ASAAI
Artificial Sense of Aesthetic Assisting Illustration

seminarska naloga pri predmetu UISP

Avtoriji: Klemen Pernuš
Marko Đukić
Miha Žagar

Mentor: Aleks Jakulin

Kazalo

Uvod

V okviru predmeta umetna inteligenca in simbolično programiranje smo za seminarsko nalogo izbrali “prepoznavanje risarskih gibov s strojnim učenjem”. Kasneje smo jo preimenovali v “Artificial Sense of Aesthetic Assisting Illustration - ASAAI”, kar v prostem prevodu pomeni umetni čut za estetsko pomoč pri risanju. Poglavitni namen naloge je bil napisati program, s katerim bi predstavili možnost uporabe tabličnega računalnika kot orodja za hitro risanje raznih diagramov.

Cilj naše naloge je bil torej napisati preprost program za risanje osnovnih grafičnih elementov (črta, kvadrat, krog, ...), ki pa bi imel drugačen princip risanja, kot ga imajo podobni programi. Pri klasičnih risarskih programih mora uporabnik najprej povedati, kaj hoče narisati (s klikom na določeno ikono), šele nato pa lahko nariše željen lik. Pri našem programu pa naj bi uporabnik prostoročno narisal željen lik, pri čemer naj bi program iz tega vnosa znal ugotoviti, kaj je uporabnik želel narisati, vnos spremeniti v željen lik in ga pravilno umestiti med že narisane like.

Problemi in rešitve

Problem prepoznave

Pri izdelavi programa smo naleteli na nekaj problemov, katere smo bolj ali manj uspešno rešili.

Prvi problem je bil kako sploh prepoznavati grafične elemete. Pri tem smo imeli dve možni rešitvi.
Prva - lažja a precej zaprta rešitev je bila, da bi uporabnikov vnos ovrednotili z nekimi vnaprej izbranimi parametri, nato pa bi te parametre primerjali s parametri raznih grafičnih elemetov, ki bi jih program imel v svoji bazi. Na podlagi tega primerjanja bi se nato odločili za lik, ki se najbolje ujema z uporabnikovim vnosom. Taka rešitev je razmeroma enostavna, saj je potrebno najti le zadostno število parametrov, s katerimi bi z dovolj veliko verjetnostjo lahko določili pravilen lik. Vendar pa je ta rešitev precej omejujoča, saj program zna pravilno prepoznati le tiste elemente, katere ima v svoji bazi.
Druga rešitev problema prepoznavanja vnosa pa je, da iz vnosa znamo povedati, kakšen tip lika je uporabnik narisal - enostaven (ena sama ravna črta) ali bolj kompleksen (lomljena črta ali krivulja) - ter nato ta lik pravilno umestiti v kontekst že narisanih likov.

Odločili smo se za drugo rešitev, pri kateri pa smo se omejili samo na prepoznavo ravnih in lomljenih črt. Problem risanja krivulj smo pustili za kasnejšo obravnavo.
Problem prepoznave ravne črte smo rešili na sledeč način: program najprej razlikuje med ravno in lomljeno črto. To naredi tako, da iz uporabnikovega vnosa (t. i. sledi) izračuna premico po metodi linearne regresije. Nato izračuna povprečno oddaljenost točk sledi do dobljene premice. Če je povprečna oddaljenost pod določeno mejo (ki se jo program priuči ali pa jo nastavi uporabnik), se program odloči, da je uporabnik hotel narisati ravno črto; v nasprotnem primeru (če je povprečna oddaljenost nad določeno mejo) pa se odloči za lomljeno črto.
Konstrukcija ravne črte je sledeča: izračunamo centralno točko sledi, dolžino, ki jo bo črta imela (razdalja med najbolj oddaljenima točkama sledi) ter smerni vektor premice, dobljene z linearno regresijo. Postavimo se v centralno točko, ji prištejemo polovico smernega vektorja ter polovico vektorja, ki je nasproten smernemu. Dobljeni dve točki sta oglišči iskane ravne črte.
Postopek prepoznave lomljene črte sestoji iz prepoznavanja ravne črte, pri čemer pa na vsakem koraku točko iz vnosa, ki je najbolj oddaljena od črte, označimo kot novo robno točko in nato rekurzivno ponovimo postopek prepoznave med začetno in robno točko ter med robno in končno točko. Postopek deluje dobro, saj oglišča izbira na pravilen način.
Pred tem postopkom smo uporabljali postopek, ki je deloval obratno – najprej smo izračunali vsa možna oglišča (točke sledi smo zapisali v polarni obliki ter spremljali radij), potem pa smo jih

