therapy, SAGE Publications, Portland 2005.

[5] Bubica N.; Mladenović M.; Boljat I.: Programiranje kao alat za razvoj apstraktnog mišljenja. 1997.https://bib.irb.hr/datoteka/702093.Programiranje_kao_alat_za_razvoj_apstraktnog_miljenja-CUC-zbornik.pdf , s interneta 16. kolovoza 2021.

[6] Kafai Y.; Burke Q.: Why children need to learn programming, MIT Press, Cambridge, 2014.

[7] Dewi S.; Hasanah H.: The comparison between visual learners and kinesthetic learners. 2020. https://ejournal.iainmadura.ac.id/index.php/panyonara/article/download/3151/1732 , s interneta 20. kolovoza 2021.

[8] Quille K.; Bergin S.: Does Scratch improve self-efficacy and performance when learning to program in C#?. 2020. https://www.researchgate.net/profile/Keith-Quille/publication/{}An-empirical-study.pdf , s interneta 20. kolovoza 2021.

[9] Azmia M., i dr. (2018). Ultimate Load Behaviour of Steel-Concrete Composite Plate Girders with Inclined Stiffeners. Kejuruteraan časopis SI 1(5) 2018: str. 59-70. https://www.ukm.my/jkukm/wp-content/uploads/2018/si1/5/9.pdf, s interneta 20. kolovoza 2021.

[10] Zaharija G.; Mladenović S.; Boljat I.: Introducing basic programming concepts to elementary school children. 2013. https://pdf.sciencedirectassets.com/277811/1-s2.0-S1877042813X00382/1-s{}gxrqb&type=client , s interneta 20. kolovoza 2021.

[11] National Instruments, Shiralkar M. (2012). LabVIEW Graphical Programming Course. https://cnx.org/exports/549b1c9b-5dcc-4909-bdeb-91565036f41f@4.6.pdf/labview-graphical-programming-course-4.6.pdf, s interneta 20. kolovoza 2021.

[12] Lee T. i dr.: Computational Thinking with Games in School Age Children. 2013 https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.258.7534&rep=rep1&type=pdf, s interneta 28. kolovoza 2021.

[13] Wcislik M. Tehničko Sveučilište Kielce, Poljska, IFAC publikacije, Programming of sequential systemin ladder diagram language. https://reader.elsevier.com/reader/sd/pii/S1474667017337114?token={}&originRegion=eu-west-1&originCreation=20210830235716, s interneta 28. kolovoza 2021.

[16] Smith B.; MacGregor J.: What is Collaborative Learning. 1992 https://www.researchgate.net/profile/Jean-Macgregor/publication/242282475_What_is_Collaborative_Learning/links/53f279060cf2bc0c40eaa8be/What-is-Collaborative-Learning.pdf, s interneta 28. kolovoza 2021.

[17] Joorabchi M. O., Mesbah A., Kruchten P. Sveučilište British Columbia. Real Challenges in Mobile App Development. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.724.2463&rep=rep1&type=pdf, s interneta 28. kolovoza 2021.

POPIS OZNAKA I KRATICA

CAD – Computer aided design , eng. računalno potpomognuti dizajn

CAM – Computer assisted manufacturing , eng. računalno potpomognuta proizvodnja

G-kôd – Programski jezik CNC programiranja

C, C++, C# – Klasični programski jezici

CNC – Computer numerical control , eng. računalno upravljanje strojem

CAMotics – Alat za simulaciju rada CNC stroja

WPF – Windows presentation foundation, skup programskih biblioteka za izradu UI elemenata

XAML – Oblik formatiranja tekstualne datoteke sa strukturiranim podacima

UI – User interface, eng. korisničko sučelje

UX – User experience, eng. korisničko iskustvo

API – Application programming interface, eng. sučelje aplikacijskog programiranja

