SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2858

RAZVOJ AGENTA ZA DRUŠTVENU IGRU CATAN

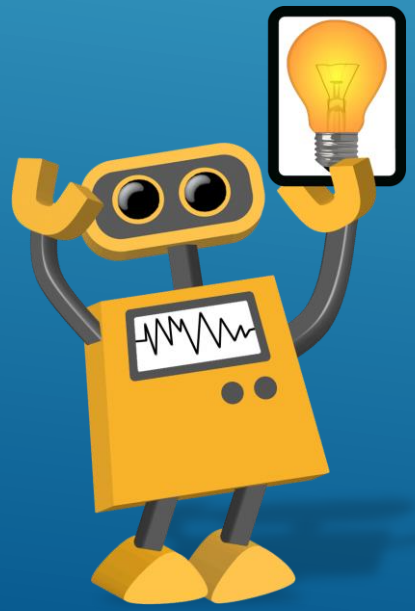
IVAN SKORIĆ

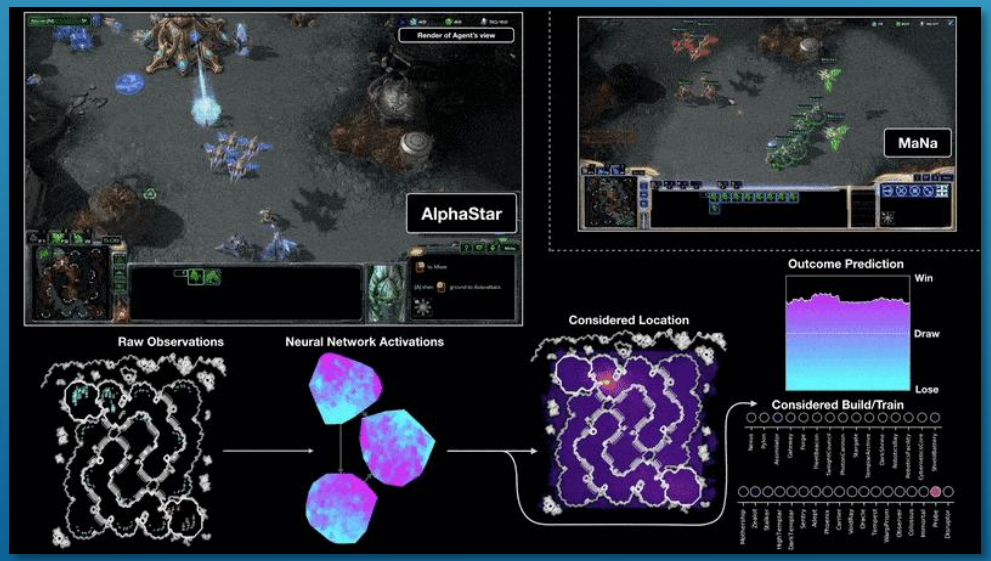
Mentor: doc. dr. sc. Marko Đurasević

04. srpnja 2022.

UVOD

- Umjetna inteligencija (AI):
  - Učenje, donošenje odluka i zaključaka, rješavanje problema
- Umjetna inteligencija u igranju igara:
  - Deep Blue (šah), AlphaZero (šah, Go, Shogi), AlphaStar (StarCraft II)
- Cilj rada:
  - Razvoj agenta za stratešku društvenu igru Catan



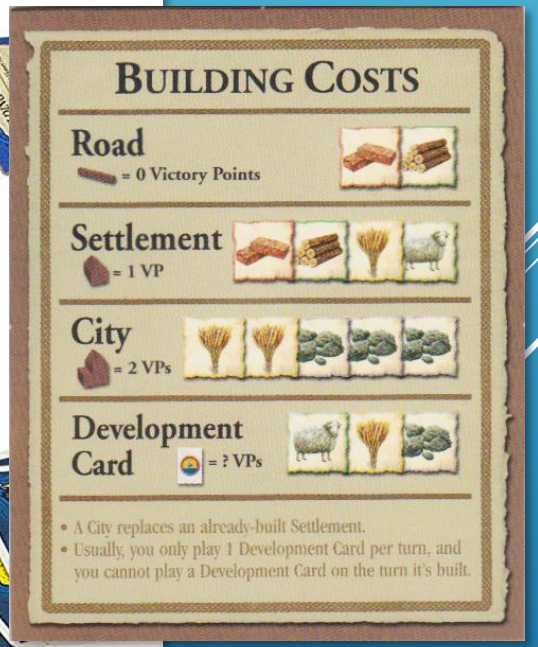


CATAN


Hills
Produce Brick
Forest
Produces Lumber
Mountains
Produce Ore
Fields
Produce Grain
Pasture
Produces Wool
Desert
Produces Nothing

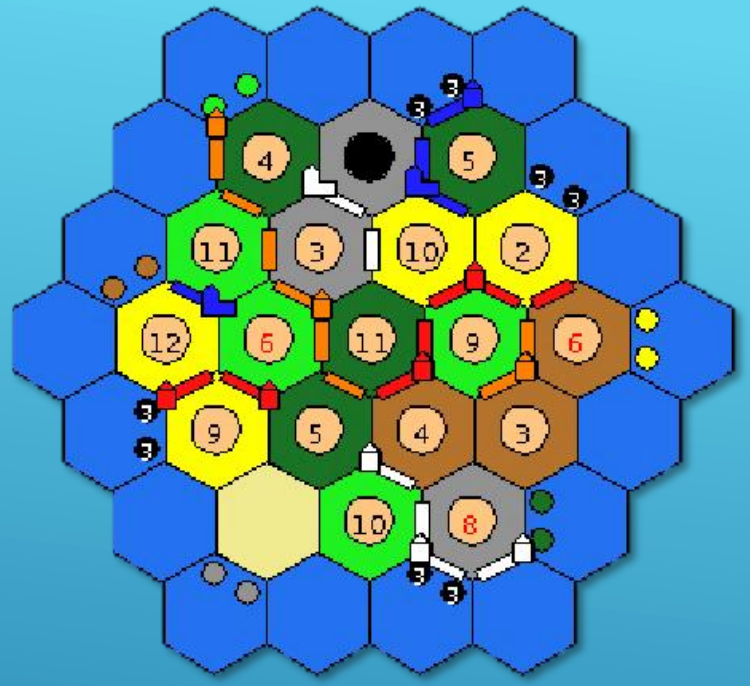

LONGEST ROAD
2 Victory Points!
This card goes to the player with the longest road of at least 5 segments. Another player who builds a longer road takes this card.

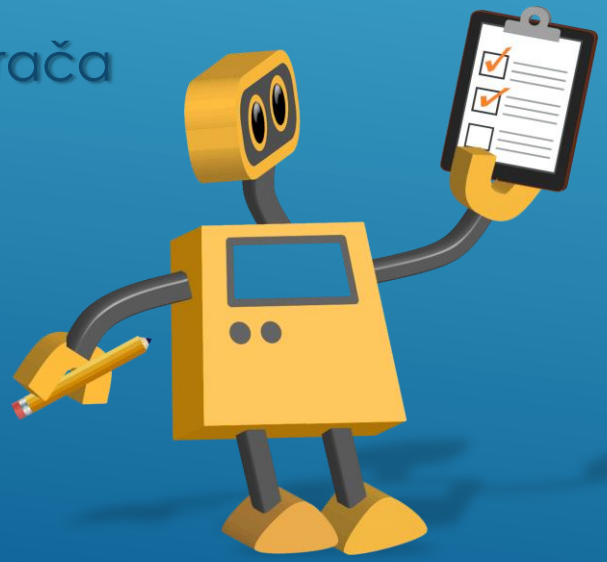

LARGEST ARMY
2 Victory Points!
The first player to play 3 knight cards gets this card. Another player who plays more knight cards takes this card.


BUILDING COSTS
Road
= 0 Victory Points
Settlement
= 1 VP
City
= 2 VPs
Development Card
= ? VPs
• A City replaces an already-built Settlement.
• Usually, you only play 1 Development Card per turn, and you cannot play a Development Card on the turn it's built.