izločali. Vendar pa ta postopek ni dal pričakovanih rezultatov, saj je izločal napačna oglišča.
Postopek, ki smo ga prav tako opustili, je bilo prepoznavanje s pomočjo "Houghove transformiranke". Vendar pa se je tudi ta postopek izkazal kot neuspešen, saj zaradi človeškega faktorja (uporabnik riše s prosto roko) nismo mogli zadovoljivo prepoznati lomljenih črt.

Problem umestitve

Ko zaključimo s prepoznavo vnosa, imamo namesto prostoročne risbe množico ravnih črt, katero je potrebno pravilno umestiti v množico že narisanih črt. To pa nas pripelje do drugega problema, na katerega smo naleteli pri našem programu. Umestitev posamezne črte v množico že narisanih črt nam daje veliko število možnosti, kako to črto postaviti. Zato smo vpeljali pojem `omejitev´.

Program naj bi pomagal pri risanju pravilnih likov, kot so pravokotnik, kvadrat, trikotnik, ipd., zato smo pri prepoznavanju uporabnikovih vnosov vpeljali naslednje omejitve:

- vodoravnost in navpičnost črt,
- zaokroževanje dolžin črt,
- vzporednost dveh črt pri pogoju, da sta enake dolžine,
- stikanje dveh črt v vozliščih.

Te omejitve so zadostovale pri risanju zgoraj omenjenih likov. Njihovo uveljavljanje program izvaja na črtah narisanega lika med sabo in pri nadaljevanju risanja nekega lika (v kolikor smo začeli risanje iz nekega vozlišča že narisanega lika). Povedano velja za like, narisane z eno potezo. V kolikor rišemo vsako črto posebej, uveljavljanje omejitev izvedemo na vseh črtah na zaslonu.
Uveljavljanje poteka po naslednjem principu:

- črte se umešča glede na vrstni red pri risanju (po vrsti od začetka),
- pri prvi črti se samo preveri, ali ji ustreza omejitev vodoravnosti/navpičnosti in se jo umesti,
- za vsako naslednjo črto poleg vod./navp. preverimo tudi odnose z drugimi črtami (t.j. ostale omejitve: stikanje, enaka dolžina, vzporednost – v tem vrstnem redu) in jo umestimo,
- izjemi:
    - eno samo na novo narisano črto se primerja z vsemi že narisanimi,
    - pri nadaljevanju risanja lika se nove črte primerja s črtami narisanega lika.

Izkazalo se je, da z večjim številom omejitev raste tudi kompleksnost uveljavljanja omejitev in ugotavljanja razmerja med njimi (prednost ene omejitve pred drugo, ipd.). Zaradi tega smo se tudi

odločili za bolj lokalno (znotraj lika), ne pa globalno različico uveljavljanja omejitev (glede na vse narisane like na zaslonu).