CPU – Central processing unit, eng. procesor računala

GPU – Graphics processing unit, eng. grafički procesor

RAM – Random access memory, eng. sistemska memorija računala

SAŽETAK

Kroz ovaj su diplomski rad proučene glavne točke edukativnih računalnih alata tehničko-tehnološke prirode, njihova primjena u odgojno-obrazovnim okruženjima, te uloga koju imaju u razvoju znanja učenika. Iako se rad prvenstveno fokusira na razvoj novog alata za učenje CNC programiranja, u svrhu definiranja potrebnih karakteristika aplikacije, bilo je potrebno analizirati i niz alata iz različitih područja tehničke znanosti. Prema upoznatim značajkama proučenih alata, kreiran je koncept i razvojni plan za novu aplikaciju. Razvijena aplikacija služi se postojećim računalnim alatom otvorenog kôda za simulaciju rada CNC stroja, CAMotics. Najveći dio samog razvojnog procesa aplikacije otpao je na osmišljavanje i kreiranje novog grafičkog korisničkog sučelja kao i implementaciju spomenutih novih funkcionalnosti. Za razvoj sučelja korištena je Windows Presentation Foundation (WPF) platforma kojom je omogućeno kreiranje modernog sučelja jednostavnih crta i grafičkih koncepata uz poštivanje i nekih već prisutnih vizualnih odabira izvorne aplikacije. Pri samom kraju ovoga rada, pažnja se posvećuje procesu testiranja aplikacije, kao i osiguravanja kvalitete gotovog proizvoda. Predstavljaju se koncepti za kategorično ispitivanje mišljenja korisnika, te mogućnosti potencijalnog unaprjeđivanja. Novi računalni alat osmišljen je da u svojoj cijelosti što više odgovara potrebama suvremenog nastavnog procesa, te je kao takav među prvima u ovoj kategoriji nastavnih pomagala.

ABSTRACT

In this paper, we will examine the main focal points of educational computer applications in the technology domain, their use in the real educational environments, as well as their role in the formation of knowledge in students. Even though this paper's primary concern is the development of a new teaching tool for CNC programming, to define the needed features of the application, a close examination of the existing tools from a wide variety of technical applications is required. With the derived list of essential features for the new application, a conceptual and development plans were conceived. The new application makes use of an existing open-source tool for CNC machining simulation, CAMotics. The most significant part of the development process was taken up by the designing and construction phase of the new user interface, as well as the implementation of the mentioned new functionalities. For the development of the UI, Windows Presentation Foundation (WPF) framework was used, it enabled the creation of a modern interface with a simple design and visuals, all while adhering to some of the design ideas of the original application. At the very end of this paper, attention is turned to the application testing and quality assurance processes. In this last chapter, concepts for the examination of user experiences, as well as considerations for future updates and developments are presented. In the end, the produced application is designed in its entirety to adhere as much as possible to the contemporary standards of the teaching profession, and as such it is among the first in its category of computer tools.

Prilog 1. – Priprema sata

S V E U Č I L I Š T E U R I J E C I
FILOZOFSKI FAKULTET RIJEKA
ODSJEK ZA POLITEHNIKU

Ime i prezime: Paolo Šestan

P R I P R E M A
Z A I Z V O Đ E NJ E N A S T A V E

Škola: _____ Mjesto: Rijeka

Razred: 3. *Zanimanje: CNC operater

Nastavni predmet: CAD CAM tehnologije

Kompleks: Programiranje CNC strojeva

Metodička (nastavna) jedinica: Rad sa složenim funkcijama G-kôda

**Datum izvođenja: _____ **Mentor:_____

S A D R Ž A J N I P L A N

Podjela kompleksa na teme (vježbe, operacije)
(Uz svaku temu /vježbu, operaciju/ navedite broj nastavnih sati i podvucite onu koja se u pripremi obrađuje)

Redni broj	Naziv tema u kompleksu	Broj sati	
		teorija	vježbe
1.	Upoznavanje naprednih naredbi	1	0
2.	Kombiniranje naprednih naredbi i pomaka	1	1
3.	Vježba: Izrada gotovog rada	0	1

* Popunjava se ako se nastava održava u srednjoj strukovnoj školi
** Popunjava se ako obrazac služi za nastavnu praksu studenta

Karakter teme (vježbe, operacije) – metodičke jedinice

Obrada novih sadržaja, vježbanje i provjeravanje stečenog znanja.