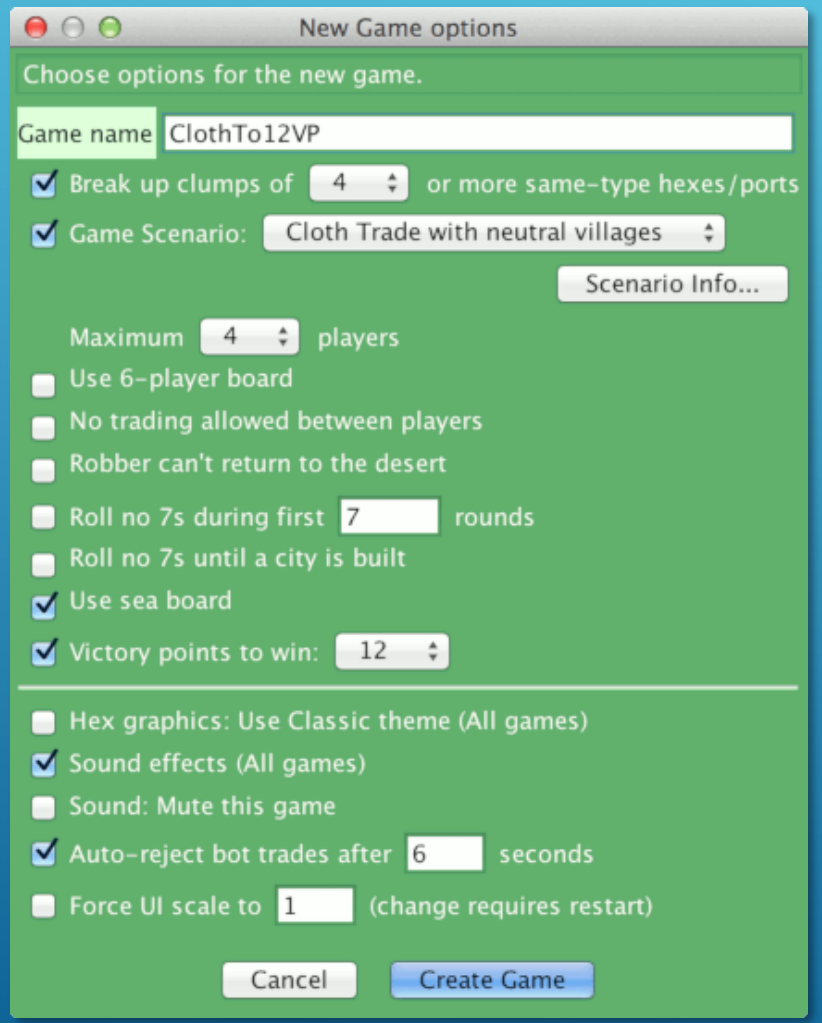
RADNI OKVIRI ZA CATAN

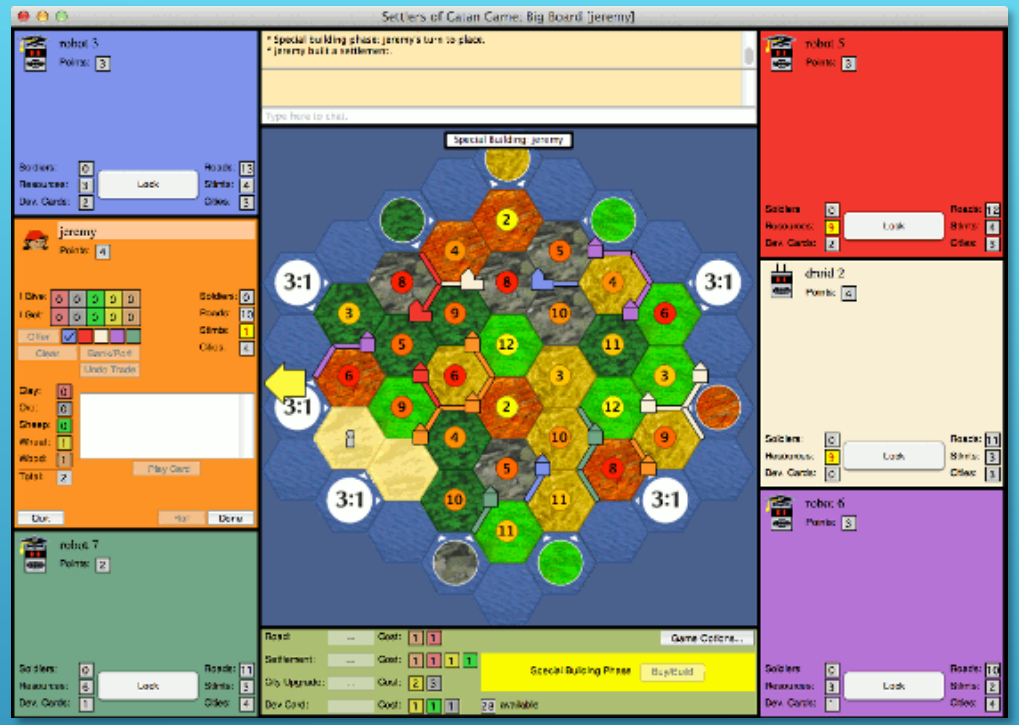
- Open-source
- Postojeća implementacija agenta
- Mogućnost implementacije novog agenta
- Igranje kroz GUI
- Igranje bez GUI-a → prikupljanje statistike, učenje agenata
- Jednostavan model stanja
- Fleksibilno programsko sučelje igrača



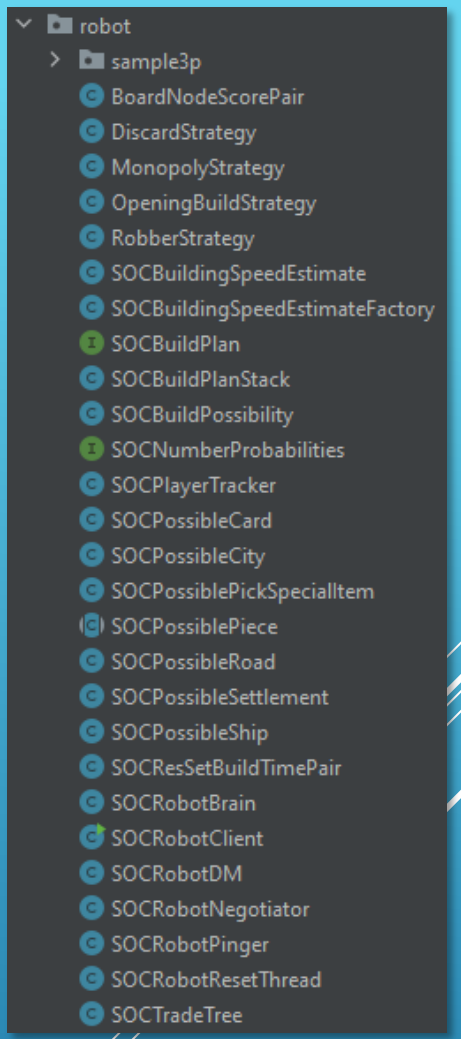


JSettlers


New Game options
Choose options for the new game.
Game name
ClothTo12VP
Break up clumps of
4
or more same-type hexes/ports
Game Scenario:
Cloth Trade with neutral villages
Scenario Info...
Maximum
4
players
Use 6-player board
No trading allowed between players
Robber can't return to the desert
Roll no 7s during first
7
rounds
Roll no 7s until a city is built
Use sea board
Victory points to win:
12
Hex graphics: Use Classic theme (All games)
Sound effects (All games)
Sound: Mute this game
Auto-reject bot trades after
6
seconds
Force UI scale to
1
(change requires restart)
Cancel
Create Game