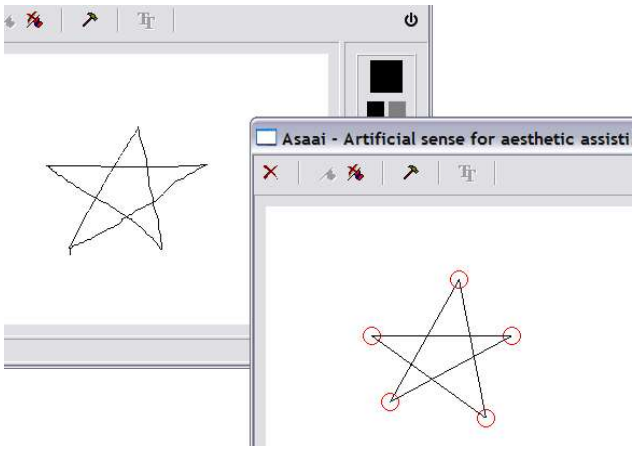
Problem uporabniškega vmesnika

Tretji problem pri izvedbi naše naloge pa je bil problem uporabniškega vmesnika. Do tega problema je prišlo, ker je bil osnovni namen progama uporaba le-tega na tabličnem računalniku. Pri tabličnem računalniku celotna interakcija uporabnika z računalnikom poteka preko vnosnega peresa. Dostop do tistih nekaj tipk, ki sicer so na tabličen računalniku, pa je precej težaven, saj se do njih dostopa z isto roko, s katero se tablični računalnik drži! Tako nam za upravljanje s programom v bistvu ostane le vnosno pero. Pero je zelo podobno miški pri namiznih računalnikih, saj ima dva gumba ter pozna klik in dvojni klik. Iz peresa tako dobimo štiri različne dogodke, s katerimi skoraj v celoti uporavljamo s programom.

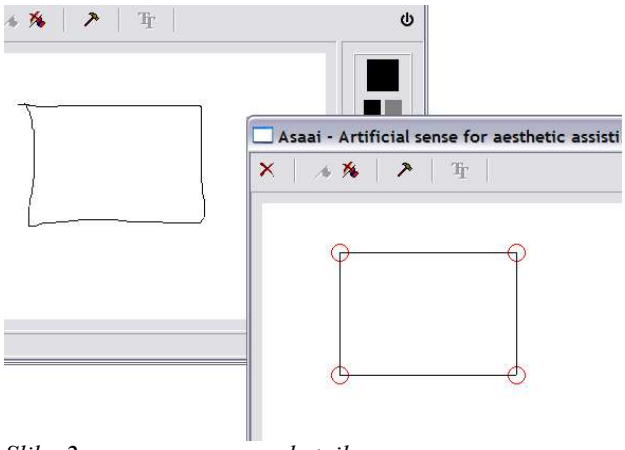
Imeli smo željo napisati tak uporabniški vmesnik, ki bi uporabniku omogočal risanje, izbiranje, brisanje ter premikanje elementov, brisanje omejitev ter dodajanje teksta posameznim črtam. Za risanje nismo imeli prav veliko izbire, saj je edina "naravna" izbira pisanje s peresom, čemur ustreza pritisk na levi miškin gumb. Naslednja stvar je bila izbira elementa, za kar smo uporabili klik z desnim miškinim gumbom. Desni klik smo uporabili tudi pri brisanju omejitev. Za premikanje smo izbrali pritisnjen desni miškin gumb, za brisanje elementov pa desni dvoklik. Pri vnosu teksta nam je zmanjkalo miškinih kombinacij, zato smo tu naredili izjemo in za vnos teksta uporabili klik na ikono.

Končni rezultat

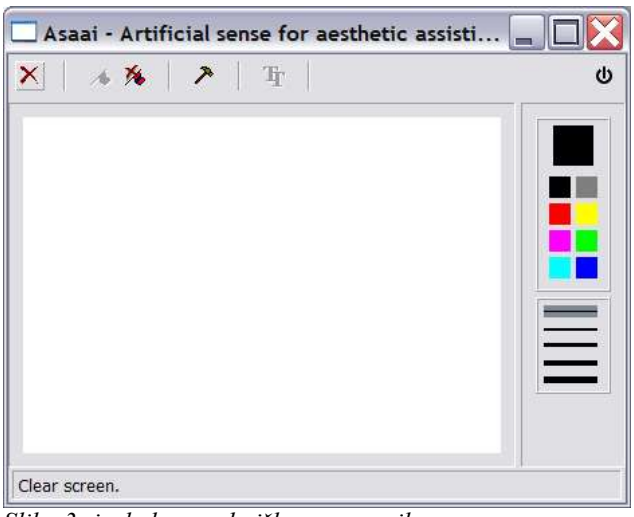
V končnem izdelku nam je zadovoljivo uspelo rešiti prvi problem, to je problem prepozave, saj nam program zna pravilno prepoznati še tako kompleksen lik, narisan z lomljeno črto (slika 1). Prepoznavo krivulj smo izpustili, saj bi z njimi kompleksnost samega problema močno narasla, kar bi se pokazalo predvsem pri umeščanju krivulj med že narisane elemente. Pri drugem problemu smo rešili zgolj osnovne probleme umeščanja, in sicer horizontalno in vertikalno poravnavo, ujemanje po dolžini in ujemanje po legi, kar nam omogoča, da precej dobro prepoznavamo razne pravokotnike (slika 2). Uporabniški vmesnik (slika 3) pa nudi le osnovne možnosti interakcije s programom. Omogoča nam, da izberemo barvo in debelino črte, da črti določimo tekst ter da določimo, ali so omejitve, ki delujejo na elemente, vidne ali ne. Izbrana črta (desni klik) je na ekranu predstavljena s kvadratkoma v njenih krajiščih (slika 4), omejitev, izbrana na izbris, pa je predstavljena z odebelitvijo (slika 5).