PLAN VOĐENJA ORGANIZACIJE NASTAVNOG PROCESA

Cilj (svrha) obrade metodičke jedinice:
(Navedite ŠTO OD UČENIKA OČEKUJETE na kraju, nakon obrade nastavne građe, zbog čega se građa obrađuje)

Učenici će steći konkretna tehnička znanja o načinu sastavljanja CNC programa, naučiti će koristiti se temeljnim programskim načinom zaključivanja, spajanjem manjih dijelova programa (naredbi, funkcija) u zajedničku funkcionalnu cjelinu.

Ishodi učenja (postignuća koja učenik treba ostvariti za postizanje cilja):
(Posebno upišite koja znanja; koje vještine i umijeća, te koju razinu samostalnosti i odgovornosti učenik treba steći nakon obrade nastavne teme. Ishode formulirati jasno i jednoznačno kako bi se mogli nedvojbeno provjeriti evaluacijom.)

ZNANJE I RAZUMIJEVANJE *(obrazovna postignuća)*:

- navesti i opisati neke složene CNC operacije
- opisati način spajanja manjih operacija u veći, kontinuirani rad na CNC stroju
- precizno objasniti ponašanje simulacije stroja nakon pokrenutog programa

VJEŠTINE I UMIJEĆA *(funkcionalna postignuća)*:

- pravilni odabir konkretnih operacija za dani zadatak
- prema vlastitom navođenju odabrati točnu putanju za dani zadatak

SAMOSTALNOST I ODGOVORNOST *(odgojna postignuća)*:

- samostalno koristiti se dostupnom tehničkom dokumentacijom G-kôda, služiti se dobivenim alatima i aplikacijom za simulaciju rada CNC stroja.
- koristiti se točnim naredbama pri izvršavanju pomaka CNC stroja.

Organizacija nastavnog rada – artikulacija metodičke jedinice:

(Pregledno u tablicu upišite, zasebno za uvodni, glavni i završni dio u obliku teza: ŠTO se obrađuje – sadržaj, KAKO se obrađuje – metode rada i KOLIKO se obrađuje – trajanje nastavnog rada)

Dio sata	Faze rada i sadržaj	Metodičko oblikovanje	Vrijeme (min)
UVODNI DIO	• Ponavljanje naučene teorije iz prethodnog sata • Naglašavanje najvažnijih točaka koje će učenicima biti potrebne za rješavanje naredne vježbe	Metoda razgovora, demonstracije, dijalog s učenicima, Što je naučeno, što će tek naučiti	10
GLAVNI DIO	• Predstavljanje novoga zadatka, prikaz nacrta kojeg će morati rekreirati u aplikaciji za simuliranje rada CNC stroja • Samostalni rad učenika	Samostalno rješavanje zadatka	30
ZAVRŠNI DIO	• Pregledavanje i vrednovanje rada učenika	Pregled i evaluacija gotovih radova.	5

Posebna nastavna sredstva, pomagala i ostali materijalni uvjeti rada:

(Navedite što je konkretno potrebno i količine koje su potrebne. Izdvojite zasebno sredstva, pomagala i ostalo.)

Računalo s instaliranom aplikacijom za simuliranje rada CNC stroja.

Korelativne veze metodičke jedinice s ostalim predmetima i područjima:

(Navedite nastavni predmet i konkretno područje – temu.)