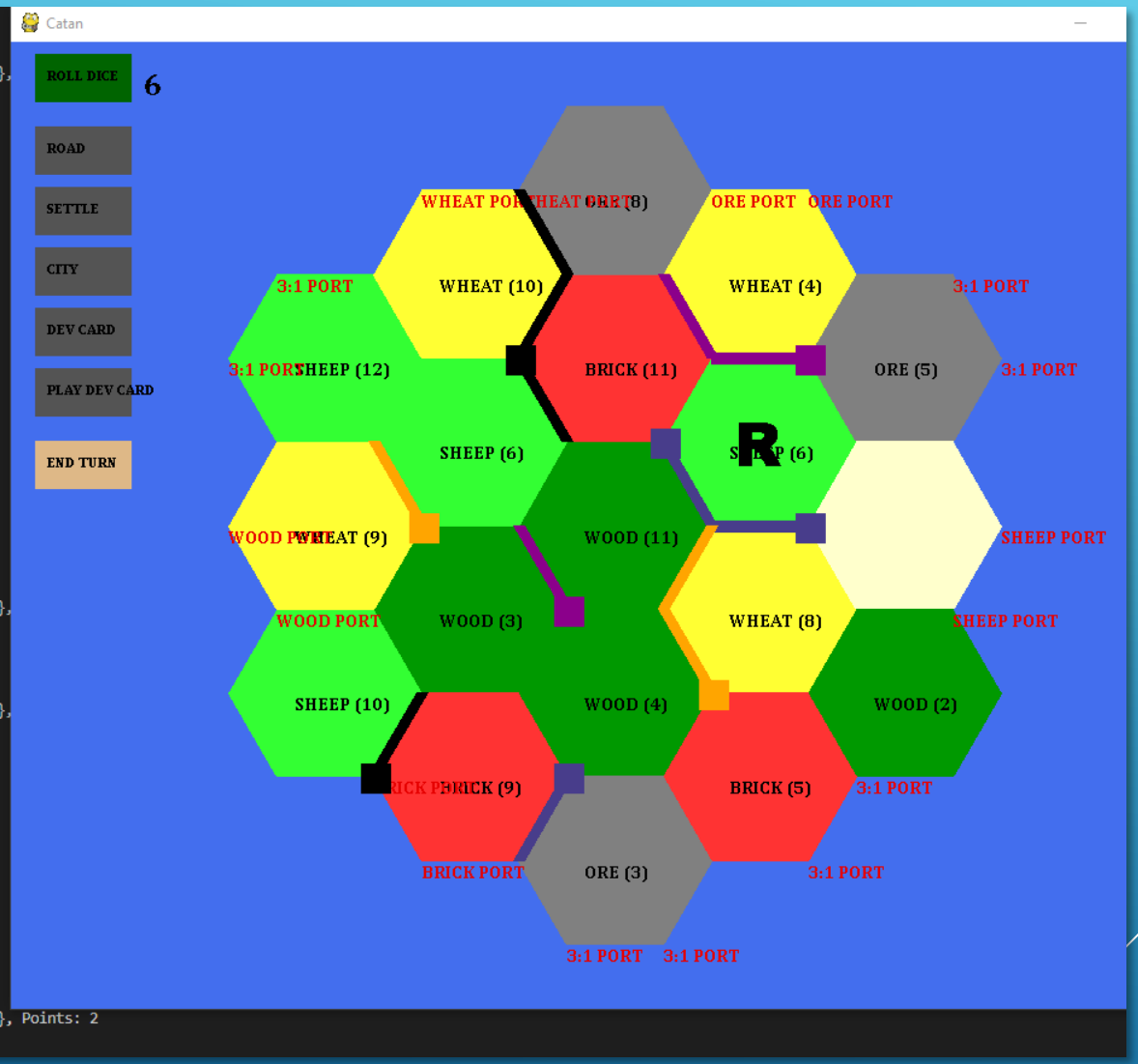

Settlers of Catan Game: Big Board 'jeremy'
robot 3
jeremy
robot 5
druid 2
robot 6
robot 7
Special Building: jeremy
3:1

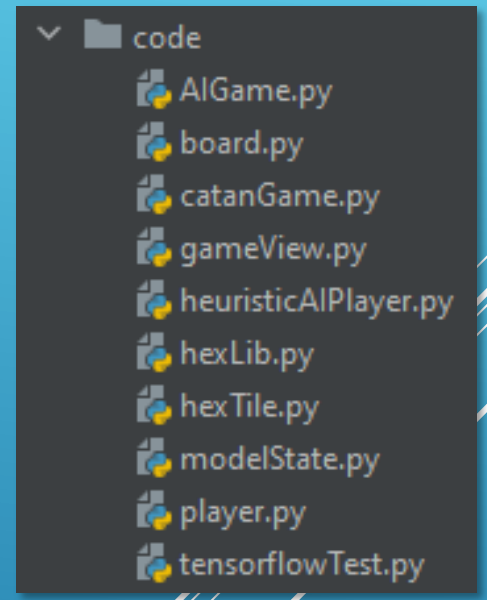

Settlers of Catan Game: ClothTo12VP [Player]
Player
droid 1
droid 2
robot 6
Unlocked
3:1


robot
sample3p
BoardNodeScorePair
DiscardStrategy
MonopolyStrategy
OpeningBuildStrategy
RobberStrategy
SOCBuildingSpeedEstimate
SOCBuildingSpeedEstimateFactory
SOCBuildPlan
SOCBuildPlanStack
SOCBuildPossibility
SOCNumberProbabilities
SOCPlayerTracker
SOCPossibleCard
SOCPossibleCity
SOCPossiblePickSpecialItem
SOCPossiblePiece
SOCPossibleRoad
SOCPossibleSettlement
SOCPossibleShip
SOCResSetBuildTimePair
SOCRobotBrain
SOCRobotClient
SOCRobotDM
SOCRobotNegotiator
SOCRobotPinger
SOCRobotResetThread
SOCTradeTree