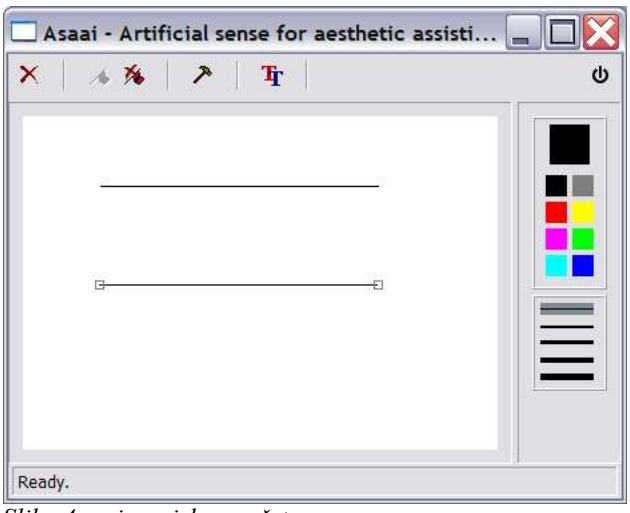
Slika 1: prepoznava kompleksnega lika



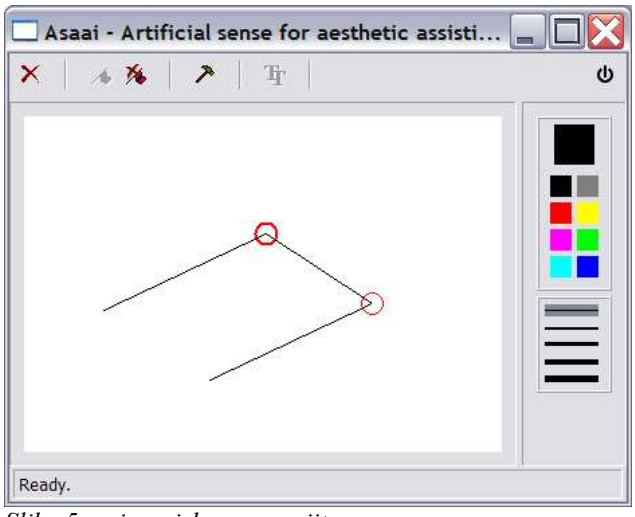
Slika 2: prepoznava pravokotnika



Slika 3: izgled uporabniškega vmesnika



Slika 4: primer izbrane črte



Slika 5: primer izbrane omejitve

Zaključek

Z rešitvijo zadanega problema smo načelno zadovoljni, vendar pa bi bilo treba za neko splošno uporabno aplikacijo rešiti še kar nekaj problemov. Prvi izmed njih je boljše umeščanje posameznih elementov. Pri tem bi bilo nujno dodati še nekatere omejitve, kot so omejitev vzporednosti, omejitev nadaljevanja, omejitev kota, ... Naslednje, kar bi bilo potrebno dodati, je pravilnejše obnašanje pri lomljenju in ponovnem dodajanju omejitev. Prav tako bi morali dodati tudi prepoznavo krivulj. Tu predvsem manjka prepoznava krogov in elips. To bi bilo rešljivo, če bi v program vnesli še prvi način prepoznave, ki je opisan v problemih, torej prepoznavo s pomočjo raznih parametrov. Naslednja stvar, ki bi jo rabili, je manipulacija s tekstom, ki je v našem programu res čisto osnovna. Vse skupaj pa bi bilo potem potrebno zapakirati v nek sposoben uporabniški vmesnik z vsemi potrebnimi funkcijami (odpri, shrani, ...).