1. Grafička komunikacija – Kotiranje i označavanje.

Metodički oblici koji će se primjenjivati tijekom rada:

(Upišite na koji način ćete prezentirati sadržaj u pojedinom dijelu sata ili nastavnog rada)

Uvodni dio:
Metoda razgovora – dijalog s učenicima, ponavljanje naučenog s prethodnog sata.
Metoda demonstracije – upoznavanje učenika sa konceptima koje će koristiti pri rješavanju zadatka

Glavni dio:
Samostalni rad učenika.

Završni dio:
Pregled učeničkih radova i ispitivanje njihovih zaključaka.

Izvori za pripremanje nastavnika:

(Literatura s potpunim bibliografskim podacima, prikupljenim podacima, uvidom u konkretnu praksu i drugo.)

Krunoslav Curić - Programiranje CNC glodalica i tokarilica, udžbenik za strukovne škole

Izvori za pripremanje učenika:

(Udžbenik ili/i pomoćna literatura s potpunim bibliografskim podacima i sl.)

Mladen Bošnjaković - Programiranje CNC strojeva

TIJEK IZVOĐENJA NASTAVE – NASTAVNI RAD

(Detaljna razrada teza iz tablice artikulacije – napisati onako kako će se izvoditi pred učenicima – "scenarij" nastavnog procesa)[*]

UVODNI DIO

Na samom početku uvodnog djela nastavnog sata, nastavnik pozdravlja učenike te izvršava potrebne administrativne aktivnosti (evidencija prisutnosti, itd.). U slijedećem koraku, nastavnik s učenicima ponavlja prethodno obrađivano gradivo te ih postepeno uvodi u temu današnje nastavne jedinice.

U vremenu predviđenom za uvod, sa nastavnikove strane je najbitnije učenicima precizno predstaviti temu kojom će se baviti za vrijeme trajanja nastavne jedinice. Nastavnik uvod započinje ponavljanjem najbitnijih značajki naučenih na prethodnom satu, a koje će biti neophodne za uspješno savladavanje zadatka koji im predstoji.

U nastavku je scenarij izvođenja sata pisan u prvom licu. Podebljanim slovom N označen je govor nastavnika, a podebljanim slovom U označeni su mogući odgovori ili pitanja učenika.

N: Pozdravljam sve još jednom, danas ćemo održati sat nastave čija će glavni cilj biti odrađivanje jedne vježbe koja je po rasporedu po vašem nastavnom planu i programu. Vjerujem da da ćemo dobro surađivati i da će vam ovo predavanje i vježba koja će uslijediti bit zanimljivi, vježba i sam sadržaj predavanja nije kompliciran ali je vrlo bitno da smo svi usredotočeni i da pratimo prezentaciju i opise zadatka koji će biti prikazani. Ako vam u bilo kojem trenutku kroz tok ovog predavanja nešto ne bude potpuno jasno ili postoje bilo kakve druge nedoumice u vezi sadržaja ili bilo čega drugoga, slobodno podignite ruku i postavite pitanje.
Ako nema nikakvih pitanja u vezi ovoga dijela, možemo nastaviti na predstavljanje teme ovoga predavanja.

- Ovdje će nastavnik odgovarati na eventualna pitanja učenika

Predstavljanje sadržaja predavanja i vježbe koji slijede

N: Kako sam već najavio, trajanje ovoga predavanja je naredni školski sata za vrijeme kojeg je planirano naučiti nešto više o radu sa složenim funkcijama u G-kôdu.
Za početak ćemo objasniti što sve spada u složene funkcije G-kôda, kada ih koristiti i zašto su vrlo važne u jako velikom broju CNC programa i radu strojeva. Također ćemo spomenuti kako se veći broj jednostavnih i složenih funkcija ili naredbi može kombinirati u jednu zajedničku cjelinu kako bismo takvim relativno jednostavnim programom mogli riješiti vrlo kompleksne zadatke i izraditi neki gotov proizvod.
Većina ovoga sata će nam se svesti na rješavanje ove već spomenute vježbe, njome ćemo pokušati na jedan intuitivan način prikazati neke od ovih principa koje ćemo spomenuti kroz nekoliko teorijskih detalja koji slijede u nastavku.
Za vrijeme rješavanja vježbe, dobiti ćete zadatak kojeg ćete samostalno rješavati koristeći se metodama koje ćemo sada proći.
Zadaci će se na kraju ocjenjivati, pa vas molim za pažnju i pitanja ukoliko bude nejasnoća.