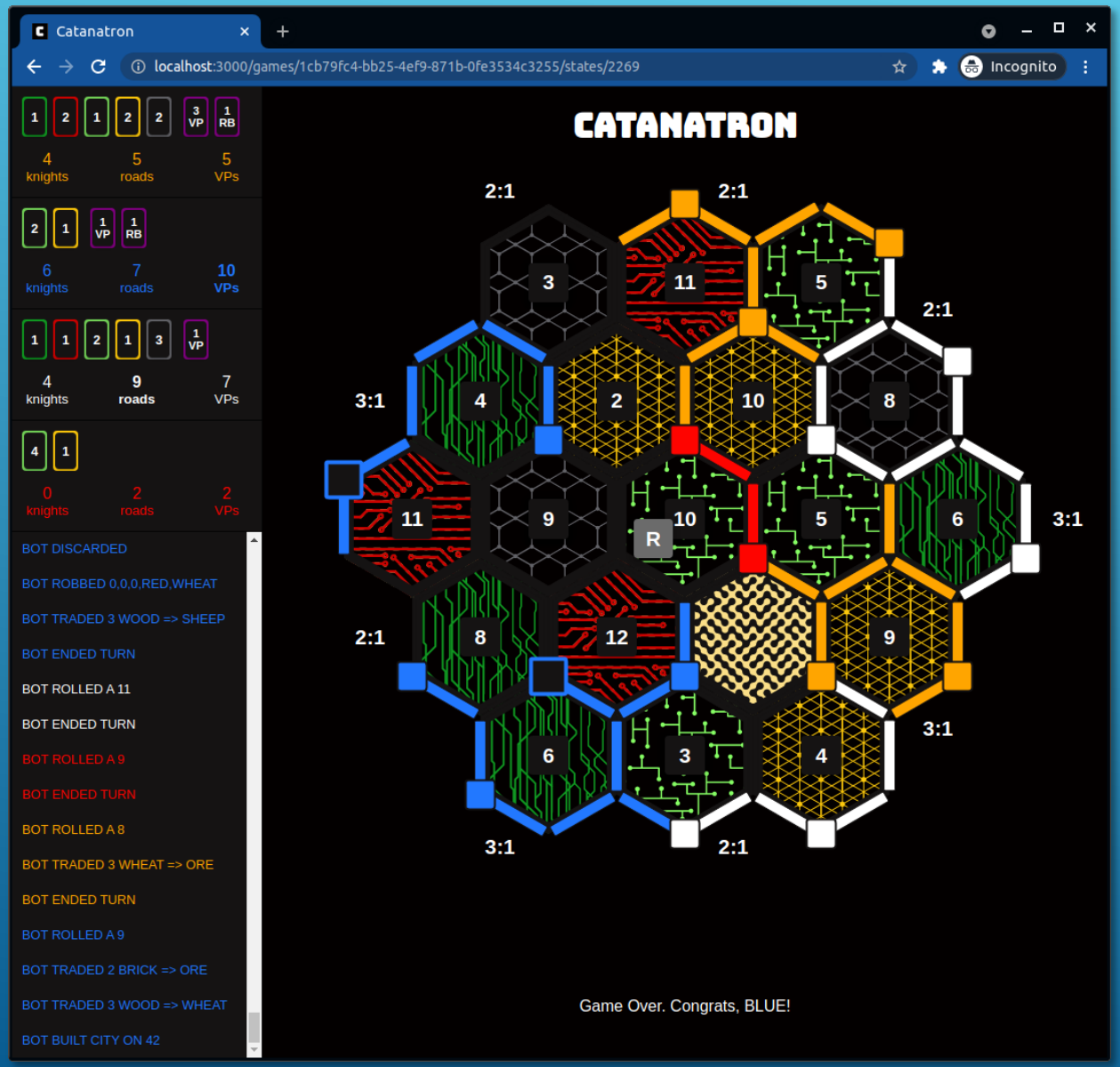
CatanAI

```
Random-Greedy-AI collects 1 WHEAT from Settlement
Player:Random-Greedy-AI, Resources:{'ORE': 0, 'BRICK': 1, 'WHEAT': 2, 'WOOD': 1, 'SHEEP': 1},
MaxRoadLength:1, LongestRoad:False

Insufficient Resources to Build City. Build Cost: 3 ORE, 2 WHEAT
Player:C, Resources:{'ORE': 6, 'BRICK': 2, 'WHEAT': 0, 'WOOD': 5, 'SHEEP': 1}, Points: 2
C Built a Road. MaxRoadLength: 2
Player:C, Resources:{'ORE': 6, 'BRICK': 1, 'WHEAT': 0, 'WOOD': 4, 'SHEEP': 1}, Points: 2
Insufficient Resources to Build Settlement. Build Cost: 1 BRICK, 1 WOOD, 1 WHEAT, 1 SHEEP
Player:C, Resources:{'ORE': 6, 'BRICK': 1, 'WHEAT': 0, 'WOOD': 4, 'SHEEP': 1}, Points: 2
Ending Turn!
--------------------------------------------------------------------------------
Current Player: Random-Greedy-AI
Dice Roll =  4 { 3 1 }
Player:A, Resources:{'ORE': 5, 'BRICK': 2, 'WHEAT': 1, 'WOOD': 0, 'SHEEP': 2}, Points: 2
MaxRoadLength:3, LongestRoad:False

B collects 1 WOOD from Settlement
Player:B, Resources:{'ORE': 6, 'BRICK': 2, 'WHEAT': 1, 'WOOD': 2, 'SHEEP': 0}, Points: 3
MaxRoadLength:2, LongestRoad:False

C collects 1 WHEAT from Settlement
C collects 1 WOOD from Settlement
Player:C, Resources:{'ORE': 6, 'BRICK': 1, 'WHEAT': 1, 'WOOD': 5, 'SHEEP': 1}, Points: 2
MaxRoadLength:2, LongestRoad:False

Random-Greedy-AI collects 1 WOOD from Settlement
Player:Random-Greedy-AI, Resources:{'ORE': 0, 'BRICK': 1, 'WHEAT': 2, 'WOOD': 2, 'SHEEP': 1},
MaxRoadLength:1, LongestRoad:False

AI Player Random-Greedy-AI playing...
Random-Greedy-AI Built a Road. MaxRoadLength: 2
Player:Random-Greedy-AI, Resources:{'ORE': 0, 'BRICK': 0, 'WHEAT': 2, 'WOOD': 1, 'SHEEP': 1},
--------------------------------------------------------------------------------
Current Player: A
Dice Roll =  6 { 5 1 }
A collects 1 SHEEP from Settlement
Player:A, Resources:{'ORE': 5, 'BRICK': 2, 'WHEAT': 1, 'WOOD': 0, 'SHEEP': 3}, Points: 2
MaxRoadLength:3, LongestRoad:False

Player:B, Resources:{'ORE': 6, 'BRICK': 2, 'WHEAT': 1, 'WOOD': 2, 'SHEEP': 0}, Points: 3
MaxRoadLength:2, LongestRoad:False

Player:C, Resources:{'ORE': 6, 'BRICK': 1, 'WHEAT': 1, 'WOOD': 5, 'SHEEP': 1}, Points: 2
MaxRoadLength:2, LongestRoad:False

Random-Greedy-AI collects 1 SHEEP from Settlement
Player:Random-Greedy-AI, Resources:{'ORE': 0, 'BRICK': 0, 'WHEAT': 2, 'WOOD': 1, 'SHEEP': 2}, Points: 2
MaxRoadLength:2, LongestRoad:False
```





Catanatron




```
Terminal
bcollazo (e) venv  master  ~  catanatron  catanatron-play --players=R,R,R,W --num=100
Playing 100 games...                                                    100% 0:00:00
RandomPlayer:RED                                                         21%
RandomPlayer:BLUE                                                        20%
RandomPlayer:ORANGE                                                      20%
WeightedRandomPlayer:WHITE                                               39%
```

Last 10 Games

#	SEATING	TURNS	RED VP	BLUE VP	ORANGE VP	WHITE VP	WINNER
91	RED,ORANGE,WHITE,BLUE	223	4	9	11	7	ORANGE
92	WHITE,ORANGE,RED,BLUE	118	2	2	2	10	WHITE
93	WHITE,RED,BLUE,ORANGE	332	4	10	2	7	BLUE
94	RED,ORANGE,BLUE,WHITE	216	2	10	6	2	BLUE
95	BLUE,RED,WHITE,ORANGE	140	2	6	5	10	WHITE
96	BLUE,WHITE,RED,ORANGE	393	5	7	10	8	ORANGE
97	WHITE,RED,BLUE,ORANGE	263	11	2	4	2	RED
98	ORANGE,WHITE,RED,BLUE	139	2	2	5	10	WHITE
99	WHITE,BLUE,RED,ORANGE	583	4	10	9	3	BLUE
100	ORANGE,WHITE,RED,BLUE	220	11	2	5	2	RED

Player Summary

	WINS	AVG VPs
RandomPlayer:RED	21	5.02
RandomPlayer:BLUE	20	5.16
RandomPlayer:ORANGE	20	5.35
WeightedRandomPlayer:WHITE	39	6.88

Game Summary

AVG TICKS	AVG TURNS	AVG DURATION
888.01	301.48	0.064 secs

```
bcollazo (e) venv  master  ~  catanatron
```

```python
class Player:
    """Interface to represent a player's decision logic.

    Formulated as a class (instead of a function) so that players
    can have an initialization that can later be serialized to
    the database via pickle.
    """

    def __init__(self, color, is_bot=True):
        self.color = color
        self.is_bot = is_bot

    def decide(self, game, playable_actions):
        """Should return one of the playable_actions.

        Args:
            game (Game): complete game state. read-only.
            playable_actions (Iterable[Action]): options right now
        """
        raise NotImplementedError
```

IMPLEMENTACIJA

- Tri *baseline* igrača
- Random Player:
  - Nasumično bira jednu od raspoloživih akcija
- Weighted Random Player:
  - Nasumično bira akciju uz zadane težine
  - Izgradnja grada > izgradnja naselja > izgradnja ceste…
- Greedy VP Player:
  - Bira akciju s maksimalnim rezultirajućim brojem VP-a
  - "Pohlepna" strategija

