



# Projekat – Sendvič (Hasami Shogi)

**Aleksandar Milosavljević**  
**Vladan Mihajlović**

# Osnovne informacije

- Cilj projekta:
  - Formulacija problema
  - Implementacija algoritama za traženje (algoritama za igre)
  - Implementacija procene stanja korišćenjem pravila i zaključivanja
- Jezik: Lisp
- Broj ljudi po projektu: 3
- Datum objavljivanja projekta: 30.10.2017.
- Rok za predaju: 5.1.2018.



# Ocenjivanje

- Broj poena:
  - Projekat nosi maksimalno 20% od konačne ocene
  - Poeni se odnose na kvalitet urađenog rešenja, kao i na aktivnost i zalaganje studenta
- Status:
  - Projekat je obavezan!
  - Minimalni broj poena koji se mora osvojiti je 5!
  - Očekuje od vas da ozbiljno shvatite zaduženja!
  - Ukoliko ne uradite projekat u navedenom roku, naredna prilika je tek sa sledećom generacijom, po pravilima koja će biti tada definisana!



# Takmičenje/turnir

- Posle predaje projekta biće organizovano takmičenje.
- Planirani termin takmičenja je sredina januara.
- Prva tri mesta na turniru donose dodatne poene: 5 za prvo mesto, 3 za drugo i 2 za treće mesto (računaju se kao dodatni poeni za angažovanje u toku semestra).



# Pravila ponašanja

- Probajte da uradite projekat sami, bez pomoći kolega ili prepisivanja.
- Poštujte tuđi rad! Materijal sa Web-a i iz knjiga i radova možete da koristite, ali samo pod uslovom da za sve delove koda ili rešenja koje ste uzeli od nekog navedete referencu!
- Ne dozvolite da od vas neko prepisuje, tj. da neko od kolega koristi vaš rad i vaše rezultate!
- Ako radite u timu, ne dozvolite da vaš kolega iz tima ne radi ništa! Nađite mu zaduženja koja može da uradi – ako mu nešto ne ide, nađite mu druga zaduženja.



# Faze izrade projekta

- Formulacija problema i implementacija interfejsa
  - Rok: 12.11.2017. godine
- Implementacija operatora promene stanja
  - Rok: 26.11.2017. godine
- Implementacija Min-Max algoritma za traženje sa alfa-beta odsecanjem
  - Rok: 10.12.2017. godine
- Definicija heuristike (procena stanja)
  - Rok: 5.1.2018. godine
- Rezultat svake faze je izveštaj koji sadrži dokument sa obrazloženjem rešenja i datoteku sa kodom.

# Opis problema *Sendvič*

- Problem je igra *Sendvič* ((*Dai*) *Hasami Shogi*)
- Tabla je oblika kvadrata stranice  $n$  koju zadaje korisnik (preporučeno  $n=9$ )
- Dva igrača crni i beli (X i O) naizmenično odigravaju po jedan potez pomerajući svoje figure
- Tabla je na početku popunjena tako što su figure svakog igrača raspoređene tako da popune po dve vrste na suprotnim stranama table (po  $2 \cdot n$  figura)
- Pobednik je:
  - Prvi igrač koji naređa 5 susednih figura dijagonalno ili vertikalno (ne horizontalno) tako da nijedna figura nije unutar njegovog polaznog polja
  - Prvi igrač koji ostavi protivnika sa samo 4 figure
- Igra čovek protiv računara i moguće izabrati da prvi igra čovek ili računar







































# Pravila igre *Sendvič*

- Igrači povlače poteze naizmenično
- Igrač, u jednom potezu, može da pomeri samo jednu figuru
- Figura se pomera u bilo kom od dva smera u horizontalnim i vertikalnim pravcima
- Postoji dva tipa poteza:
  - Ako je susedno polje u izabranom smeru prazno figure se pomeraju u izabranom smeru sve dok ne naiđu na prepreku
  - Ako je na susednom polju u izabranom smeru figura (prijateljska ili protivnička), potez je moguć samo ako je polje susedno figuri koja se preskače u izabranom smeru prazno
- Moguće je zarobiti i ukloniti protivničke figure ako se oko njih napravi sendvič
  - Nakon poteza ukoliko postoji neprekidni niz protivničkih figure koje su u horizontalnom ili vertikalnom pravcu okružene sa dve prijateljske figure one se uklanjaju
    - Ako se jednim potezom napravi više sendviča uklanjaju se figure iz svih sendviča
  - Ako se potezom umetne figura u sendvič ona se ne uklanja