[*] Uložite nove stranice papira, odnosno onoliko koliko zahtijeva tekst "scenarija".

N: Dakle, tema kojom ovoga sata su funkcije ili naredbe u G-kôdu.
Želio bih započeti sa kratkim ispitivanjem vašeg poznavanja ove teme, pa ako se slažete volio bih da mi odgovorite na par uvodnih pitanja koja sam pripremio za vas.

N: Za početak, ako radimo na CNC glodalici, koju naredbu možemo koristiti za pomak u obliku ravne crte?

U: Koristimo G00 ili G01.

N: I što još uz naredbu moramo definirati osim nje same? Kako definiramo smjer pomaka?

U: Dodavanjem X i Y koordinata uz naredbu.

N: A kako bi stroju naredili pomak po visini? Kako postavimo dubinu reza?

U: Sa Z koordinatom.

N: Za početak je vrlo važno zapamtiti razliku između G00 i G01. G00 je na svakom stroju rezervirana za pomak brzinom slobodnog hoda, to jest za hod bez rezanja, G00 možemo koristiti samo kada oštrica radnog alata sigurno neće dotaknuti radni materijal. Zašto je to važno?

U: Zato jer alat ne može podnijeti opterećenja rezanja pri tolikoj brzini.

N: Uredu, krenimo onda na novo gradivo. Zna li netko kako bismo u G-kôdu mogli narediti CNC stroju da u materijalu izreže plitki žlijeb u obliku, na primjer polukruga?

- *Učenici u ovom slučaju vjerojatno neće znati pravilno odgovoriti na pitanje jer im je ovo prvi susret sa takvim zadatkom.*

N: Sada ćemo naučiti primjenu nečeg što se zove „Radijalna interpolacija“, je li nekom poznata riječ „interpolacija“?

- *Učenici se ovdje mogu prisjećati da su interpolaciju spominjali na prethodnim satovima sa linearnom interpolacijom ili na satu matematike.*

N: U slučaju radijalne interpolacije, koristiti ćemo se, za početak dvjema koordinatama X i Y, te radijusom krivulje koju iscrtavamo.

N: Kada želimo u G-kôdu programirati putanju koja prati kružnicu, u tu ćemo svrhu koristiti naredbe G02 ili G03. Te se naredbe razlikuju samo po tome što naredbom G02 kružnicu iscrtavamo u smjeru kazaljke na satu, dok naredbom G03 istu iscrtavamo u smjeru suprotnom kazaljki na satu.

- *Nastavnik ovaj dio skicira na ploči*

N: Kao i u slučaju naredbe G01, i za nove je naredbe potrebno definirati krajnju točku luka, no osim krajnje točke, potrebno je definirati i radijus pomoću kojeg će se luk moći iscrtati.

N: U krajnju će točku alat stroja doći na isti način kao i kod G01, samo će se putem, umjesto ravne crte, pomaknuti po kružnici sa radijusom kojeg se definira na isti način kao i koordinate na kraju naredbe ali sa slovom R.

N: Je li svima jasno kako iscrtavamo luk u G-kôdu?

- *U ovom je djelu sata potrebno učenicima razjasniti sve nejasnoće koje se eventualno mogu pojaviti oko prethodno opisanog gradiva.*

N: Može li mi sada netko pokušati reći što bi se trebalo dogoditi ako u naredbu za iscrtavanje luka ili kružnice ubacimo i treću koordinatu Z?