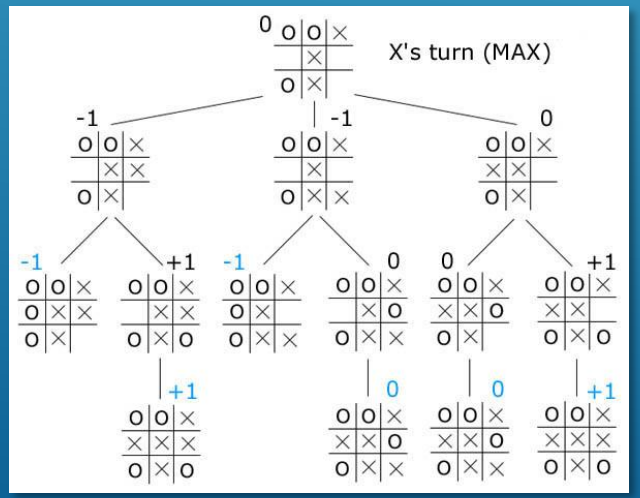
```
self.action_type_weights = {
    ActionType.BUILD_CITY: 1000,
    ActionType.BUILD_SETTLEMENT: 700,
    ActionType.BUILD_ROAD: 300,
    ActionType.BUY_DEVELOPMENT_CARD: 250,
    ActionType.PLAY_KNIGHT_CARD: 50,
    ActionType.PLAY_MONOPOLY: 50,
    ActionType.PLAY_ROAD_BUILDING: 50,
    ActionType.PLAY_YEAR_OF_PLENTY: 50
}
```

```
weights = list(map(lambda action:
                   self.action_type_weights.get(action.action_type, 1),
                   playable_actions
                   ))

return random.choices(playable_actions, weights, k=1)[0]
```

IMPLEMENTACIJA

- Value Function Player:
  - Bira akciju s maksimalnom rezultirajućom vrijednosti heurističke funkcije
  - Heuristika:
    - Skup pravila za usmjeravanje pretraživanja stanja
    - *Koliko je stanje igre povoljno za igrača?*
    - Broj gradova i naselja, dužina ceste, proizvodnja resursa…
  - 48 podataka o stanju igre → 18 agregiranih ulaznih vrijednosti

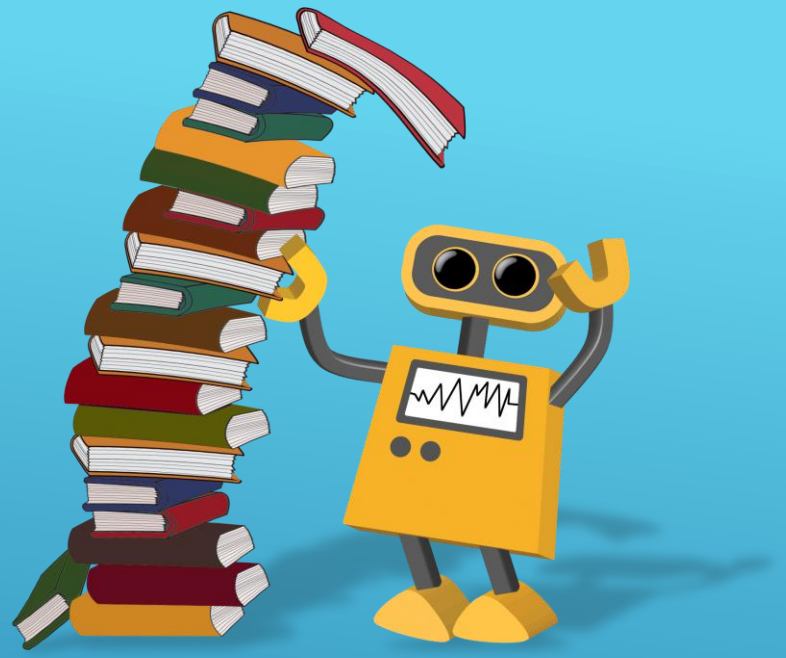
```
self.value_multipliers: dict[Value, int] = {
    Value.VP: 1500,

    Value.CITIES: 400,
    Value.SETTLEMENTS: 250,
    Value.ROAD_LENGTH: 150,
    Value.KNIGHTS: 80,
    Value.DEV_CARDS: 40,

    Value.RESOURCE_TYPES: 120,

    Value.PORT_VALUE: 100,

    Value.BUILDABLE_NODES: 120,
    Value.BUILDABLE_EDGES: 80,

    Value.DISCARD_PENALTY: -100,

    Value.ENEMY_VP: -200,

    Value.ENEMY_RESOURCE_PRODUCTION: -100,
    Value.ENEMY_RESOURCE_TYPES: -50,

    Value.ENEMY_BUILDABLE_NODES: -100,
    Value.ENEMY_BUILDABLE_EDGES: -50
}
```

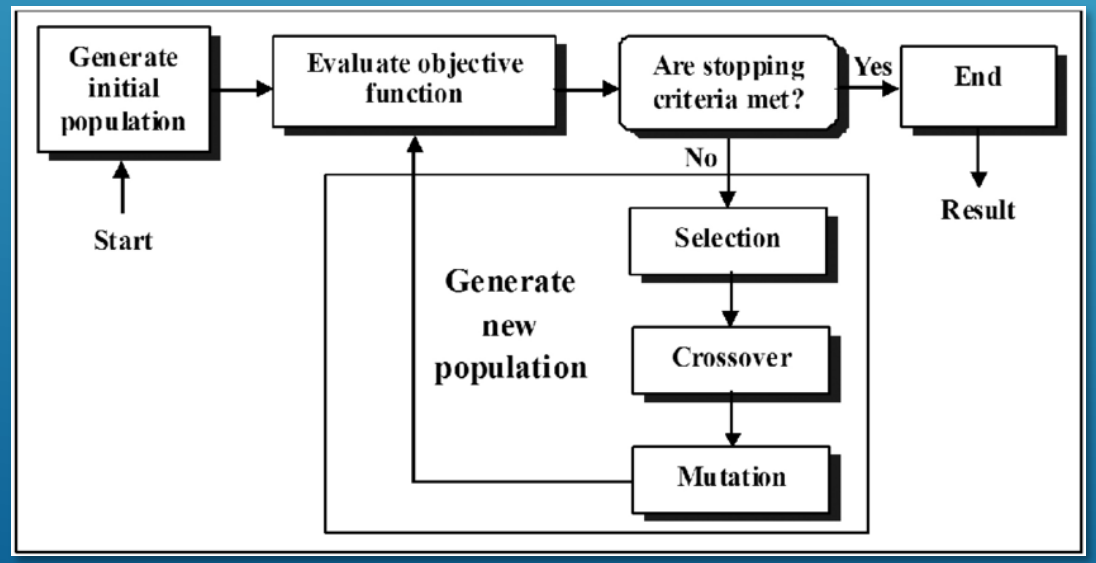


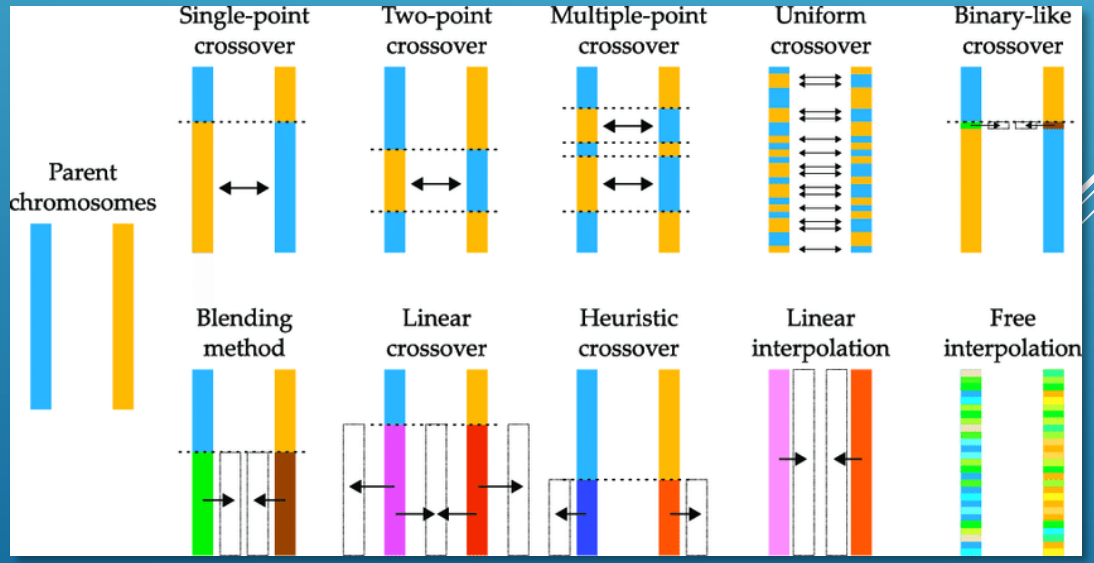

```
self.expense_multipliers: dict[Expense, int] = {
    Expense.CITY: 100,
    Expense.SETTLEMENT: 50,
    Expense.ROAD: 20,
    Expense.DEV_CARD: 20
}

self.resource_multipliers: dict[Resource, int] = {
    Resource.BRICK: 100,
    Resource.ORE: 100,
    Resource.SHEEP: 100,
    Resource.WHEAT: 100,
    Resource.WOOD: 100
}
```