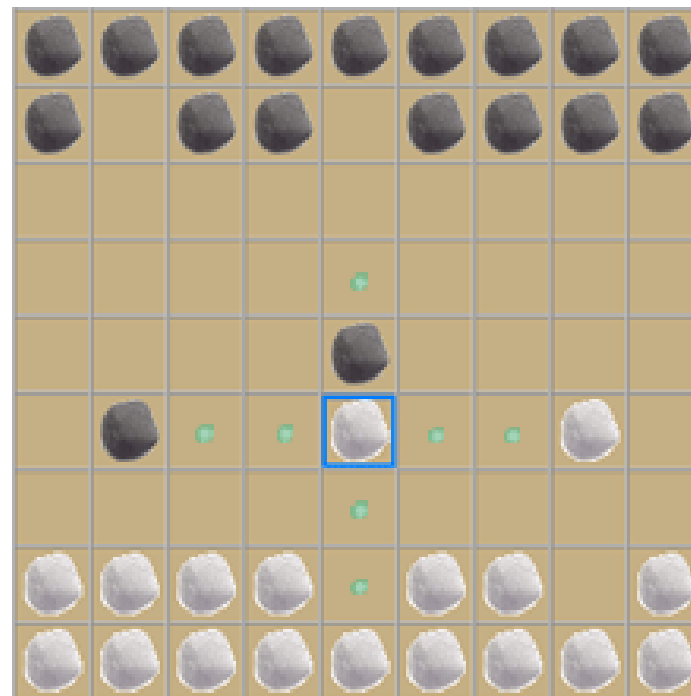
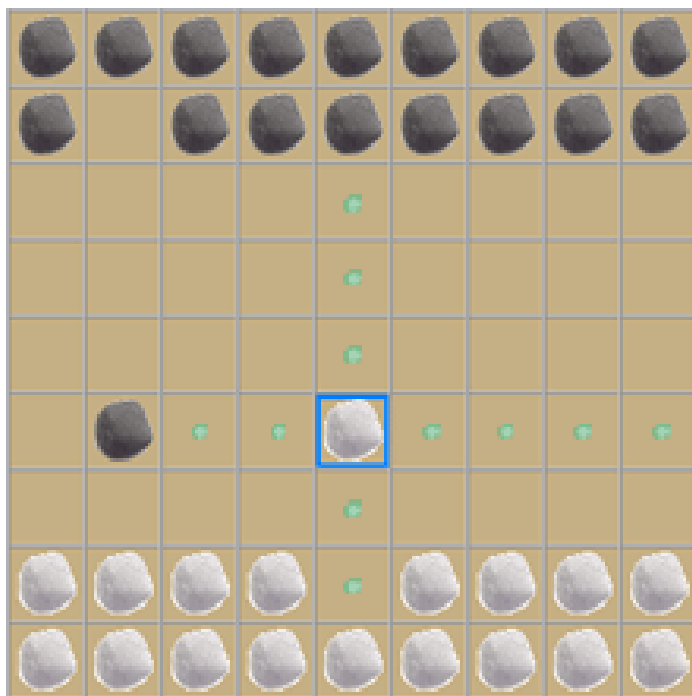


# Sendvič – Početak igre

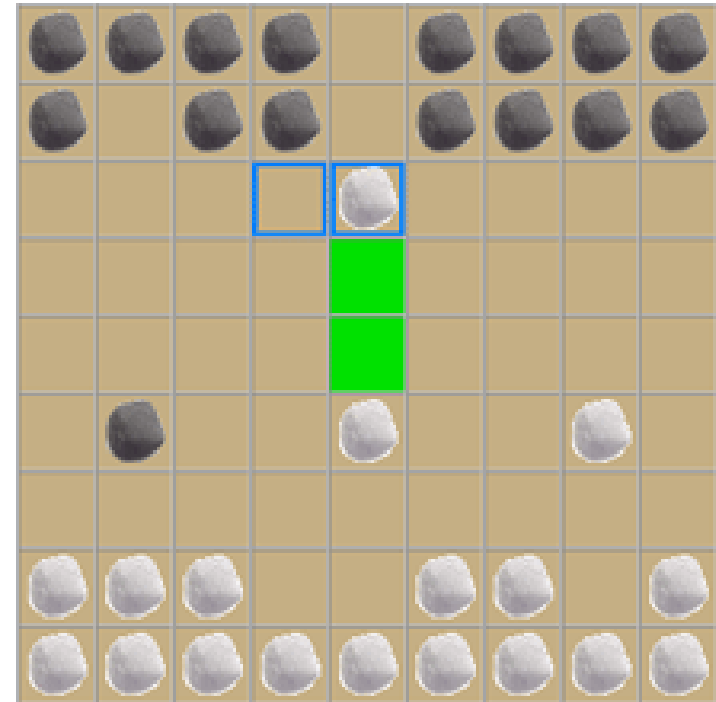
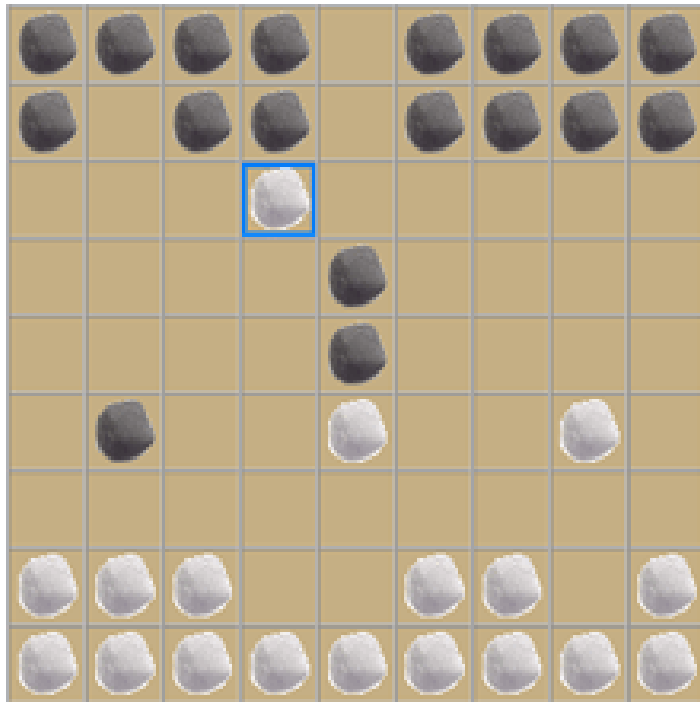
	9	8	7	6	5	4	3	2	1	
A										A
B										B
C										C
D										D
E										E
F										F
G										G
H										H
I										I
	9	8	7	6	5	4	3	2	1	