U: Alat stroja će se pomaknuti gore ili dolje.

N: Ako pri naredbi za iscrtavanje luka dodamo i treću dimenziju, alat će se u smjeru te treće osi Z pomicati u isto vrijeme dok se pomiče i u smjeru luka zadanog naredbom G02 ili G03. Dakle, alat će pomicati putanjom koja je u bočnom presjeku ravna crta, ali u tlocrtu je i dalje luk.

N: Točna putanja kojom će se vrh alata stroja pomaknuti zove se heliks ili spirala. Takvom ćemo se trodimenzionalnom krivuljom koristiti za glodanje provrta ili u naprednim operacijama kao što je na primjer urezivanje navoja.

N: Sada ću vam na ploči objasniti kako funkcioniraju naredbe za kružni luk, molim vas da pratite i pripremite pitanja da razjasnimo nejasnoće. Nakon demonstracije ćete vi napraviti svoju vježbu.

N: Dakle, ako želimo napraviti provrt pomoću glodala manjeg od potrebne veličine gotovog provrta, možemo se poslužiti radijalnom interpolacijom sa dodanom trećom osi Z.

- *Nastavnik ovaj dio skicira na ploči*

N: Započnemo sa pravocrtnim pomakom do mjesta na kojem moramo napraviti provrt, zatim naredbom G02 započnemo polukružni luk u smjeru kazaljke na satu jednako kao i u prethodnom primjeru, no ovoga puta dodamo i tu treću koordinatu. Za vrijednost Z koordinate postavimo neki negativni broj, ovisi koliku dubinu reza želimo postići za prvu polovicu kružnice. Obično nećemo ulaziti više od nekoliko milimetara u radni komad za svaku polovicu kružnice.

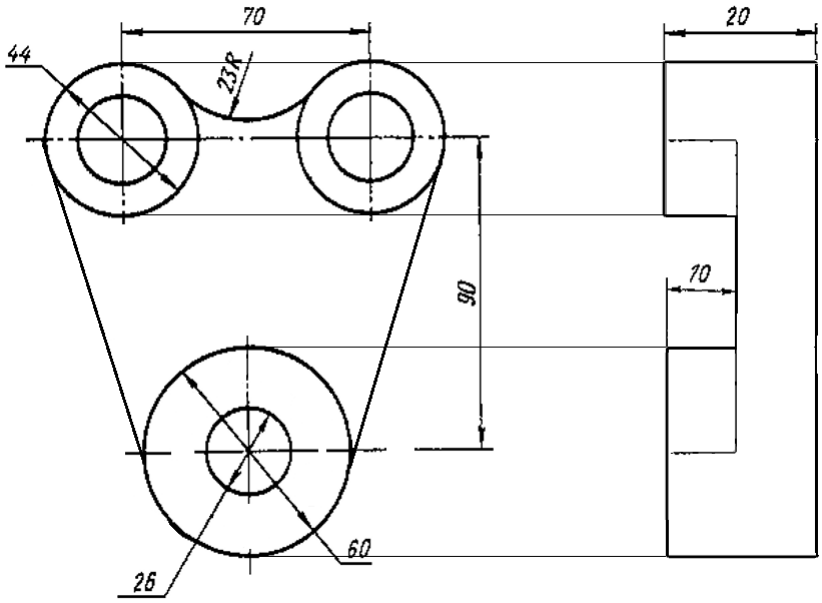
N: Je li svima jasan ovaj dio?

- *Nastavnik će ovdje po potrebi odgovoriti na eventualna pitanja.*

N: Uredu, ako više nema pitanja možemo krenuti na rješavanje današnje vježbe.

PRAKTIČNI DIO

U ovome djelu sata nastavnih učenicima najavljuje početak vježbe, te ih upućuje da pokrenu aplikaciju za simuliranje rada CNC stroja. Nastavnik razredu šalje vježbu na uvid, te upozori učenike da ju još ne krenu rješavati. Na slici 1. prikazana je vježba koju učenici dobivaju u zadatku.