IMPLEMENTACIJA

- Value Function Player:
  - Učenje parametara – genetski algoritam:
    - Turnirska selekcija → 4 nasumična igrača
    - Funkcija dobrote → broj pobjeda u turniru
    - Križanje → aritmetičko i heurističko
    - Mutacija → normalna raspodjela (Gauss)

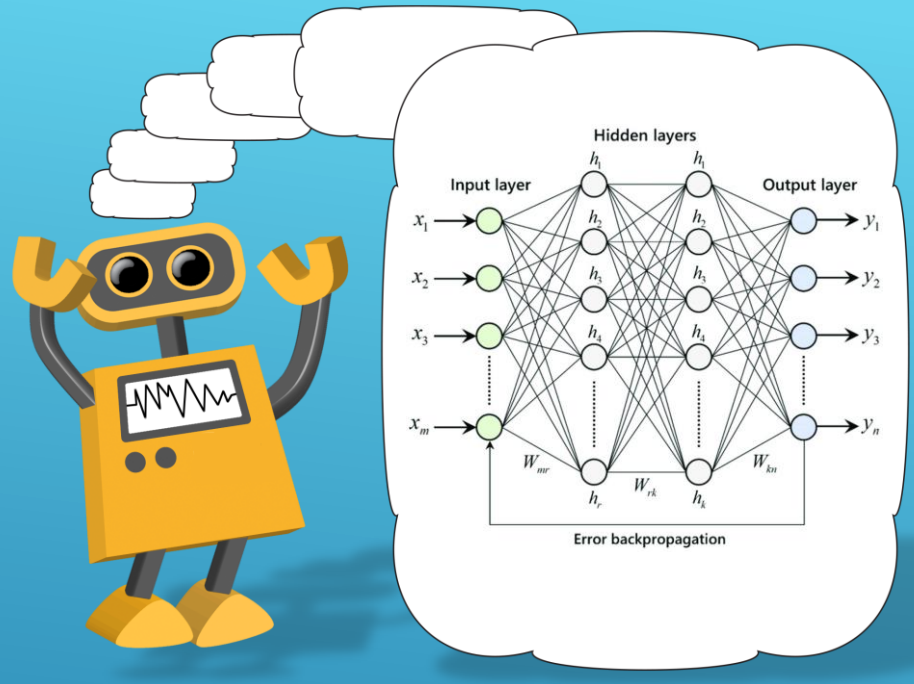


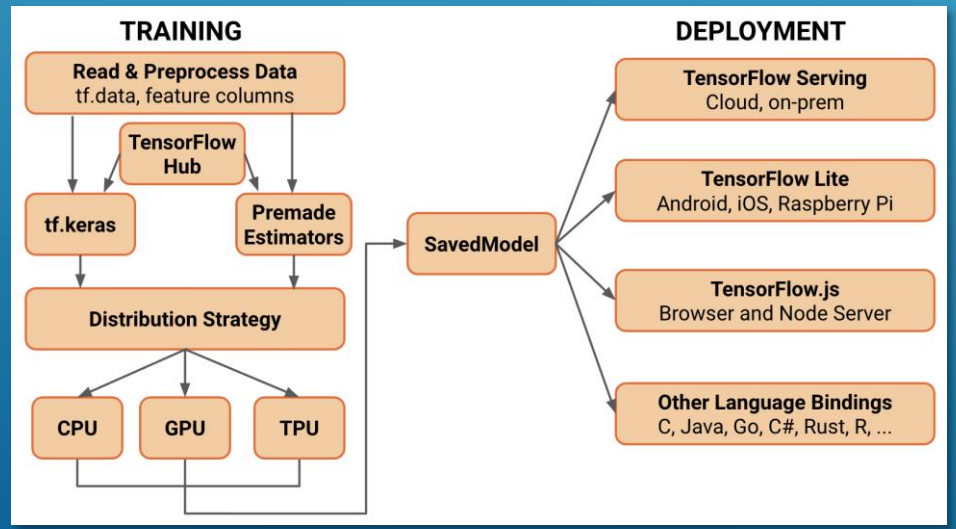




IMPLEMENTACIJA

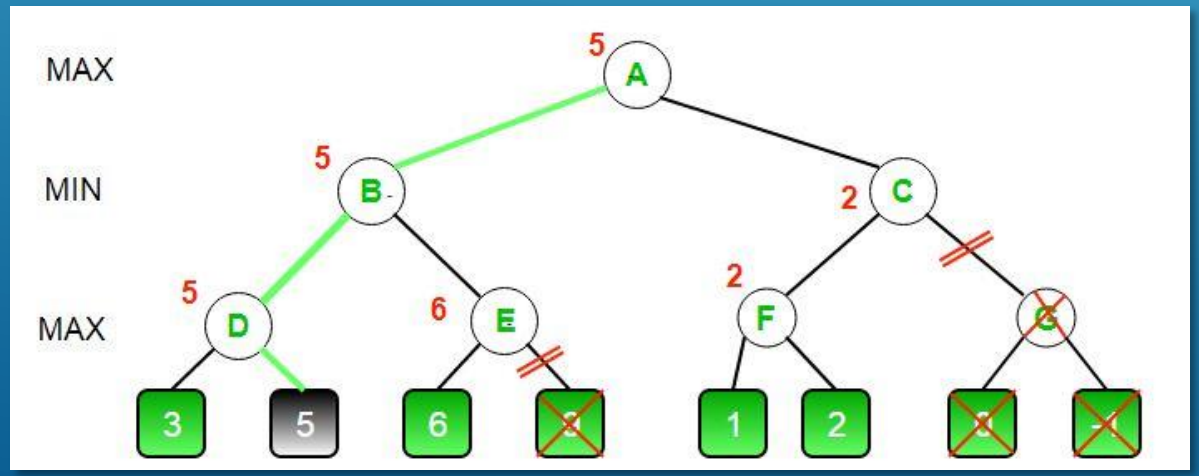
- Neural Network Player:
  - Evaluira stanja neuronskom mrežom:
    - Ulaz: 48 podataka o stanju
    - Izlaz: procijenjena vrijednost stanja
  - TensorFlow model
  - Učenje mreže – neuroevolucija:
    - Pred-treniranje na heurističku funkciju VFP-a
    - Genetski algoritam nad težinama mreže
  - Arhitektura mreže:
    - → 48 – 32 – 16 – 1 →