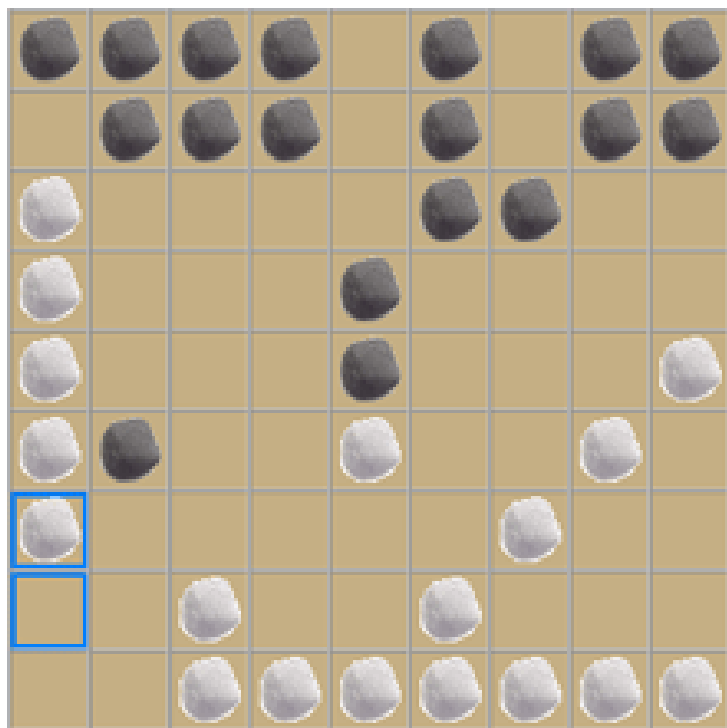
# Sendvič – Primeri poteza



# Sendvič – Primer sendvič poteza



# Sendvič – Kraj igre



- Leva vertikalna linija predstavlja kraj igre
- Desna dijagonalna linija NE predstavlja kraj igre jer su dve bele figure unutar početnog polja



# Zadatak I – Formulacija problema i interfejs (1)

- Definirati način za predstavljanje stanja problema (igre)
- Napisati funkciju za postavljanje početnog stanja na osnovu zadate veličine table
- Napisati funkcije za testiranje ciljnog stanja, tj. da li neko od igrača ima 5 figura u liniji van početnog polja ili ima manje od 4 figure