Slika 1. Vježba koju će učenici izrađivati na satu, veličine su prikazane u mm.

Objašnjavanje zadatka vježbe

Nastavnik učenicima objašnjava kako je današnja vježba relativno jednostavna te da će im biti potrebno samo poznavanje osnovnih elemenata sa prethodnog sata i ono što su naučili u uvodu ovoga sata. Nastavnik prema slici 1. pojašnjava što se od njih traži.

N: Za uspješno rješavanje vježbe morati ćete se koristiti radijalnom interpolacijom kako biste postigli iscrtavanje vanjskih elemenata predmeta, ali i spiralnom interpolacijom za izradu provrta.

N: Na predmetu u zadatku svi su provrti jednaki, promjera 26mm, ali od alata, za čitav zadatak možete koristiti samo obično glodalo promjera 20 mm.

N: Bitno je pri izradi provrta koristiti spiralnu interpolaciju, nikako običnu radijalnu interpolaciju pa stepenasto spuštati Z os.

N: Ima li nekih pitanja?

- *U ovom djelu nastavnik odgovara na pitanja učenika prije početka njihovog samostalnog rada.*

N: Ako nema drugih pitanja, možemo početi sa rješavanjem zadatka.

Samostalni rad učenika

Kada je nastavnik uvjeren da su većina učenika shvatila zadatak i kada ne bude više pitanja, učenici mogu započeti sa rješavanjem zadatka.
Za vrijeme rješavanja učenici ne smiju međusobno komunicirati, ali mogu nastavnika upitati bilo koji dio koji im je nejasan. Nastavnik će čitavom razredu nastojati odgovoriti što potpunije kako bi i učenici koji nisu u potpunosti shvatili teorijski dio, mogli riješiti zadatak.

Praćenje procesa rada i zalaganja učenika

Nastavnik je dužan nadzirati čitav proces rješavanja zadatka na način da obilaskom razreda, ukoliko je potrebno usmjerava učenike koji imaju problema pri rješavaju, i navodi ih na pravi put ka točnom rješenju. Nastavnik mora poticati učenike da postavljaju pitanja ako im je nešto nejasno i stvoriti ugodnu radnu atmosferu. Također je potrebno vremenski kontrolirati rad kako bi čim veći broj učenika bio gotov u vremenskim okvirima nastavne jedinice. Nastavnik također mora savjetovati učenike kako bi mogli biti brži i uspješniji u zadatku.

ZAVRŠNI DIO

Evaluacija i vrednovanje rada učenika:

Nastavnik informira učenike kako je dozvoljeno vrijeme za rješavanje vježbe prošlo, te da ukoliko netko od učenika ne preda svoj rad sada, to neće biti u mogućnosti učiniti kasnije. Nastavnik će nakon provedene analize rada svakog učenika, prenijeti povratnu informaciju o uspješnosti rješavanja njihova zadatka. Konačna ocjena sastojati će se od sljedećih elemenata:

- Je li učenik u potpunosti dovršio zadatak u danom vremenu
- Je li učenik postavljao pitanja i interesirao se za vježbu
- Kako je učenik pratio upute nastavnika
- Je li učenik ispravno koristio naučene principe iz uvoda u sat kako bi dobio konačni rezultat

Nakon vrednovanja i davanja povratnih informacija učenicima na njihove radove kratko ponavljamo što su sve danas na satu naučili o naprednijim mogućnostima u G-kôdu te o njihovim zapažanjima kako se napisani program ponaša pri spajanju većeg broja kompleksnijih i jednostavnih operacija u jednu cjelinu.

Izgled ploče

(Skicirati potpuni izgled ploče nakon obrađene teme /naslov, skice, crteži, tekst/ .)