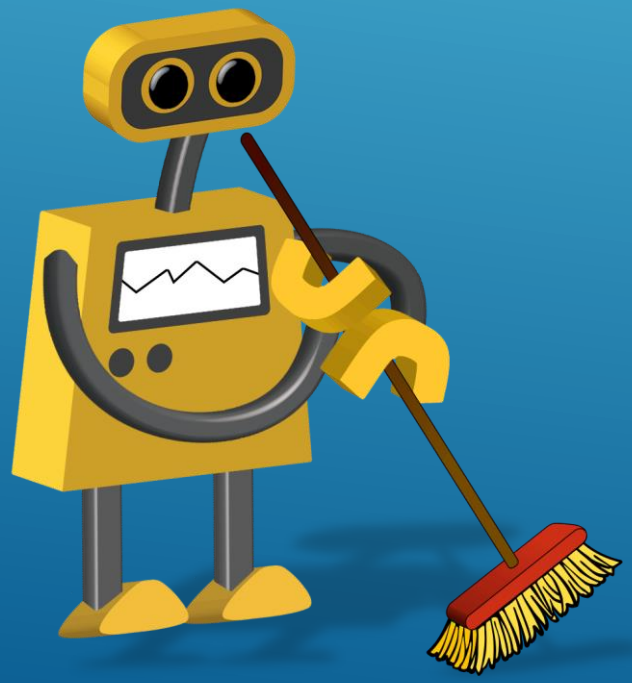
IMPLEMENTACIJA

- Tree Search Player:
  - Pretraživanje stabla igre algoritmom minimaks
  - Algoritam minimaks:
    - Igrač maksimizira svoj dobitak, protivnici ga minimiziraju
    - Heuristika → pretraživanje do ograničene dubine
    - Alfa-beta podrezivanje
  - Potrebno ekspandirati nedeterminističke akcije na sve moguće ishode (npr. bacanje kockica)
  - Isti parametri kao i VFP



IMPLEMENTACIJA

- Dorade radnog okvira:
  - Skripte za pokretanje igri (`play-batch` i `play-ui`)
  - Paralelno višeprocesorsko izvođenje igri (paket `concurrent`)
  - Univerzalna implementacija genetskog algoritma

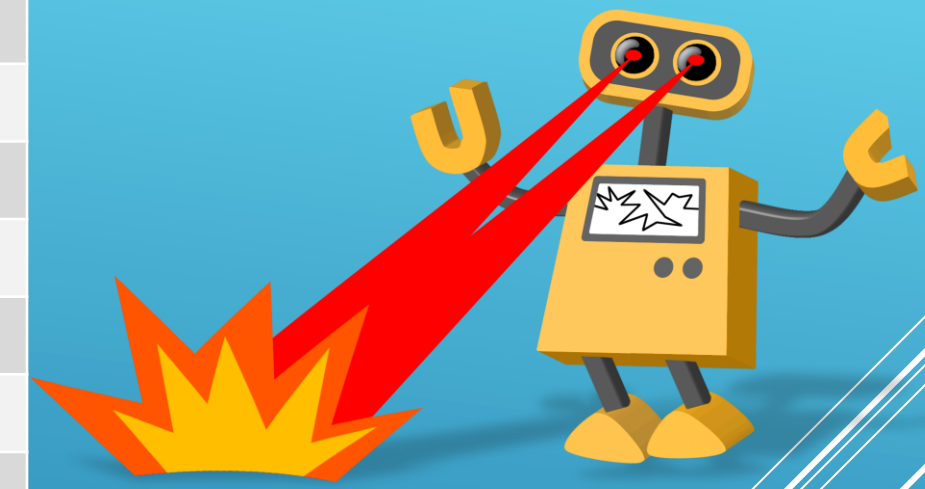


```python
def play_batch_core(num_games, players, game_config, accumulators=[]):
    for accumulator in accumulators:
        if isinstance(accumulator, SimulationAccumulator):
            accumulator.before_all()

    with concurrent.futures.ProcessPoolExecutor(max_workers=max(os.cpu_count() - 2, 1)) as executor:
        futures = [executor.submit(play_game, players, game_config, deepcopy(accumulators)) for _ in range(num_games)]

        for future in concurrent.futures.as_completed(futures):
            game, accumulator_copies = future.result()
            for accumulator, copy in zip(accumulators, accumulator_copies):
                accumulator.join(copy)
            yield game

    for accumulator in accumulators:
        if isinstance(accumulator, SimulationAccumulator):
            accumulator.after_all()
```

REZULTATI

Agent	Pobjede – 1v1 [%]	Broj igri
Tree Search Player + (d=3)	60% vs Tree Search Player (d=3)	1000
Tree Search Player (d=3)	60% vs Value Function Player +	1000
Value Function Player +	70% vs Neural Network Player +	1000
Neural Network Player +	52% vs Value Function Player	1000
Value Function Player	99% vs Greedy VP Player	1000
Greedy VP Player	60% vs Weighted Random Player	1000
Weighted Random Player	55% vs Random Player	1000
Random Player	-	-
Neural Network Player	-	-

Agent	Pobjede – 4 igrača [%]	Broj igri
Tree Search Player + (d=3)	42%	1000
Tree Search Player (d=3)	22%	
Value Function Player +	26%	
Neural Network Player +	10%	

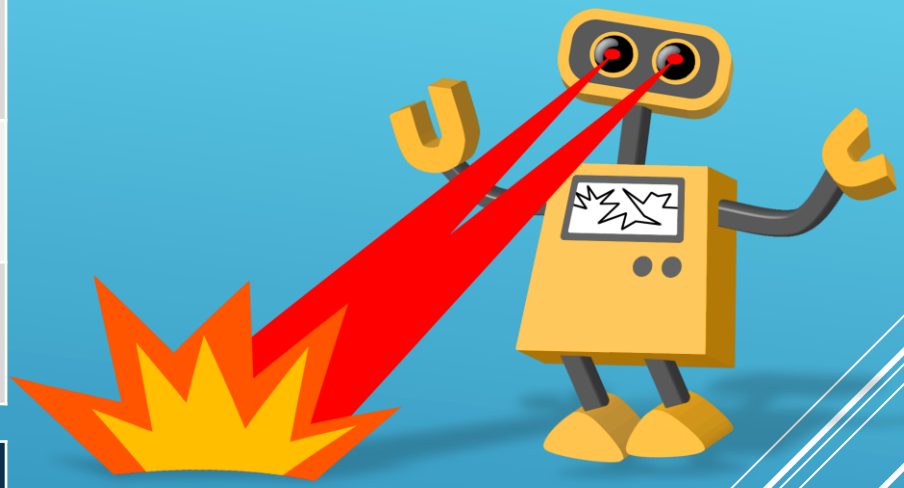


REZULTATI

Agent	Pobjede – 1v1 [%]	Pobjede – 2v2 [%]	Broj igri
Tree Search Player + (d=3)	48%	62%	200
Alpha-Beta Player	52%	38%	
Value Function Player +	45%	51%	1000
Catanatron VFP	55%	49%	
Neural Network Player +	100%	100%	1000
MCTS Player	0% (< 5 igri)	0% (< 5 igri)	

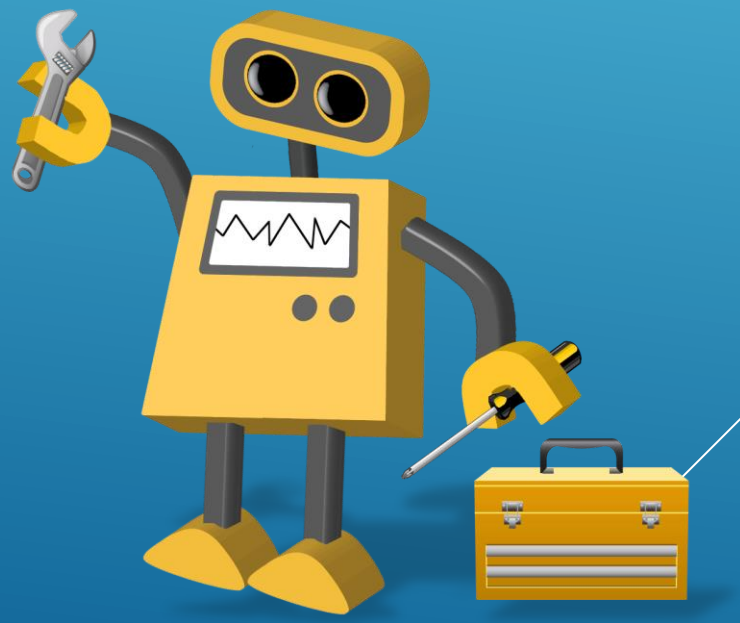
Agent	Pobjede – 4 igrača [%]	Broj igri
Tree Search Player + (d=3)	46%	200
Alpha-Beta Player	33%	
Catanatron VFP	21%	
MCTS Player	0%	