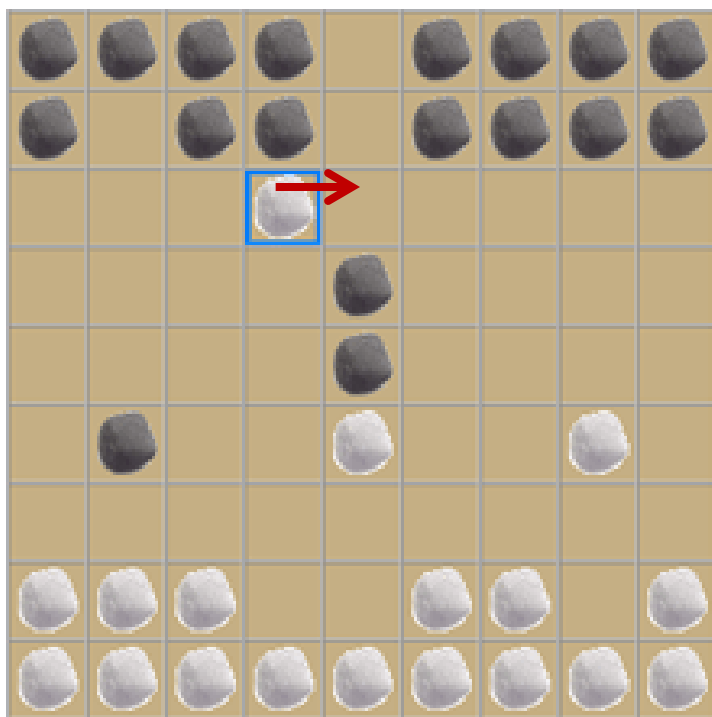




# Zadatak I – Formulacija problema i interfejs (2)

- Omogućiti izbor ko će igrati prvi (čovjek ili računar)
- Prvi igra uvek igrač X, a drugi igrač O
- Implementirati funkcije koje obezbeđuju prikaz trenutnog stanja problema (igre)
- Realizovati funkcije koje na osnovu zadatog poteza igrača, u obliku  $((\text{vrsta}_p \text{ kolona}_p) (\text{vrsta}_k \text{ kolona}_k))$ , omogućavaju:
  - proveru da li je potez valjan
  - odigravanje poteza ukoliko jeste, tj. promenu trenutnog stanje problema (igre)



# Zadatak I – Interfejs



	1	2	3	4	5	6	7	8	9
A	X	X	X	X	-	X	X	X	X
B	X	-	X	X	-	X	X	X	X
C	-	-	-			-	-	-	-
D	-	-	-	-	X	-	-	-	-
E	-	-	-	-	X	-	-	-	-
F	-	X	-	-	O	-	-	O	-
G	-	-	-	-	-	-	-	-	-
H	O	O	O	-	-	O	O	-	O
I	O	O	O	O	O	O	O	O	O

Potez O: ((C 4)(C 5))

# Zadatak II –

## Operatori promene stanja

- Napisati funkcije za operatore promene stanja problema (igre) u opštem slučaju (proizvoljno stanje na tabli)
  - Na osnovu trenutne (proizvoljne) situacije na tabli (stanja) i zadatog (validnog) poteza formira novu situaciju na tabli (stanje). Ne menjati postojeće stanje već napraviti novo i na njemu odigrati potez.
  - Na osnovu trenutne (proizvoljne) situacije na tabli (stanja) i igrača koji je na potezu formira listu svih mogućih situacija na tabli (stanja), korišćenjem funkcije iz prethodne tačke
- Realizovati funkcije koje obezbeđuju odigravanje partije između dva igrača (čoveka, ne računara i čoveka)
  - unos poteza i provera da li je potez moguć
  - ukoliko nije moguć zahtevati unos novog poteza
  - ukoliko je moguć odigrati ga i promeniti trenutno stanje
  - prikazati novonastalo stanje sistema





# Zadatak III – Min-max algoritam

- Implementirati Min-Max algoritam sa alfa-beta odsecanjem za zadati problem
- Obezbediti da funkcija Min-Max sa alfa-beta odsecanjem ima ulazni parametar kojim se definiše dubina pretraživanja
- Obezbediti da funkcija Min-Max sa alfa-beta odsecanjem vrati potez koji treba odigrati ili stanje u koje treba preći
- Funkciju za određivanje heuristike ne treba implementirati
  - Napraviti funkciju koja za odgovarajuća stanja vraća karakteristične vrednosti samo u svrhu testiranja ispravnosti napravljenog Min-Max algoritma



# Zadatak IV – Heuristika

- U implementaciju Min-Max-a sa alfa-beta odsecanjem dodati funkciju za procenu stanja koja se poziva kada se dostigne zadata dubina traženja.
- Implementirati funkciju koja vrši procenu stanja na osnovu pravila zaključivanja
- Funkcija za procenu stanja kao parametre treba da ima oznaku igrača za kojeg računa valjanost stanja, kao i samo stanje za koju se računa procena.
- Procena stanja se mora vršiti isključivo korišćenjem mehanizma zaključivanja nad prethodno definisanim skupom pravila. Zadatak je formulisati skup pravila i iskoristiti ih na adekvatan način za izračunavanje heuristike.
- Za izvođenje potrebnih zaključaka (izvršavanje upita nad skupom činjenica kojima se opisuje stanje) koristiti mašinu za zaključivanje.
- Implementirati funkciju koja prevodi stanje u listu činjenica ...