	WINS	AVG VP	AVG SETTLES	AVG CITIES	AVG ROAD	AVG ARMY	AVG DEV VP
TreeSearchPlayer:RED	91	8.11	2.16	1.34	0.57	0.53	1.06
AlphaBetaPlayer:BLUE(depth=2,value_fn=base_fn,prunning=False)	66	7.21	2.71	1.93	0.30	0.01	0.04
ValueFunctionPlayer:ORANGE(value_fn=base_fn)	43	6.60	2.73	1.75	0.12	0.01	0.12
MCTSPlayer:WHITE(10:False)	0	2.40	2.00	0.06	0.00	0.01	0.27



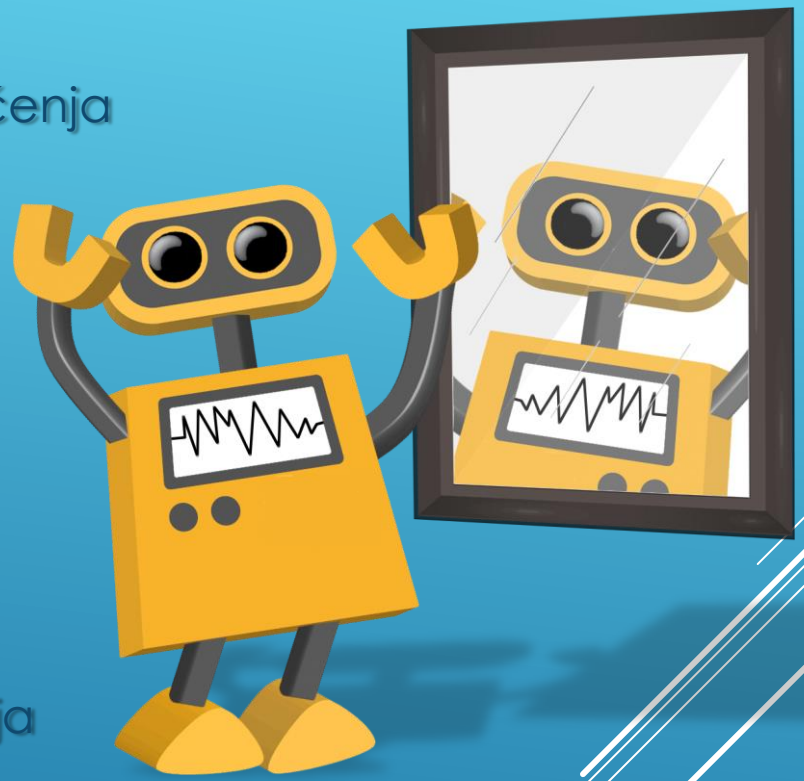
MOGUĆA POBOLJŠANJA

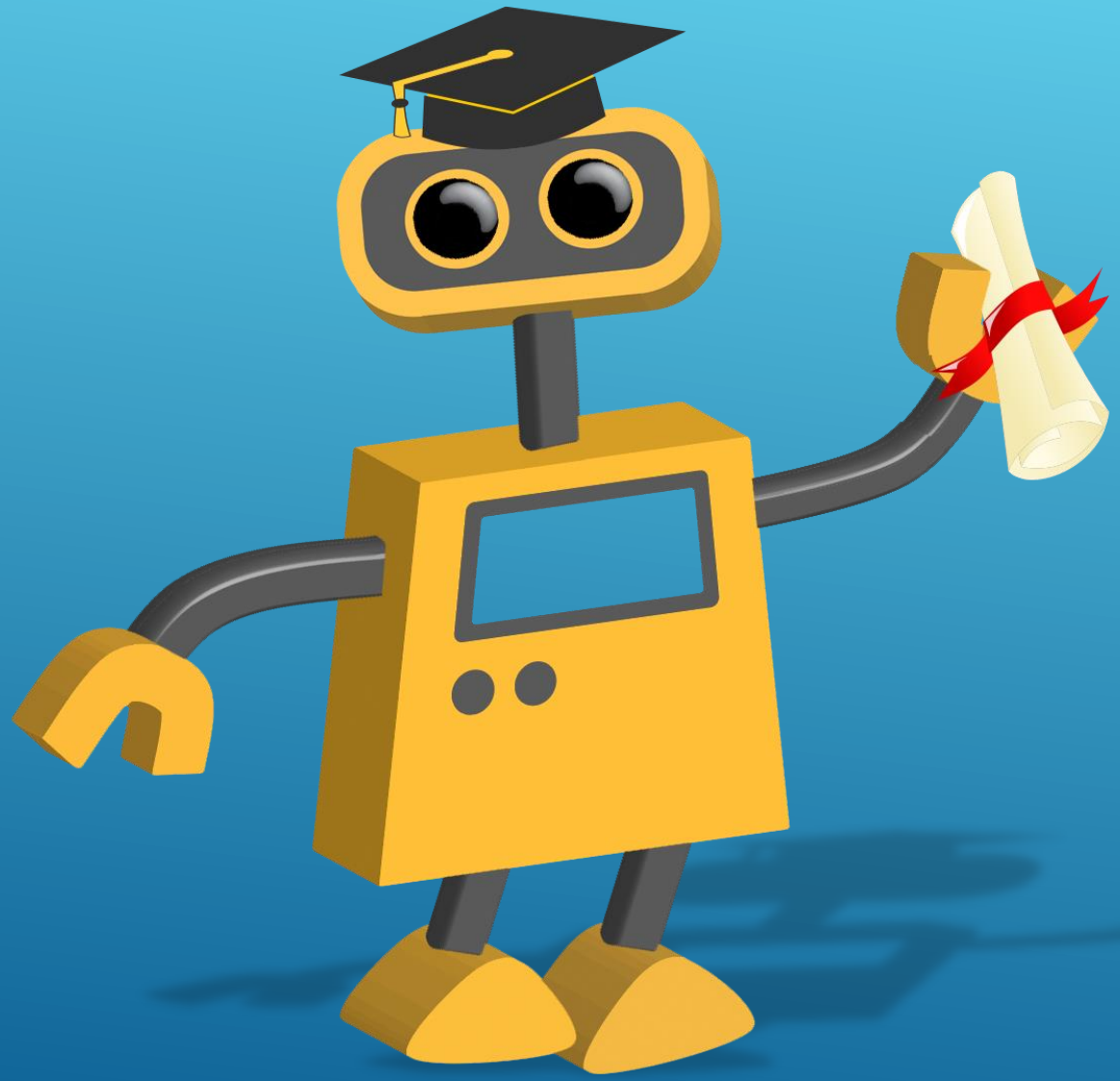
- Povećati nezavisnost varijabli heurističke funkcije (broj gradova i naselja → broj bodova)
- Podijeliti odabir akcije na zasebne strategije ovisno o vrsti akcije
- Dodavanje ulaza u heurističku funkciju:
  - Prosječna udaljenost između naselja, broj odvojenih nizova cesti…
- Podržano učenje:
  - *Q-learning, exploration & exploitation…*
  - Biblioteka Gym
- Učenje (GA) s različitim brojevima igrača



ZAKLJUČAK

- Društvene igre i videoigre:
  - Idealan izazov za razvoj umjetne inteligencije i strojnog učenja
  - Testiraju sva poželjna obilježja inteligencije
- Agenti za Catan:
  - Zadovoljavajući rezultati korištenjem heuristike i GA
  - Problem nedeterminizma
  - Bolje metode podržanog učenja?
- Iskustvo:
  - Stvaranje heuristike iz pravila igre i osobnog iskustva igranja
  - Učenje heurističkih modela genetskim algoritmom
  - Rad s bibliotekom TensorFlow, paralelno programiranje, razvoj web-aplikacija u Pythonu…





HVALA NA PAŽNJI!

Programsko rješenje nalazi se u javnom GitLab repozitoriju:
https://gitlab.com/iskoric/catanatron

Upute za korištenje:
catan_ai/README.md